Complexity Theory

NP-Complete Problems
May 16, 2022

iCIS | Software Science
Radboud University

Recap

More Satisfiability Problems

Integer Linear Programming

Graph Problems
- Clique
- Vertex Cover
- Coloring

This lecture

- Most important topic: **NP**-completeness proofs
- Details will be written on the board (not on the slides)
- When possible, I give references to the book

Outline

Recap

More Satisfiability Problems

Integer Linear Programming

Graph Problems
- Clique
- Vertex Cover
- Coloring

Nondeterministic Polynomial Decision Problems

Definition
An algorithm $A : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ **verifies** X if for all $w \in \{0,1\}^*$, we have $w \in X$ if and only if there is a $y \in \{0,1\}^*$ such that $A(w,y) = 1$. This y is called **the certificate**.

Definition
An algorithm $A : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ verifies X **in nondeterministic polynomial time** if A verifies X and if for its time complexity we have $T(n) = \mathcal{O}(n^k)$ for some k.

Definition
We say that $X \in$ **NP** if there is an algorithm A that verifies X in nondeterministic polynomial time.

Reducibility

Definition
We say that A **(polynomially) reduces to** B if we have a function $f : \{0,1\}^* \to \{0,1\}^*$ such that

- $f \in \mathcal{O}(n^k)$ for some k;
- for all $w \in \{0,1\}^*$ we have $w \in A$ if and only if $f(w) \in B$.

If this is the case, we write $A \leq_P B$.

NP-hard and NP-complete

Definition
A decision problem B is said to be **NP**-**hard** if for all decision problems $A \in \mathbf{NP}$ we have $A \leq_P B$.

Definition
A decision problem B is said to be **NP**-**complete** if $B \in \mathbf{NP}$ and if B is **NP**-hard.

Theorem
If A reduces to B, A is **NP***-hard, then B is* **NP***-hard*

SAT

The **Boolean satisfiability problem**, also known as SAT is the following problem

> Given a formula φ, is φ satisfiable? So: is there an assignment of atoms to truth values under which φ is true?

We already saw: SAT is in **NP**.
In fact: SAT is **NP**-complete.

Variations of SAT

Definition
A formula is in **conjunctive normal form** if it is written as conjunction of disjunctions of possibly negated atoms.

Examples:

- $(A \vee \neg B \vee C) \wedge (A \vee C) \wedge (A \vee B \vee \neg B)$
- $A \vee \neg B \vee C$

Not in CNF:

- $(A \vee \neg B \wedge C) \wedge (A \vee C)$
- $A \vee \neg(B \wedge C) \vee C$

SAT for formulas in CNF is **NP**-complete.

Outline

Recap

More Satisfiability Problems

Integer Linear Programming

Graph Problems
- Clique
- Vertex Cover
- Coloring

How to prove that a problem is NP-complete?

How to prove that a problem is NP-complete?

To show that A is **NP**-complete, take the following steps:

(i) Show that A is in **NP**.

(ii) Pick a decision problem C of which you know that it is **NP**-complete.

(iii) Prove that $C \leq_P A$.

At most 3-conjunctive normal form

Definition

A formula is in **at most 3-conjunctive normal form** if it can be written as a conjunction of disjunctions of literals, and each disjunction has at most 3 literals.

Example:

$$(A \vee \neg B \vee C) \wedge (\neg A \vee B) \wedge \neg C$$

Not an example:

$$(A \vee \neg B \vee C \vee D \vee \neg E) \wedge (\neg A \vee B) \wedge \neg C$$

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

We look at the decision problem $\leq_3$CNF

> Given a formula in at most 3-conjunctive normal form, is it satisfiable?

Theorem

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

Exactly 3-conjunctive normal form

Definition

A formula is in **exactly 3-conjunctive normal form** if it can be written as a conjunction of disjunctions of literals, and each disjunction has exactly 3 literals.

Example:

$$(A \vee \neg B \vee C) \wedge (\neg A \vee B \vee \neg D) \wedge (\neg A \vee \neg B \vee \neg C)$$

Not an example:

$$(A \vee \neg B \vee C) \wedge (\neg A \vee B) \wedge \neg C$$

3CNF **is NP-complete**

We look at the decision problem 3CNF

> Given a formula in exactly 3-conjunctive normal form, is it satisfiable?

Theorem

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

Theorem 34.10 in “Introduction to Algorithms”.

Outline

Integer Linear Programming

Integer linear programming:

> Given a finite set of variables and a finite set of inequalities using those variables, is there a solution that satisfies all the inequalities?

Examples of Integer Linear Programming

Can we find integers x_1, x_2, x_3 such that

$$\begin{array}{rcl} 3 \cdot x_1 + 5 \cdot x_2 - 2 \cdot x_3 & \leq & 7 \\ 8 \cdot x_1 + 2 \cdot x_2 - 4 \cdot x_3 & \leq & 4 \\ x_1 + x_2 + x_3 & \leq & 37 \end{array}$$

Examples of Integer Linear Programming

Can we find integers x_1, x_2, x_3 such that

$$\begin{array}{rcr} 3 \cdot x_1 + 5 \cdot x_2 - 2 \cdot x_3 & \leq & 7 \\ 8 \cdot x_1 + 2 \cdot x_2 - 4 \cdot x_3 & \leq & 4 \\ x_1 + x_2 + x_3 & \leq & 37 \end{array}$$

Solution $x_1 = x_2 = x_3 = 0$.

Examples of Integer Linear Programming

Can we find integers x_1, x_2, x_3 such that

$$\begin{aligned} 3 \cdot x_1 + 5 \cdot x_2 - 2 \cdot x_3 &\leq 7 \\ 8 \cdot x_1 + 2 \cdot x_2 - 4 \cdot x_3 &\leq 4 \\ x_1 + x_2 + x_3 &\leq 37 \end{aligned}$$

Solution $x_1 = x_2 = x_3 = 0$.

Can we find integers x_1 and x_2 such that

$$\begin{aligned} 2 \cdot x_1 + 3 \cdot x_2 &\geq 37 \\ x_1 + x_2 &\leq 3 \\ x_1 &\geq 1 \\ x_2 &\geq 1 \end{aligned}$$

Examples of Integer Linear Programming

Can we find integers x_1, x_2, x_3 such that

$$\begin{array}{rcl} 3 \cdot x_1 + 5 \cdot x_2 - 2 \cdot x_3 & \leq & 7 \\ 8 \cdot x_1 + 2 \cdot x_2 - 4 \cdot x_3 & \leq & 4 \\ x_1 + x_2 + x_3 & \leq & 37 \end{array}$$

Solution $x_1 = x_2 = x_3 = 0$.

Can we find integers x_1 and x_2 such that

$$\begin{array}{rcl} 2 \cdot x_1 + 3 \cdot x_2 & \geq & 37 \\ x_1 + x_2 & \leq & 3 \\ x_1 & \geq & 1 \\ x_2 & \geq & 1 \end{array}$$

No solution!

NP-completeness of ILP

We look at the decision problem ILP

> Given an integer linear program, does it have a solution?

Theorem

ILP *is* **NP***-complete.*

Exercises 34.5-2 and 34.5-3 in the book.

Outline

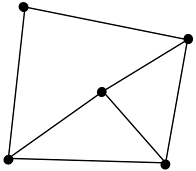
Cliques

Definition
Let $G = (V, E)$ be a graph. A **clique** in G is a subset $V' \subseteq V$ such that for every distinct $v, v' \in V'$ there is an edge between v and v'.

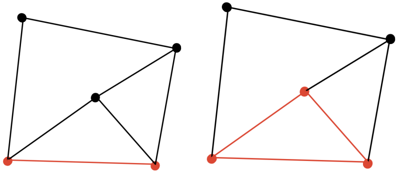
Briefly, a clique is a subgraph in which every two distinct vertices are adjacent.

Example of cliques

Consider the following graph:



Some cliques in that graph:



Clique **is NP-complete**

We look the following decision problem Clique:

> Given a graph G and an integer k, does G have a clique with k vertices?

Theorem

Clique *is* **NP***-complete.*

Theorem 34.11 in "Introduction to Algorithms".

Illustration of the proof

Consider $(a \vee b \vee c) \wedge (\neg a \vee \neg b \vee c) \wedge (\neg a \vee b \vee c)$.

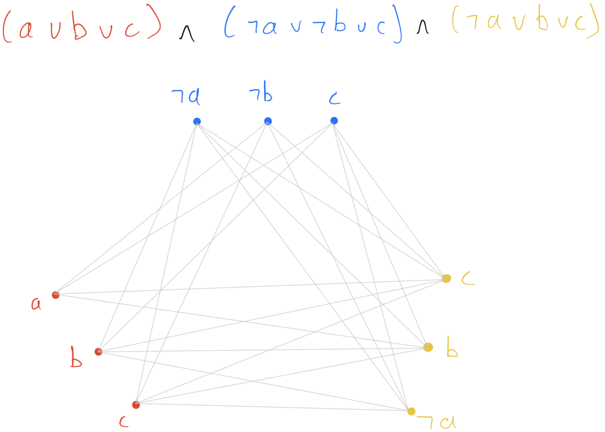
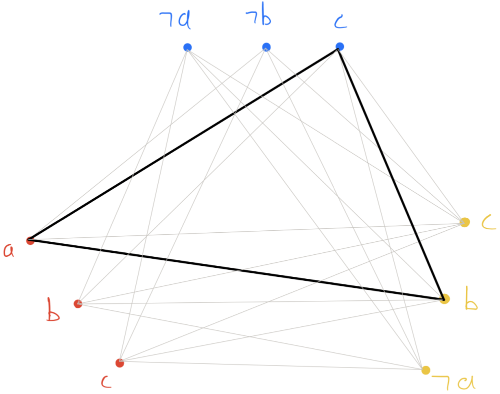


Illustration of the proof

Pick a from $(a \vee b \vee c)$, b from $(\neg a \vee b \vee c)$, and c from $(\neg a \vee \neg b \vee c)$. We get the following clique

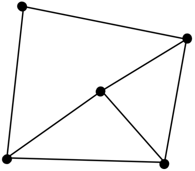


Vertex Covers

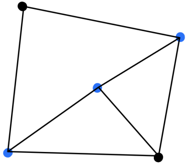
Definition
Let $G = (V, E)$ be a graph. A **vertex cover** of G is a subset $V' \subseteq V$ of vertices such that every edge has at least 1 endpoint in V'.

Example of a vertex cover

Consider the following graph:



A vertex cover in that graph:



VertexCover **is NP-complete**

We look at the decision problem VertexCover

> Given a graph G and an integer k, does G have a vertex cover with k vertices?

Theorem

VertexCover *is* **NP***-complete.*

Theorem 34.12 in “Introduction to Algorithms”.

Illustration of the proof

Consider the following graph (clique in blue):

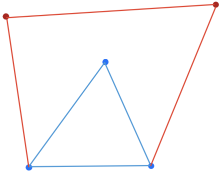
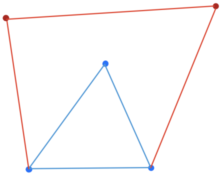
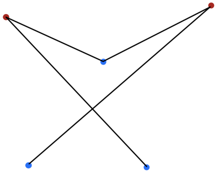


Illustration of the proof

Consider the following graph (clique in blue):



From the construction, we get (vertex cover in red):

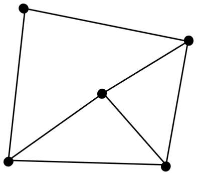


Coloring

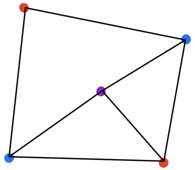
Definition
Let $G = (V, E)$ be a graph. A **3-coloring** is a function $c : V \to \{b, p, r\}$ such that for every edge e between v and w we have $c(v) \neq c(w)$.

Example of a coloring

Consider the following graph:



A 3-coloring of that graph:



3Color **is NP-complete**

We look at the decision problem 3Color

> Given a graph G, does G have a 3-coloring?

Theorem
3Color *is* **NP***-complete.*

See the note!