

CMSC 451 - Algorithm Design

Lecture 16 - NP-Completeness: 3SAT + Independent Set

Recap:

P: Set of languages (decision probs.)
solvable in poly-time

NP: Set of languages verifiable in poly-time

$\leq_p$: $L_1 \leq_p L_2$ means there is a poly-time function f s.t.

$$\forall x \quad x \in L_1 \text{ iff } f(x) \in L_2$$

NP-hard: L is NP-hard if $\forall L' \in \text{NP}, L' \leq_p L$

NP-complete: L is NP-complete if

(1) $L \in \text{NP}$

(2) L is NP-hard

If any NP-complete problem is solvable in poly-time, they all are ($P = \text{NP}$)

If any problem in NP is not solvable in poly-time, then no NP-complete problem is solvable in poly-time ($P \neq \text{NP}$)

Boolean Satisfiability + Cook's Theorem:

- Prove problems NP-complete through reductions
- Need one initial NP-complete problem to start

Boolean Satisfiability (SAT)

Boolean formula:

variables: x, y, z or $x_1, x_2, \dots$

→ Each can be assigned true or false

operations:

$\wedge$ - "and" $x \wedge y$ - true if both true

$\vee$ - "or" $x \vee y$ - true if either/both true

$\neg$ - "not" $\neg x$ - reverses true $\leftrightarrow$ false

often written: $\bar{x} \equiv \neg x$

Examples:

$$F_1(x, y, z) = (x \wedge (y \vee \bar{z})) \wedge ((\bar{y} \wedge \bar{z}) \vee \bar{x})$$

$x = T$
 $y = z = F$

$$F_2(x, y) = (\bar{z} \vee x) \wedge (z \vee y) \wedge (\bar{x} \wedge \bar{y})$$

Def: A formula is satisfiable if it is possible to assign values to variables so it evaluates to true

Not
satisfiable

Cook's Theorem: SAT is NP-Complete

The proof is long & technical, but here is a high-level intuitive overview

SAT $\in$ NP: Given a formula $F(x_1, \dots, x_n)$
certificate is assignment to variables

$x_1 \leftarrow T, x_2 \leftarrow F, x_3 \leftarrow T, \dots$

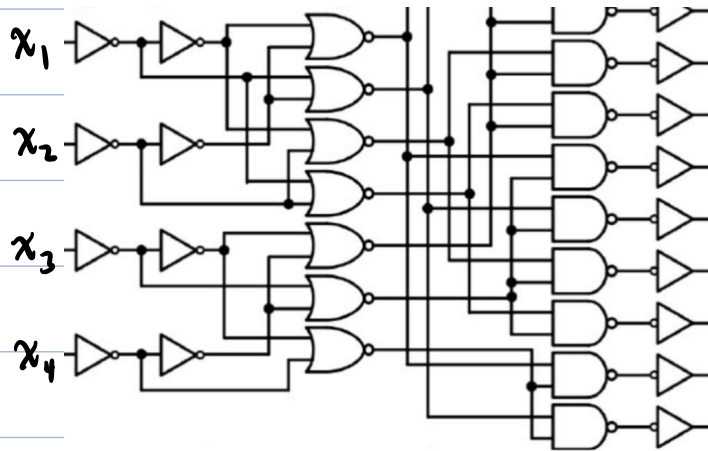
To verify, plug these into formula
evaluate & check that result = true

SAT is NP-hard:

- Given any language $L \in \text{NP}$,
want to show $L \leq_p \text{SAT}$ ($x \in L$ iff $f(x) \in \text{SAT}$)
- For any instance x for L we know
 - Can verify that $x \in L$
 - Involves a certificate + verification program
 - Encode certificate as a bit string
e.g. $C = 0101101\dots$
 - Express bits as variables
 - $x_1 = 1^{\text{st}}$ bit of C ($0 = F, 1 = T$)
 - $x_2 = 2^{\text{nd}}$ bit of C
 - $\vdots$

- Verification is a poly-time program.

- Program can be expressed as a circuit of polynomial size



- Circuit can be expressed as boolean formula $f(x)$ of polynomial size.

- This formula faithfully simulates the verification program:

$$x \in L \text{ iff } f(x) \in \text{SAT}$$

$$\therefore \forall L \in \text{NP} \quad L \leq_p \text{SAT} \Rightarrow \text{SAT is NP-hard!}$$

□

Cook proved a stronger result -

Satisfiability is NP-complete, even if we restrict the format of the formulas to 3CNF.

Notation:

literal: A variable or its complement, x_i or $\bar{x}_i$

3-conjunctive normal form (3CNF):

A formula that is the conjunction ($\wedge$) of clauses, each of which is the disjunction ($\vee$) of 3 literals.

Example:

$$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge \dots$$

joined by "ands"

clause = "or" of 3 literals

Def: 3SAT - Satisfiability of formulas expressed in 3CNF form.

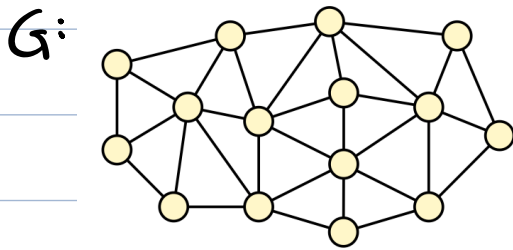
Theorem (Cook). 3SAT is NP-complete

Reductions are simpler

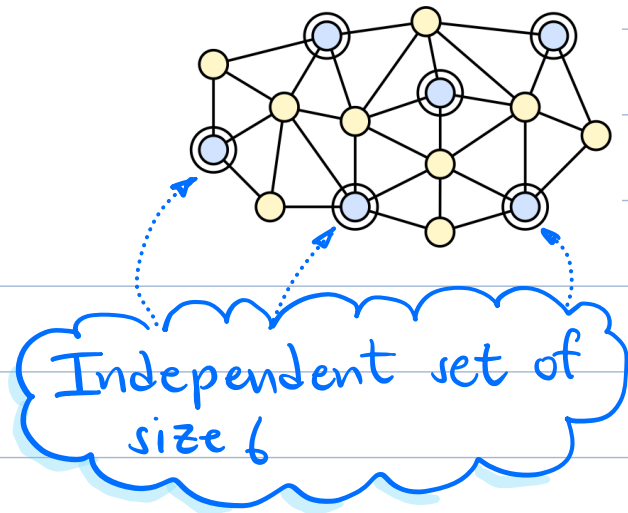
Given that 3SAT is NP-Complete, let's try another.

Independent Set (IS): Given a graph $G = (V, E)$ + integer k , does G contain a subset of vertices $V' \subseteq V$ of size k so that no two are adjacent?

Example:



$(G, 6) \in IS$
 $(G, 7) \notin IS$



Claim: IS is NP-Complete

Need to show: (1) $IS \in NP$ (verifiable)

(2) IS is NP-hard

(1) $IS \in NP$:

Given instance (G, k) , the certificate consists of a set of k vertices of G .

We consider each pair u, v from this set & check that they are not adjacent.

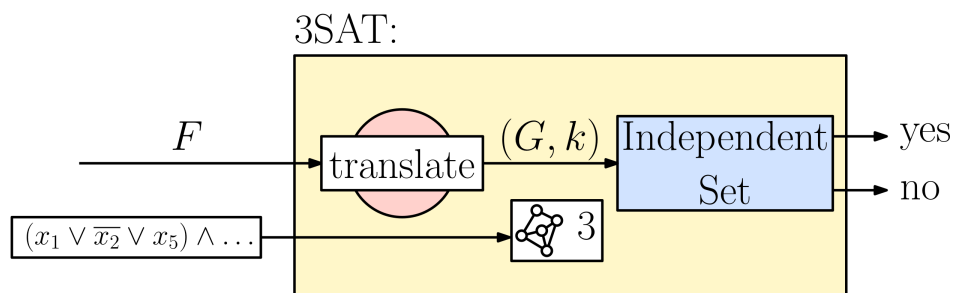
If so, we accept the certificate & otherwise we reject it.

(2) IS is NP-hard:

We'll show that: $3SAT \leq_p IS$

There exists poly-time function f that maps a 3CNF formula F to a pair (G, k) s.t.

$F \in 3SAT$ iff $(G, k) \in IS$



How? These problems seem unrelated!



Let's see what they share in common:

Selection:

3SAT: Which variables are true
≡ which literals are true

IS: Which vertices are in indep. set

literals map to
vertices?

Requirements:

3SAT: Logical "and" of clauses, each
is "or" of 3 literals

⇒ at least one true literal per clause

IS: V' must contain k vertices

set k = number
of clauses

Restrictions:

3SAT: $x_i + \bar{x}_i$ cannot both be true

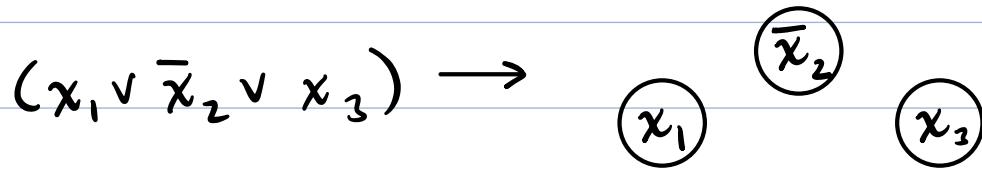
IS: If $(u, v) \in E$, $u + v$ cannot
both be in indep. set

Create edge between

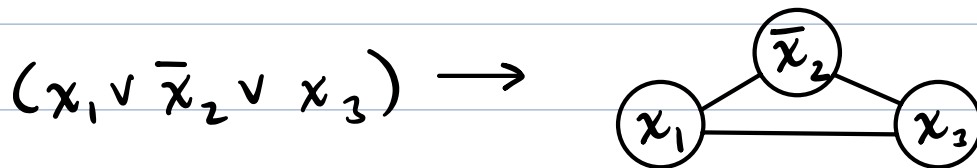


Plan:

- For each clause, create a vertex for each literal. (Clusters of 3 vertices)

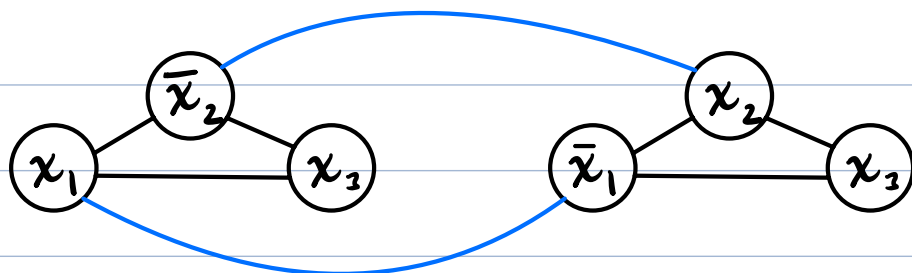


- Connect vertices with each cluster (Forces us to select only one as the "satisfying" literal)



- Connect each vertex x_i with its complements $\bar{x}_i$ (conflict links)
(Forces us to choose one or other to be true)

$$(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



Reduction: Given formula F in 3CNF:

$k \leftarrow$ number of clauses

for each clause $(x_a \vee x_b \vee x_c)$:

- create vertices (x_a) (x_b) (x_c)

- create edges (x_a, x_b) , (x_b, x_c) , (x_c, x_a)

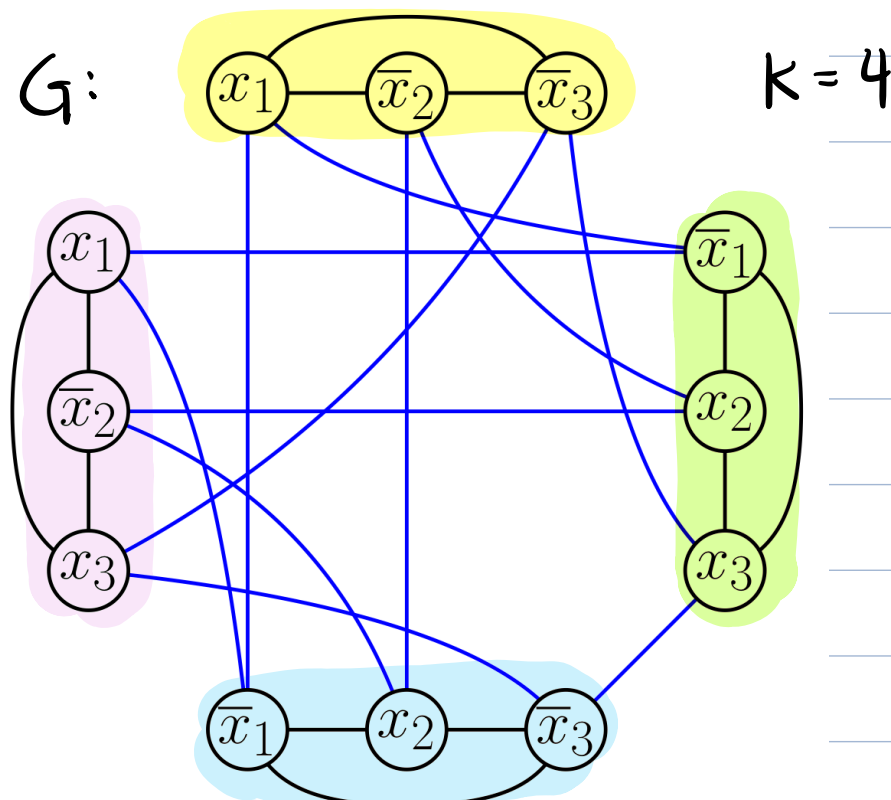
for each variable x_i :

- create edges between all copies of (x_i) and $(\bar{x}_i)$

return (G, k)

Example: $F =$

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$$



This can be computed in poly-time in size of formula.

(Exercise)

Why does it work? Some intuition...

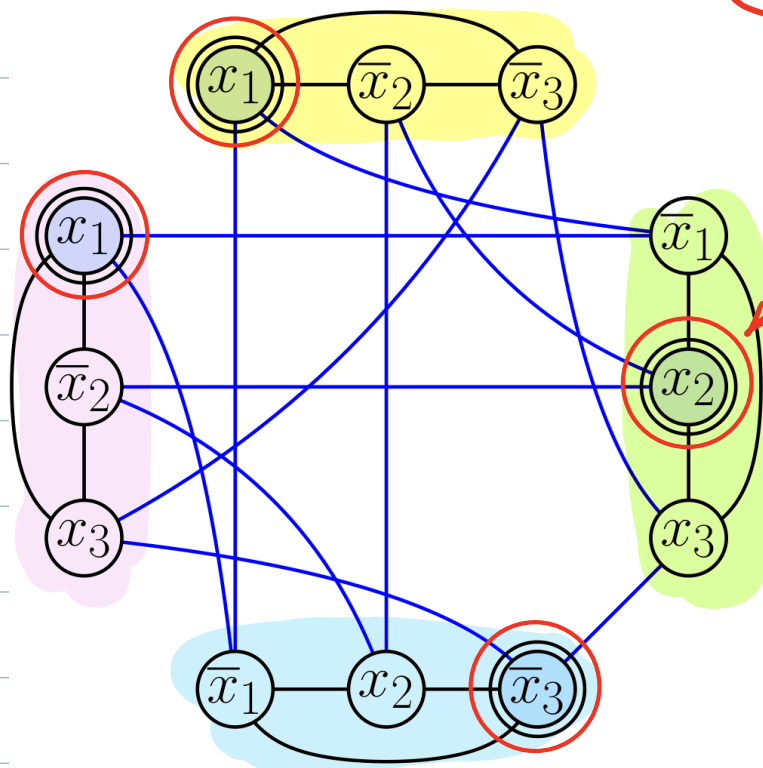
$\Rightarrow$ If F is satisfiable, there must be at least one true literal per clause $\rightarrow$ add it to indep. set

$\Leftarrow$ If G has indep. set of size k , must be one vertex from each cluster. Set this literal true.

This assignment satisfies F .

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$$

$$\begin{array}{l} x_1 = T \\ x_2 = T \\ x_3 = F \end{array}$$



There may be choices on which vertex to use

Circled vertices form an indep. set size $k=4$

Formal proof:

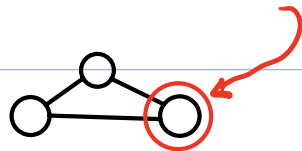
Claim: F is satisfiable iff G has ind. set size k

($\Rightarrow$) If F is satisfiable then (because F in CNF form) each clause of F must have at least one true literal.

- Select one true literal from each clause and add to V'
 - There are k clauses $\Rightarrow |V'| = k$
 - Both (x_i) and $(\bar{x}_i)$ cannot be true, so no edge between vertices of V'
- $\Rightarrow V'$ is ind. set of size k ✓

($\Leftarrow$) If G has ind. set V' of size k

- Because clusters are connected, can take at most one from each cluster.



- Because $k = \text{no. of clauses} = \text{no. of clusters}$ must take exactly one from each cluster.

- Because $(x_i) \text{ --- } (\bar{x}_i)$ are adjacent, both cannot be in V'
 - if $x_i \in V' \rightarrow \text{set } x_i \leftarrow T$
 - $\bar{x}_i \in V' \rightarrow \text{set } x_i \leftarrow F$
 - if neither $\rightarrow \text{set } x_i \leftarrow \text{either } T \text{ or } F$
 - This assignment satisfies every clause of F
- $\Rightarrow F$ has a satisfying assignment $\checkmark$
- $\square$

Done! Showed IS is NP-complete.

Notes:

- Hardest part is the reduction
- Reduce known NP-hard to our problem
- Reduction merely translates common elements.
- It is agnostic (doesn't care) which variables are true, which vertices are in ind. set or even if an ind. set exists!

Summary:

- 3SAT is NP-complete
- IS is NP-complete ($3SAT \leq_p IS$)

