Monash University
Faculty of Information Technology

FIT2014 Theory of Computation

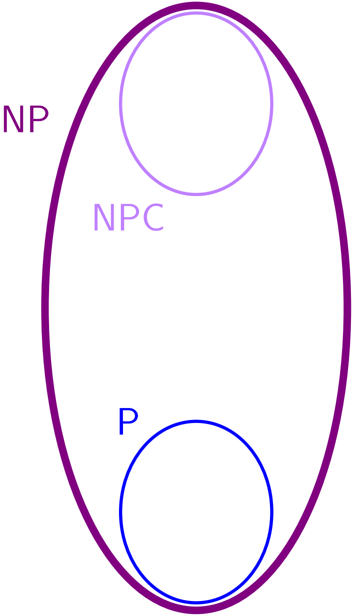
Lecture 30
Proving NP-completeness

slides by Graham Farr

Overview

- Using NP-complete problems to show that other problems are NP-complete.
- 3-SAT is NP-complete.
- VERTEX COVER is NP-complete.
- INDEPENDENT SET is NP-complete.
- CLIQUE is NP-complete.
- Dealing with NP-completeness.

If $P \neq NP$:



In NP, not known to be in P:

SATISFIABILITY, 3-SAT,
HAMILTONIAN CIRCUIT,
3-COLOURABILITY,
VERTEX COVER,
INDEPENDENT SET, . . .

GRAPH ISOMORPHISM,

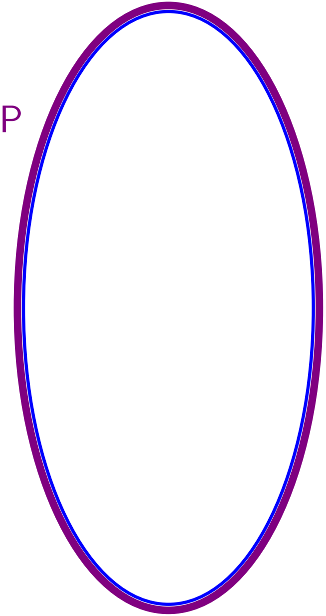
INTEGER FACTORISATION, . . .

In P:

2-SAT,
EULERIAN CIRCUIT,
2-COLOURABILITY,
CONNECTED GRAPHS,
SHORTEST PATH,
PRIMES,
Invertible matrices,
. . . ,
All Context-Free Languages,
All Regular Languages.

If P = NP:

If P = NP



SATISFIABILITY, 3-SAT,
HAMILTONIAN CIRCUIT,
3-COLOURABILITY,
VERTEX COVER,
INDEPENDENT SET, ...

GRAPH ISOMORPHISM,

INTEGER FACTORISATION, ...

2-SAT,
EULERIAN CIRCUIT,
2-COLOURABILITY,
CONNECTED GRAPHS,
SHORTEST PATH,
PRIMES,
Invertible matrices,
...,
All Context-Free Languages,
All Regular Languages.

Proving NP-completeness

Now that we have one NP-complete language,
it is much easier to prove NP-completeness for many other languages.

Theorem.
If K is NP-complete, L is in NP, and $K \leq_P L$, then L is NP-complete.

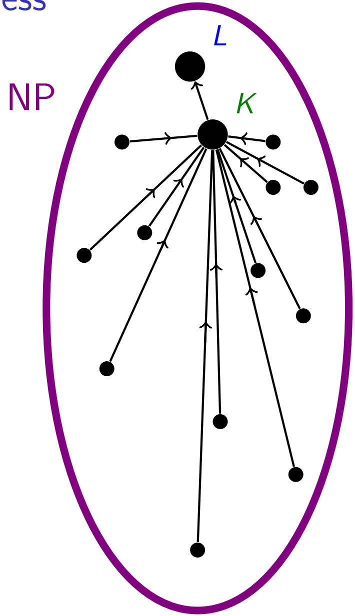
Proof.
We are given that L is in NP.

Let H be any language in NP.

Then $H \leq_P K$ (by NP-completeness of K),
and $K \leq_P L$ (given),
so by transitivity of $\leq_P$, we conclude $H \leq_P L$.

Therefore L is NP-complete. $\square$

NP-completeness



Proving NP-completeness

So, to prove a language L is NP-complete, it is sufficient to

(a) show L is in NP, *and*

(b) show that another NP-complete language polynomial-time-reduces to L.

just one reduction!

This approach didn't help us show SAT is NP-complete, since at that stage, we didn't know of any other NP-complete languages.

But now we can use SAT to show that other languages are NP-complete.

This approach was first taken by R. Karp (1972).

Proving NP-completeness: 3-SAT

3-SAT

(a) belongs to NP

(b) SAT $\leq_P$ 3-SAT

Given a Boolean expression φ in CNF:

Each clause of φ is to be replaced by a clause, or clauses, of size 3, to do the same job.

$$(x) \longmapsto \begin{array}{l} (x \vee w_{i1} \vee w_{i2}) \\ \wedge\ (x \vee w_{i1} \vee \neg w_{i2}) \\ \wedge\ (x \vee \neg w_{i1} \vee w_{i2}) \\ \wedge\ (x \vee \neg w_{i1} \vee \neg w_{i2}) \end{array}$$

. . . where w_{i1}, w_{i2} appear nowhere else

$$(x_1 \vee x_2) \longmapsto \begin{array}{l} (x_1 \vee x_2 \vee w_i) \\ \wedge\ (x_1 \vee x_2 \vee \neg w_i) \end{array}$$

. . . where w_i appears nowhere else

$(x_1 \vee x_2 \vee x_3) \longmapsto$ itself

Proving NP-completeness: 3-SAT

continued:

$$(\, x_1 \vee x_2 \vee x_3 \vee x_4 \,) \quad \longmapsto \quad (\, x_1 \vee x_2 \vee z_1 \,) \wedge (\, \neg z_1 \vee x_3 \vee x_4 \,)$$

$$(\, x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \,) \quad \longmapsto \quad (\, x_1 \vee x_2 \vee z_1 \,) \wedge (\, \neg z_1 \vee x_3 \vee z_2 \,) \wedge (\, \neg z_2 \vee x_4 \vee x_5 \,)$$

. . . *etc*,

where each z_j appears nowhere else.

So 3-SAT is NP-complete.

VERTEX COVER

We now show that VERTEX COVER is NP-complete.

Easy to show it's in NP.

To prove completeness, we show that

$$\text{3-SAT} \quad \leq_P \; \text{VERTEX COVER.}$$

VERTEX COVER

Given a Boolean expression φ in CNF with exactly 3 literals in each clause, we must show how to construct a graph G_φ and positive integer k_φ such that

$$\varphi \in \text{3-SAT} \quad \text{if and only if} \quad (G_\varphi, k_\varphi) \in \text{VERTEX COVER}$$

Suppose φ has

- variables $x_1, \ldots, x_n$
- clauses $C_1, \ldots, C_m$

VERTEX COVER

Construction of G_φ:

Vertices:

- one for each literal:

$$x_1, \neg x_1, \ldots, x_n, \neg x_n$$

- three for each clause:

$$C_{11}, C_{12}, C_{13}, \quad C_{21}, C_{22}, C_{23}, \ldots, C_{m1}, C_{m2}, C_{m3}$$

Edges:

- Join each literal to its "partner":

$$(x_1, \neg x_1), \ldots, (x_n, \neg x_n)$$

- Join up all vertices corresponding to a clause:

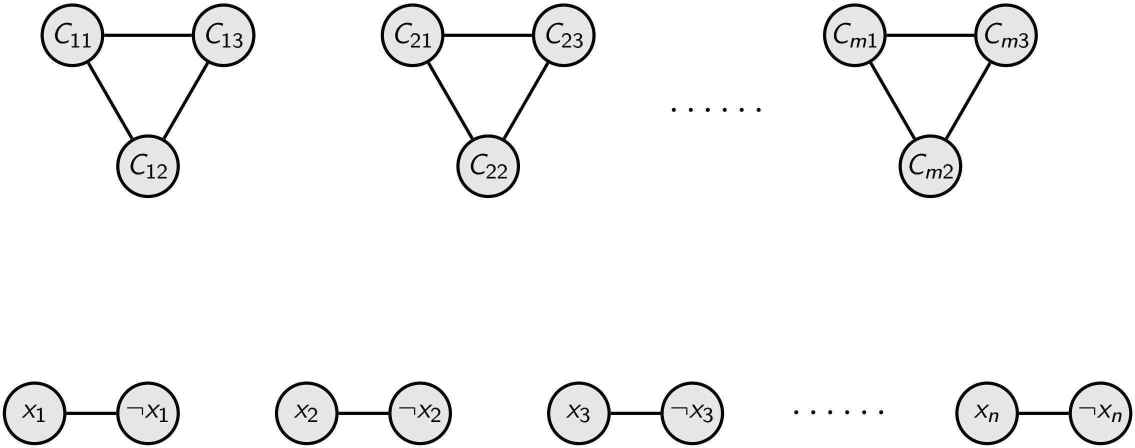
$$(C_{11}, C_{12}), (C_{11}, C_{13}), (C_{12}, C_{13}), \ldots\ldots$$

...so each clause is a separate triangle.

Proving NP-completeness

Looking at the graph so far:
How large must a vertex cover be?

The story so far:



VERTEX COVER

Observations:

Each "variable-edge" must be covered, by at least one vertex of the VC.

Each "clause-triangle" must be covered, by at least two vertices of the VC.

All variable-edges and clause-triangles are disjoint from each other.

So all vertex covers must have size $\geq 2m + n$.

Set $k_\varphi := 2m + n$.

This forces the vertex cover to have exactly one vertex from each variable-edge and exactly two vertices from each clause-triangle.

VERTEX COVER

For each position in each clause:

- Add an edge
  from the vertex representing the clause position
  to the vertex representing the corresponding literal.
- So, for the literal in the j-th position of the i-th clause:
  - If the literal is x_k, then add the edge (C_{ij}, x_k).
  - If the literal is $\neg x_k$, then add edge $(C_{ij}, \neg x_k)$.

This completes construction of G_φ.

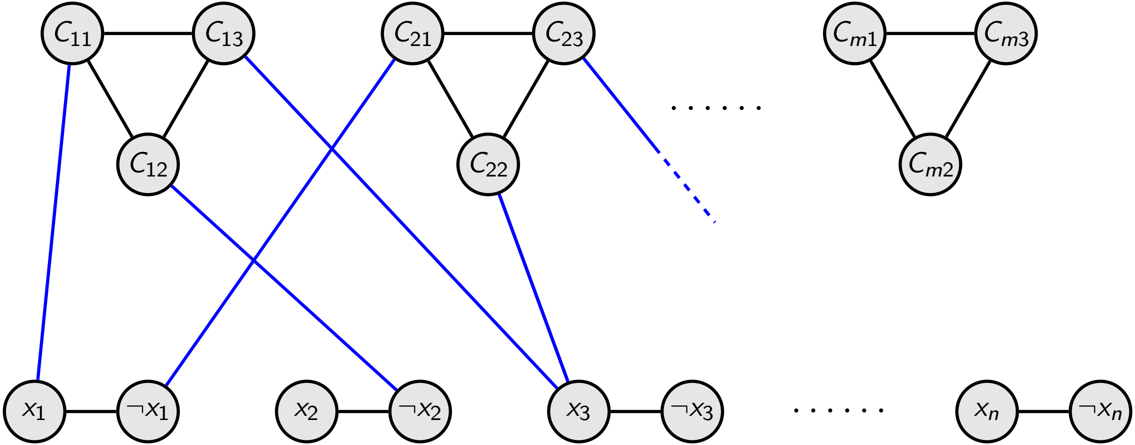
For example: if

$$\varphi \; = \; (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge \cdots$$

then ...

VERTEX COVER

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge \cdots$$



VERTEX COVER

For each variable-edge,

choosing which endvertex is in the VC $\longleftrightarrow$ assigning a truth value to that variable

vertex is chosen $\longleftrightarrow$ literal is True

The chosen literal covers all edges going from it up to the clauses.

So, if literal x_k is True and it appears in position j in clause C_i,
then the edge (x_k, C_{ij}) is covered by x_k
and does not need to be covered again in the clause-triangle for C_i .

So the vertex C_{ij} does not need to be in the vertex cover;
the clause-triangle for C_i can be covered by its other two vertices.

VERTEX COVER

So every clause containing a true literal is easily covered by two vertices.

Conversely, if a clause-triangle only has two vertices from the vertex cover,
then the other vertex (not in the VC) gives the position of a literal which must be covered.

The current truth assignment is satisfying
if and only if every clause has a true literal
if and only if the vertex cover only meets each clause-triangle twice
if and only if the vertex cover has size $\leq k_\varphi$.

VERTEX COVER

So we have:

φ is satisfiable if and only if G_φ has a vertex cover of size $\leq k_\varphi$.

It remains to show that the reduction is polynomial time.
This is fairly routine.

Proving NP-completeness

So we now have three NP-complete problems:

SAT, 3-SAT, VERTEX COVER.

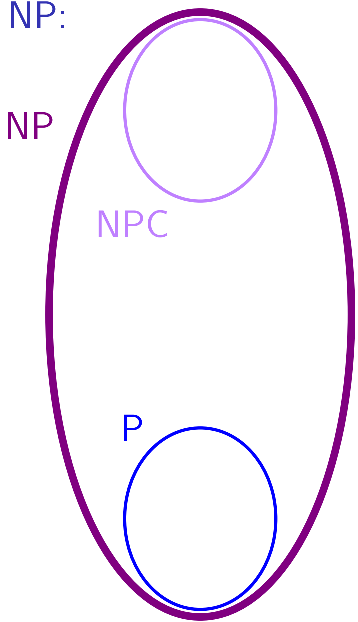
We saw in Lecture 26 (polynomial-time reductions) that

$$\begin{aligned} \text{VERTEX COVER} &\leq_P \text{INDEPENDENT SET} \\ &\leq_P \text{CLIQUE} \end{aligned}$$

so these two languages are NP-complete too.

Good exercise:
Prove that 3-COLOURABILITY is NP-complete, by reduction from INDEPENDENT SET.

If $P \neq NP$:



NP-complete:

SATISFIABILITY, 3-SAT,
HAMILTONIAN CIRCUIT,
3-COLOURABILITY,
VERTEX COVER,
INDEPENDENT SET, ...

In NP, not known to be in P or NP-complete:

GRAPH ISOMORPHISM,
INTEGER FACTORISATION, ...

In P:

2-SAT,
EULERIAN CIRCUIT,
2-COLOURABILITY,
CONNECTED GRAPHS,
SHORTEST PATH,
PRIMES,
Invertible matrices,
...,
All Context-Free Languages,
All Regular Languages.

Implications of NP-completeness

Showing that a language is NP-complete does not make it go away!
You may still need to find an algorithm for it.
NP-completeness is *evidence* that you won't find an algorithm that is

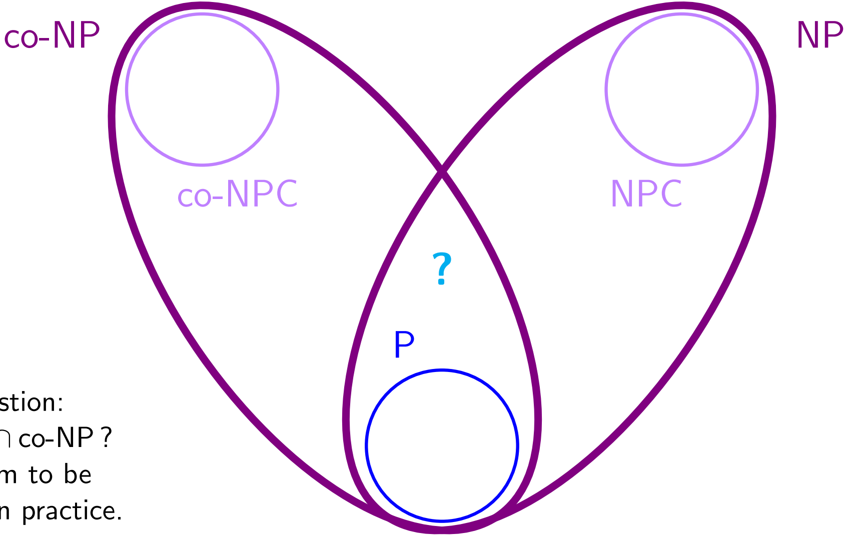
- **efficient**,
- **deterministic**,
- works in **all cases**, *and*
- solves the problem **exactly**.

So you have several options:

a **slow** algorithm (exponential time)	not efficient
randomised algorithm	not deterministic
an algorithm for **special cases**	not all cases
approximation algorithm	not exact

maybe in future: quantum computer?

If $\quad P \neq NP \quad$ and $\quad NP \neq$ co-NP:



Another open question:
Does $\quad P = NP \cap$ co-NP ?
There doesn't seem to be much difference, in practice.

Revision

Things to think about:

- How to show that 4-SAT is NP-complete?
- How to show that 3-COLOURABILITY is NP-complete?
- What about the complexity of the following problem?

  VACCINATION
  INPUT: Graph G, positive integers v, k.
  QUESTION: Can you "vaccinate" v vertices of G so that all connected unvaccinated subgraphs have $\leq k$ vertices?

Reading: Sipser, sections 7.4, 7.5.