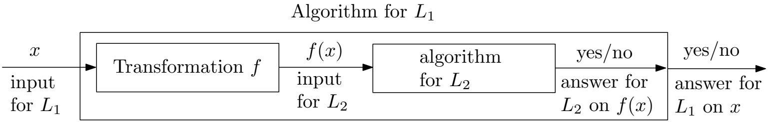
Lecture 22: NP-Completeness

Polynomial-Time Reductions

- Intuitively, $L_1 \leq_P L_2$ means L_1 is no harder than L_2.
- Given an algorithm A_2 for the decision problem L_2, we can develop an algorithm A_1 to solve L_1:



- If A_2 is polynomial-time algorithm, so is A_1.

Theorem

If $L_1 \leq_P L_2$ and $L_2 \in \mathbf{P}$, then $L_1 \in \mathbf{P}$.

Reduction between Decision Problems

Lemma (Transitivity of the relation $\leq_P$)

If $L_1 \leq_P L_2$ and $L_2 \leq_P L_3$, then $L_1 \leq_P L_3$.

Proof.

- Since $L_1 \leq_P L_2$, there is a polynomial-time reduction f_1 from L_1 to L_2.
- Similarly, since $L_2 \leq_P L_3$, there is a polynomial-time reduction f_2 from L_2 to L_3.
- Note that $f_1(x)$ can be calculated in time polynomial in $size(x)$. In particular this implies that $size(f_1(x))$ is polynomial in $size(x)$. $f(x) = f_2(f_1(x))$ can therefore be calculated in time polynomial in $size(x)$.
- Furthermore x is a yes-input for L_1 if and only if $f(x)$ is a yes-input for L_3 (why). Thus the combined transformation defined by $f(x) = f_2(f_1(x))$ is a polynomial-time reduction from L_1 to L_3. Hence $L_1 \leq_P L_3$.

The Class **NP**-Complete (**NPC**)

We have finally reached our goal of introducing class **NPC**.

Definition

The class **NPC** of **NP**-complete problems consists of all decision problems L such that

1. $L \in \mathbf{NP}$;
2. for every $L' \in \mathbf{NP}$, $L' \leq_P L$.

Intuitively, **NPC** consists of all the hardest problems in **NP**.

NP-Completeness and Its Properties

Let L be any problem in **NPC**.

Theorem

1. *If there is a polynomial-time algorithm for L, then there is a polynomial-time algorithm for every $L' \in$ **NP**.*
2. *If there is no polynomial-time algorithm for L, then there is no polynomial-time algorithm for any $L' \in$ **NPC**.*

Proof.

1. By definition of **NPC**, for every $L' \in$ **NP**, $L' \leq_P L$. Since $L \in$ **P**, by the theorem on Slide 6, $L' \in$ **P**.
2. By the previous conclusion. $\square$

NP-Completeness and Its Properties

According to the above theorem, either

1. all **NP**-Complete problems are polynomial time solvable, or
2. all **NP**-Complete problems are not polynomial time solvable.

This is the major reason we are interested in NP-Completeness.

The Classes **P**, **NP**, and **NPC**

Recall

$\mathbf{P} \subseteq \mathbf{NP}$.

Question 1

Is $\mathbf{NPC} \subseteq \mathbf{NP}$?

Yes, by definition!

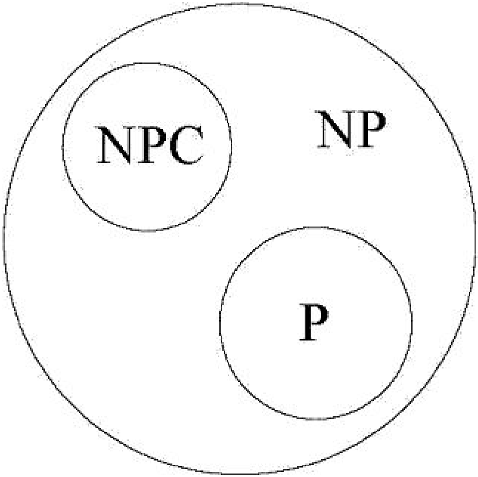
Question 2

Is $\mathbf{P} = \mathbf{NP}$?

Open problem! Probably very hard

It is generally believed that $\mathbf{P} \neq \mathbf{NP}$.

The Classes **P**, **NP**, and **NPC**



The Class **NP**-Complete (**NPC**)

From the definition of **NP**-complete, it appears impossible to prove one problem $L \in$ **NPC**!

- By definition, it requires us to show every $L^{'} \in$ **NP**, $L' \leq_P L$.
- But there are infinitely many problem in **NP**, so how can we argue there exists a reduction from every $L^{'}$ to L?
- To prove the first **NP**-complete problem, we have to use the definition of **NP**, and the simplicity of the TM helps again.

Once we have proved the first **NP**-complete problem, by to the transitivity property of the relation $\leq_p$, we have an easier way to show that a problem $L \in$ **NPC**:

(a) $L \in$ **NP**;
(b) for some $L' \in$ **NPC**, $L' \leq_P L$.

Proof.

Let L'' be any problem in **NP**. Since L' is **NP**-complete, $L'' \leq_p L'$. Since $L' \leq_p L$, by transitivity, $L'' \leq_p L$. $\square$

Cook's Theorem (Cook-Levin Theorem)

Theorem (Cook's Theorem)

$SAT \in \mathbf{NPC}$.

Proof.

See p. 310–312. $\square$

3-SAT $\in$ **NPC**

Theorem

3-SAT $\in$ **NPC**.

Proof.

Cook's Theorem actually proves that SAT $\in$ **NPC** when the formula is in conjunctive normal form. We will reduce this problem to 3-SAT. Given a Boolean formula in conjunctive normal form, with $k > 3$ literals, say $C = (\lambda_1 \vee \lambda_2 \vee \cdots \vee \lambda_k)$, we introduce new variables $y_1, \ldots, y_{k-1}$ and replace C with

$$(\lambda_1 \vee \lambda_2 \vee y_1) \wedge (\overline{y_1} \vee \lambda_3 \vee y_2) \wedge (\overline{y_2} \vee \lambda_4 \vee y_3) \wedge \cdots$$
$$\wedge(\overline{y_{k-4}} \vee \lambda_{k-2} \vee y_{k-3}) \wedge (\overline{y_{k-3}} \vee \lambda_{k-1} \vee \lambda_k)$$

The transformed formula is satisfiable iff the original formula is satisfiable (why?). □

Proving that problems are **NPC**

From SAT and 3-SAT, we will show the following problems are **NP**-complete.

1. DCLIQUE:
   - by showing 3-$\mathrm{SAT} \le_{\mathrm{P}} \mathrm{DCLIQUE}$
   - The reduction used is not natural.
2. Decision Vertex Cover (DVC):
   - by showing $\mathrm{DCLIQUE} \le_{\mathrm{P}} \mathrm{DVC}$
   - The reduction used is very natural.
3. Decision Independent Set (DIS):
   - by showing $\mathrm{DCLIQUE} \le_{\mathrm{P}} \mathrm{DIS}$
   - The reduction used is very natural.

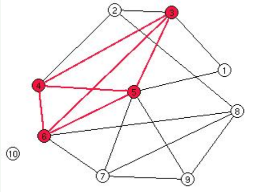
Problem: CLIQUE

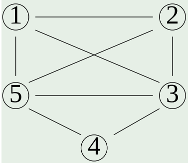
Definition (Clique)

A clique in an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices such that each pair $u, v \in V'$ is connected by an edge $(u, v) \in E$. In other words, a clique is a complete subgraph of G

Example

- a vertex is a clique of size 1, an edge a clique of size 2.





Find a clique with 4 vertices

CLIQUE

Find a clique of maximum size in a graph.

NPC Problem: DCLIQUE

The Decision Clique Problem DCLIQUE

Given an undirected graph G and an integer k, determine whether G has a clique with k vertices.

Theorem

$\mathrm{DCLIQUE} \in \mathbf{NPC}$.

Proof

We need to show two things.

(a) That $\mathrm{DCLIQUE} \in \mathbf{NP}$ and

(b) That there is some $L \in \mathbf{NPC}$ such that

$$L \leq_P \mathrm{DCLIQUE}.$$

Proof that $\mathrm{DCLIQUE} \in$ **NPC**

Claim (a)

$\mathrm{DCLIQUE} \in$ **NP**

Proof.

Proving (a) is easy.

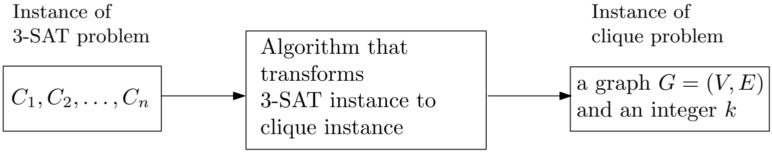
- A certificate will be a set of vertices $V' \subseteq V$, $|V'| = k$ that is a possible clique.
- To check that V' is a clique all that is needed is to check that all edges (u, v) with $u \neq v$, $u, v \in V'$, are in E.
- This can be done in time $O(|V|^2)$ if the edges are kept in an adjacency matrix (and even if they are kept in an adjacency list – how?).

□

Claim (b)

*There is some $L \in$ **NPC** such that $L \leq_P \mathrm{DCLIQUE}$.*

To prove (b) we will show that $\text{3-SAT} \leq_{\mathrm{P}} \mathrm{DCLIQUE}$.



- This will be the hard part.
- We will do this by building a 'gadget' that allows a reduction from the 3-SAT problem (on logical formulas) to the DCLIQUE problem (on graphs, and integers).

Proof that $\mathrm{DCLIQUE} \in$ **NPC** (cont)

Recall that the input to 3-SAT is a logical formula ϕ of the form

$$\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$$

where each clause C_i is a triple of the form

$$C_i = y_{i,1} \vee y_{i,2} \vee y_{i,3}$$

where each literal $y_{i,j}$ is a variable or the negation of a variable.

Example

$$C_1 = (x_1 \vee \neg x_2 \vee \neg x_3),\ C_2 = (\neg x_1 \vee x_2 \vee x_3),\ C_3 = (x_1 \vee x_2 \vee x_3)$$

We will define a polynomial transformation f from 3-SAT to $\mathrm{DCLIQUE}$

$$f : \phi \mapsto (G, k)$$

that builds a graph G and integer k such that ϕ is a Yes-input to 3-SAT if and only if (G, k) is a Yes-input to $\mathrm{DCLIQUE}$.

Proof that DCLIQUE $\in$ **NPC** (cont)

- Suppose that ϕ is a 3-SAT formula with n clauses, i.e., $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$.
- We start by setting $k = n$.
- We now construct the graph $G = (V, E)$.

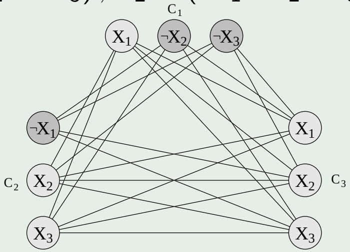
1. For each clause $C_i = x_{i,1} \vee x_{i,2} \vee x_{i,3}$ we create 3 vertices, v_1^i, v_2^i, v_3^i, in V so G has $3n$ vertices. We will label these vertices with the corresponding variable or variable negation that they represent. (Note that many vertices might share the same label) Example
2. We create an edge between vertices v_j^i and $v_{j'}^{i'}$ if and only if the following two conditions hold:
   (a) v_j^i and $v_{j'}^{i'}$ are in different triples, i.e., $i \neq i'$, and
   (b) v_j^i is not the negation of $v_{j'}^{i'}$. Example

Note that the transformation maps all 3-SAT inputs to some DCLIQUE inputs, i.e., it does not require that all DCLIQUE inputs have pre-images from 3-SAT inputs.

Proof that $\mathrm{DCLIQUE} \in \textbf{NPC}$ (cont)

Example

$$\phi = C_1 \wedge C_2 \wedge C_3$$
$$C_1 = (x_1 \vee \neg x_2 \vee \neg x_3),\ C_2 = (\neg x_1 \vee x_2 \vee x_3),\ C_3 = (x_1 \vee x_2 \vee x_3)$$



◂ Return

- Observe that the assignment X_1 =false, X_2 =false, X_3 =true satisfies ϕ (a yes-input for 3-SAT).
- This corresponds to the clique of size 3 comprising the $\neg x_2$ node in C_1, the x_3 node in C_2, and the x_3 node in C_3 (a yes-input for DCLIQUE).

Proof that DCLIQUE $\in$ **NPC** (cont)

Correctness

We claim that a 3-CNF formula ϕ with k clauses is satisfiable if and only if $f(\phi) = (G, k)$ has a clique of size k.

$\Rightarrow$: Suppose ϕ is satisfiable. Consider the satisfying truth assignment.

- Each of the k clauses has at least one true literal.
- Select one such true literal from each clause.
- Observe that these true literals must be logically consistent with each other (i.e., for any i, x_i and $\neg x_i$ will not both appear).
- Recall that in our construction of G we connect a pair of vertices if they are in different clauses and are logically consistent.
- Thus, for every pair of these literals, there must be an edge in G connecting the corresponding vertices.
- Thus these k vertices must form a clique.

Proof that $\mathrm{DCLIQUE} \in$ **NPC** (cont)

$\Leftarrow$: Suppose G has a clique of size k.

- Observe that there is no edge between vertices in the same clause.
- Hence, each clause 'contributes' exactly one vertex to the clique.
- Moreover, since the construction of G connects only logically consistent vertices by an edge, every vertex in the clique must be logically consistent.
- Hence we can assign all the vertices in the clique to be true, and this truth assignment makes ϕ satisfiable.

Proof that $\mathrm{DCLIQUE} \in$ **NPC** (cont)

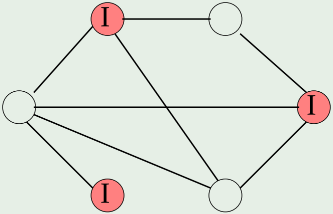
- Note that the graph G has $3k$ vertices and at most $3k(3k-1)/2$ edges and can be built in $O(k^2)$ time
- So f is a polynomial-time reduction.
- We have therefore just proven that $\text{3-SAT} \leq_{\mathrm{P}} \mathrm{DCLIQUE}$.
- Since we already know that $\text{3-SAT} \in$ **NPC** and have seen that $\mathrm{DCLIQUE} \in$ **NP**, we have just proven that $\mathrm{DCLIQUE} \in$ **NPC**.

Problem: Independent Set

Definition

An independent set is a subset I of vertices in an undirected graph G such that no pair of vertices in I is joined by an edge of G.

Example



Optimization Problem

Given an undirected graph G, find an independent set of maximum size.

NPC Problem: Decision Independent Set (DIS)

Decision Problem (DIS)

Given an undirected graph G and an integer k, does G contain an independent set consisting of k vertices?

Theorem

$\mathrm{DIS} \in \mathbf{NPC}$.

Proof.

It is very easy to see that DIS $\in$ **NP**.

- A certificate is a set of vertices $S \subseteq V$ with $|S| = k$ and, in $O(|S|^2) = O(|V|^2)$ time we can check whether or not S is an independent set.

In the next slide we will see that $\mathrm{DCLIQUE} \le_{\mathrm{P}} \mathrm{DIS}$, completing the proof. □

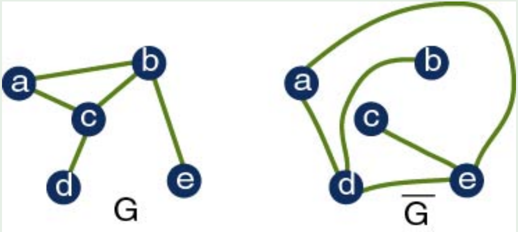
$\mathrm{DIS} \in$ **NPC**: Complement of a Graph

Definition

The complement of a graph $G = (V, E)$ is defined by $\overline{G} = (V, \overline{E})$, where

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

Example



$\mathrm{DIS} \in$ **NPC**

We can define a transformation from DCLIQUE to DIS:

$$f : (G = (V, E), k) \mapsto (\overline{G} = (V, \bar{E}), k)$$

Claim

We claim (G, k) is a yes-input to DCLIQUE if and only if $(\bar{G}, k)$ is a yes-input to DIS.

Proof.

$\Rightarrow$: Let $V^{'}$ be a clique of size k of G. Hence in $\bar{G}$, there is no edge between any pair of vertices in $V^{'}$ which means $V^{'}$ is a IS of $\bar{G}$ of size k.
$\Leftarrow$: Let $V^{'}$ be a IS of size k in $\bar{G}$. Hence in G, every pair of vertices in $V^{'}$ will be connected by an edge. Hence $V^{'}$ is a clique of G of size k. $\square$

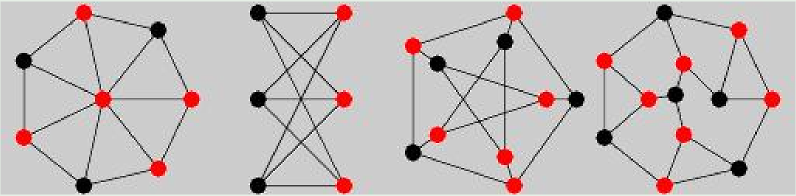
Moreover, f can be calculated in polynomial time. We have just shown that $\mathrm{DCLIQUE} \leq_{\mathrm{P}} \mathrm{DIS}$ and completed the proof that $\mathrm{DIS} \in$ **NPC**.

Problem: VC

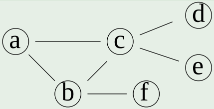
Definition (Vertex Cover)

A vertex cover of G is a set of vertices such that every edge in G is incident to at least one of these vertices.

Example



Example



Find a vertex cover of G of size two

NPC Problem: DVC

The Vertex Cover Problem (VC)

Given a graph G, find a vertex cover of G of minimum size.

The Decision Vertex Cover Problem (DVC)

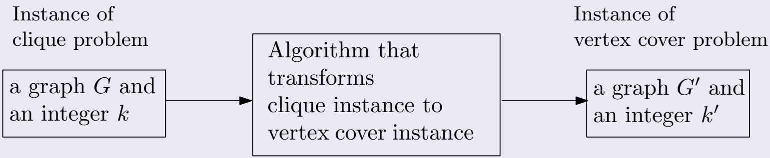
Given a graph G and integer k, determine whether G has a vertex cover with k vertices.

Theorem

DVC $\in$ **NPC**.

Proof.

- Previously we showed that DVC $\in$ **NP**.
- We now show that DCLIQUE $\leq_P$ DVC.



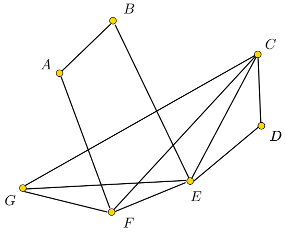
- The conclusion then follows from the fact that DCLIQUE $\in$ **NPC**.

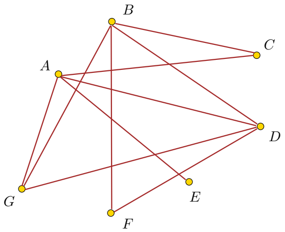
$\square$

Proof.

Let $k' = |V| - k$. We define a transformation f from DCLIQUE to DVC:

$$f : (G = (V, E), k) \mapsto (\overline{G} = (V, \bar{E}), k')$$





- f can be computed (that is, $\overline{G}$ and k' can be determined) in time $O(|V|^2)$ time.

Claim

We claim that a graph G has a clique of size k (yes-input of DCLIQUE) if and only if the complement graph $\overline{G}$ has a vertex cover of size $|V| - k$ (a yes-input of DVC).

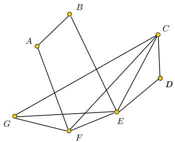
Proof.

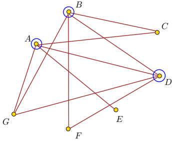
$\Rightarrow$:

- Let $V^{'}$ be a clique of size k in G, then in $\bar{G}$, there is no edge between any two vertices in $V^{'}$.
- Hence $V'' = V \setminus V'$ is a vertex cover of $\bar{G}$;
- note that this is a vertex cover of size $k' = |V| - k$.

$\Leftarrow$:

Let V' be a vertex cover of $\bar{G}$ of size $|V| - k$.

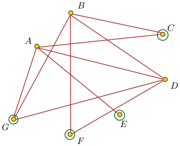




Vertex cover: A, B and D

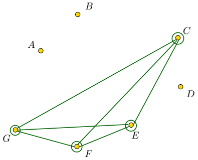
Let $V'' = V \setminus V'$.

- Note that $|V''| = k$.



Vertices in G' not in the veretex cover (no edge between them)

By the definition of vertex cover, for any $u, v \in V^{''}$, then $(u, v) \notin \bar{E}$. Thus $(u, v) \in E$. Therefore $V^{''}$ is a clique of size k in G.



Clique of size 4 in G

NP-Hard Problems

Definition

A problem L is **NP-hard** if problem in **NPC** can be polynomially reduced to it (but L does not need to be in **NP**).

In general, the optimization versions of **NP**-Complete problems are **NP**-Hard.

Example

VC: Given an undirected graph G, find a minimum-size vertex cover.
DVC: Given an undirected graph G and k, is there a vertex cover of size k?
If we can solve the optimization problem VC, we can easily solve the decision problem DVC.

- Simply run VC on graph G and find a minimum vertex cover S.
- Now, given (G, k), solve $DVC(G, k)$ by checking whether $k \geq |S|$. If $k \geq |S|$, answer Yes, if not, answer No.

Epilogue: How to Deal with Hard Problems

- Heuristics: All the hardness results (undecidability, NP-hardness) hold for any algorithm that solves the problem in general (worst-case analysis). There are many efficient algorithms solving these problems for typical cases.
    - They run fast on typical inputs and find the optimal solutions (they may be slow on some contrived inputs).
    - They run fast on all inputs and typically find near-optimal solutions (they may return bad solutions on some contrived inputs).
- Approximation algorithms: All the hardness results show that finding the optimal solutions is difficult, but there are efficient algorithms for finding solutions that are at most c times worse than the optimal ones.
- Average-case analysis: By assuming the input follows some distribution, it is possible to design algorithms whose running time is good on average.