



Lecture 18

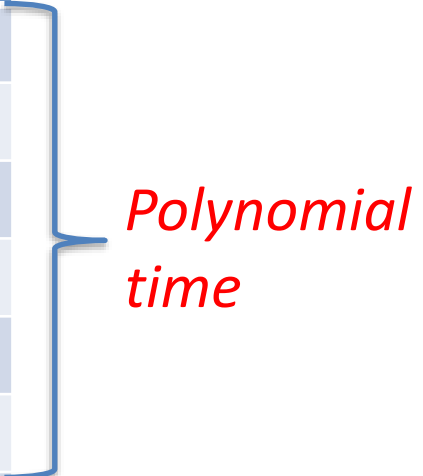
P, NP and reductions

CS 161 Design and Analysis of Algorithms

Ioannis Panageas

Different time complexities

Different algorithms can have different **time complexities**.

Some common complexity classes	Notation (input size = n)	 <i>Polynomial time</i>
Constant	$O(1)$	
Logarithmic	$O(\log n)$	
Linear	$O(n)$	
Log-linear	$O(n \log n)$	
Quadratic	$O(n^2)$	
Cubic	$O(n^3)$	
Exponential	$O(e^n)$	
Factorial	$O(n!)$	
Doubly-exponential	$O(e^{e^n})$	

We say an algorithm runs in **polynomial time** if its time complexity is **$O(n^c)$** for some constant c .

P and NP

Given a decision problem A (output yes/no), there could be many possible solutions, with possibly different time complexities.

The class P: We say **can be solved in polynomial time** or belongs in P if there exist at least one algorithm that solves the problem and runs in **polynomial** time.

P and NP

Given a decision problem A (output yes/no), there could be many possible solutions, with possibly different time complexities.

The class P: We say **can be solved in polynomial time** or belongs in P if there exist at least one algorithm that solves the problem and runs in **polynomial** time.

The class NP: It stands for **Non-deterministic polynomial time**. In high level, if the answer is “yes”, it can be verified in polynomial time.

Example: “Given a number x , is it composite?”

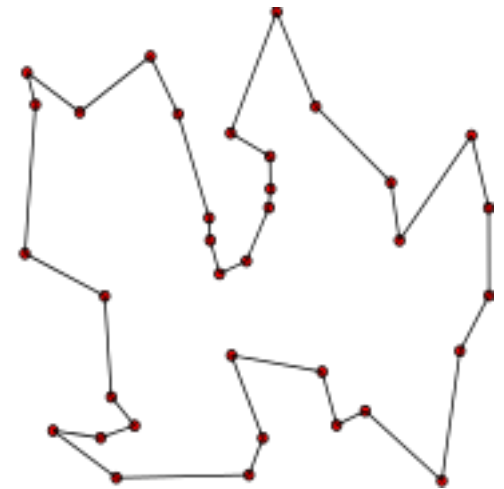
Example: “Given a graph $G(V, E)$, does it contain a cycle?”.

Optimization Problems

Problem: The traveling salesman problem

Given a list of cities and the distances between each pair of cities, **what is a shortest possible route** that visits **each city** exactly **once** and returns to the origin city?

- If there are n cities, then the “best” known solution uses dynamic programming and has time complexity $O(n^2 2^n)$.
- “best” solution \approx brute-force search + dynamic programming



This problem is suspected to be **not solvable in polynomial time**.

- We still do not know...

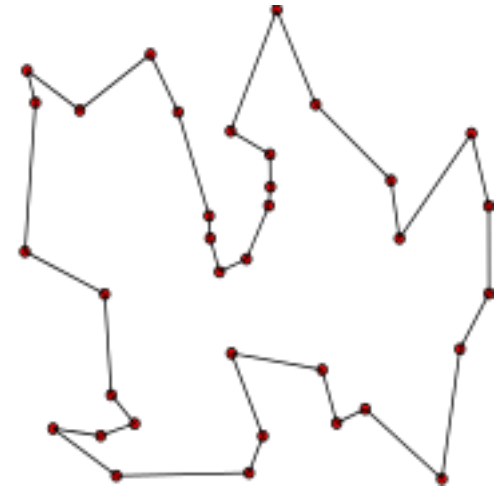
Other example: 0/1 Knapsack problem.

Convert optimization to decision problems

Problem: The traveling salesman problem

Given a list of cities and the distances between each pair of cities, is there a route of length at most k that visits each city exactly once and returns to the origin city?

- If there are n cities, then the “best” known solution uses dynamic programming and has time complexity $O(n^2 2^n)$.
- “best” solution \approx brute-force search + dynamic programming



This problem belongs to NP. Why?

Unsolvable problems?

Question: Are there unsolvable computational problems?

There are examples of unsolvable problems.

- The most famous one is called the **halting problem**.

The Halting Problem:

Given a computer program Π and some input I , determine whether Π will terminate when executed with input I .

- This is a decision (**yes/no**) problem. The answer to the halting problem is either yes or no.
 - **Yes**, if Π terminates.
 - **No**, if Π runs forever (e.g. enters an infinite loop).
- If I is not a valid input for Π , then Π executed with input I will terminate with an error message.

How do we show a problem is not in P?

Question: How can we prove that a problem is not in P?

- **Short answer:** For many problems, we don't know how!

Current Status: We do not know of any **general method** that works on all problems, that can **prove** that a problem is **not** in ***P***.

- In fact, we do not even know of any general method that can prove that a problem is not solvable in linear time.
- We can characterize their computational difficulty using **reductions**.

The idea of reductions

There are so many **different** computational **problems** that we may want to solve.

- Do we have to solve **every** one of these problems **from scratch**?

Key Idea of reductions

Given a **Problem *A*** that we **want to solve**, and suppose there is another **Problem *B*** that we **already know how** to solve.

- If we can reformulate Problem ***A*** to “**look like**” Problem ***B***, so that by solving Problem ***B***, we are able to solve Problem ***A***.

The idea of reductions

There are so many **different** computational **problems** that we may want to solve.

- Do we have to solve **every** one of these problems **from scratch**?

Key Idea of reductions

Given a **Problem A** that we **want to solve**, and suppose there is another **Problem B** that we **already know how** to solve.

- If we can reformulate Problem A to “**look like**” Problem B , so that by solving Problem B , we are able to solve Problem A .

Example: A = **maximum matching** and B = **Maxflow**.

- Then we say that we have **reduced** Problem A to Problem B .
- Problem B is at least as hard as Problem A .

NP-complete problems

NP-complete: A problem A is NP-complete if

1. **Belongs** in NP
2. Any other problem in NP reduces in poly-time to A . In other words, A is **NP-hard**.

What does this mean? A is the **“hardest”** problem in class NP.

NP-complete problems

NP-complete: A problem A is NP-complete if

1. **Belongs** in NP
2. Any other problem in NP reduces in poly-time to A . In other words, A is **NP-hard**.

What does this mean? A is the “**hardest**” problem in class NP.

In 1971, the **first** NP-complete problem appears.

Theorem: The **3-SAT** problem is NP-complete.
(Cook–Levin’s Thm, 1971)

3-SAT is NP-complete

Problem: 3-SAT

Given a Boolean expression E , such that E is a **conjunction** of **clauses**, where each clause is a **disjunction** of exactly 3 literals, is E **satisfiable**?

3-SAT is NP-complete

Problem: 3-SAT

Given a Boolean expression E , such that E is a **conjunction** of **clauses**, where each clause is a **disjunction** of exactly 3 **literals**, is E **satisfiable**?

A **literal** is a Boolean expression consisting of just a single Boolean variable, or the negation of a Boolean variable.

- **Example:** “ \bar{x}_1 ” and “ x_2 ” are literals.

A **clause** is a Boolean expression of the form “ $\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k$ ”, i.e. a **disjunction** of some literals $\ell_1, \ell_2, \dots, \ell_k$. In 3-SAT $k = 3$.

- **Example:** “ $C_1 \equiv x_1 \vee \bar{x}_2 \vee x_3$ ” is a clause.

3-SAT is NP-complete

Problem: 3-SAT

Given a Boolean expression E , such that E is a **conjunction** of **clauses**, where each clause is a **disjunction** of exactly 3 **literals**, is E **satisfiable**?

A **literal** is a Boolean expression consisting of just a single Boolean variable, or the negation of a Boolean variable.

- **Example:** “ \bar{x}_1 ” and “ x_2 ” are literals.

A **clause** is a Boolean expression of the form “ $\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k$ ”, i.e. a **disjunction** of some literals $\ell_1, \ell_2, \dots, \ell_k$. In 3-SAT $k = 3$.

- **Example:** “ $C_1 \equiv x_1 \vee \bar{x}_2 \vee x_3$ ” is a clause.

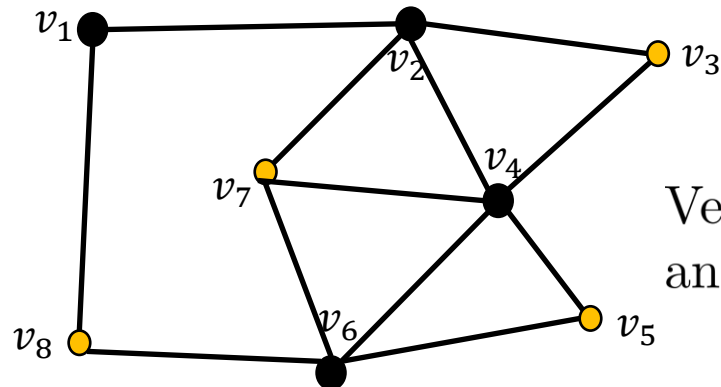
A Boolean expression is a conjunction of clauses.

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

Reductions in NP

Example: INDEPENDENT SET (IS) Problem

Given a simple undirected graph $G(V, E)$ and k , is there an **independent set** in G of size $\geq k$? Independent set is called a set $I \subset V$ of vertices such that pairwise the vertices in I **do not share** an edge.



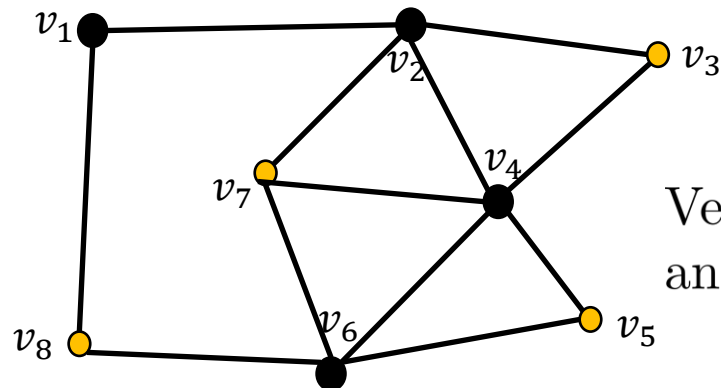
Graph G .

Vertices v_3, v_5, v_7, v_8 form an **independent set**.

Reductions in NP

Example: INDEPENDENT SET (IS) Problem

Given a simple undirected graph $G(V, E)$ and k , is there an **independent set** in G of size $\geq k$? Independent set is called a set $I \subset V$ of vertices such that pairwise the vertices in I **do not share** an edge.



Graph G .

Vertices v_3, v_5, v_7, v_8 form an **independent set**.

Claim: INDEPENDENT SET is **NP-complete**.

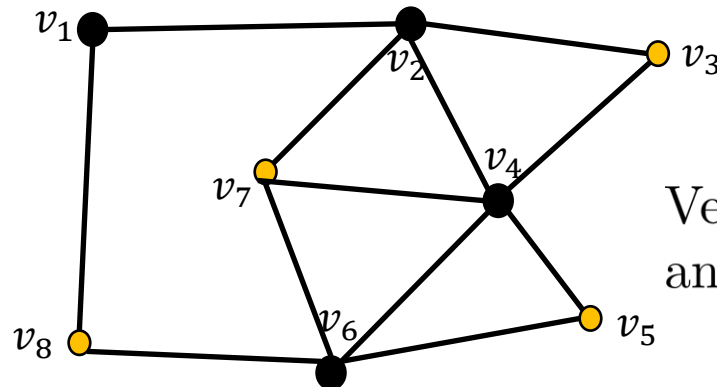
Proof: (1) INDEPENDENT SET **belongs** to **NP** (why?).

(2) Reduce 3-SAT to INDEPENDENT SET. Since 3-SAT is NP-hard, INDEPENDENT SET is NP-hard.

Reductions in NP

Example: INDEPENDENT SET (IS) Problem

Given a simple undirected graph $G(V, E)$ and k , is there an **independent set** in G of size $\geq k$? Independent set is called a set $I \subset V$ of vertices such that pairwise the vertices in I **do not share** an edge.



Graph G .

Vertices v_3, v_5, v_7, v_8 form an **independent set**.

(1), (2) imply IND. SET is NP-complete!

Claim: INDEPENDENT SET is **NP-complete**.

Proof: (1) INDEPENDENT SET **belongs** to **NP** (why?).

(2) Reduce 3-SAT to INDEPENDENT SET. Since 3-SAT is NP-hard, INDEPENDENT SET is NP-hard.

3-SAT reduction to IS

3-SAT instance: Can you assign True, False to the variables of the formula below so that the expression is True?

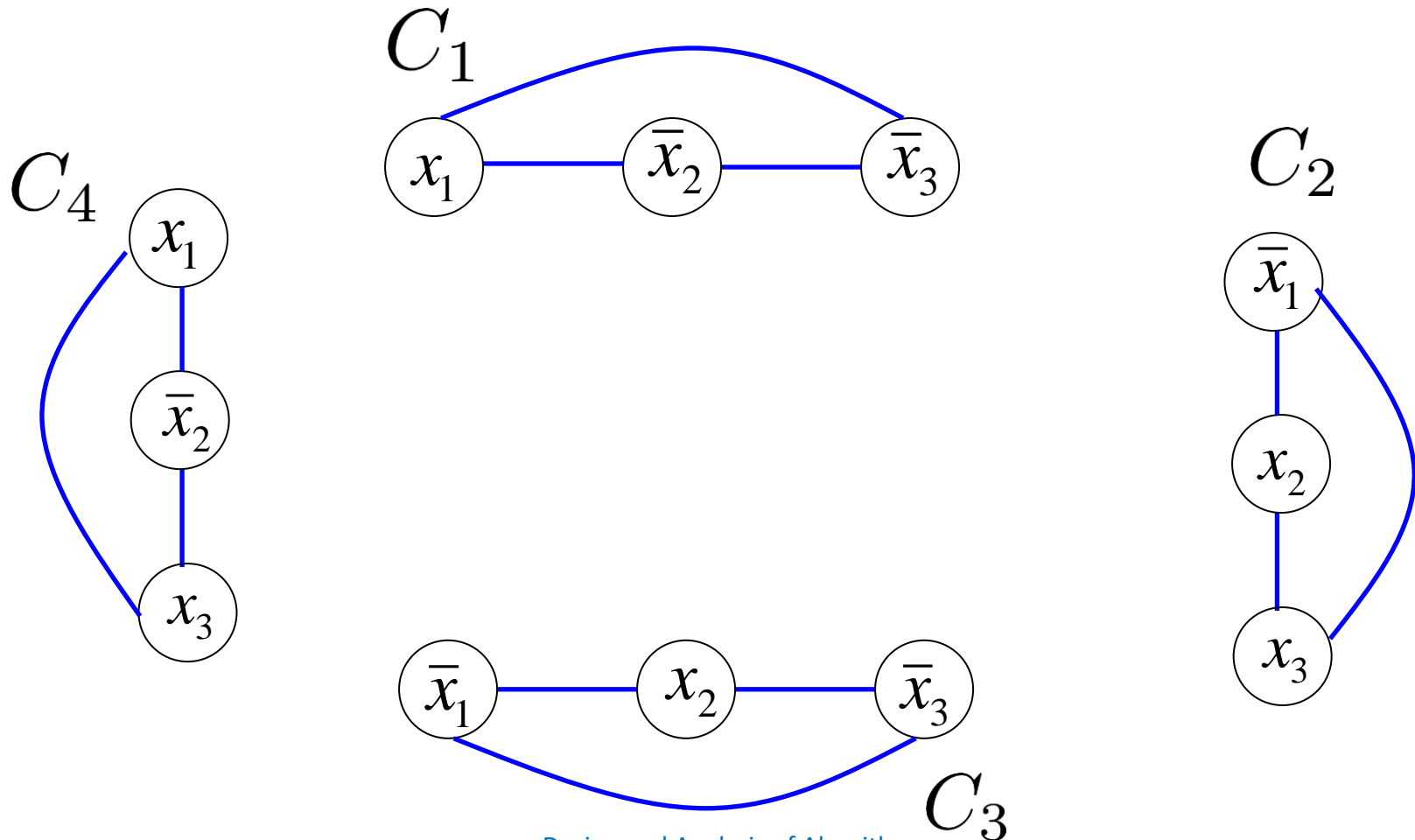
$$E = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$$

Let's **reduce** the above to an **IS** instance. We need a **graph**!

3-SAT reduction to IS

3-SAT instance: Can you assign True, False to the variables of the formula below so that the expression is True?

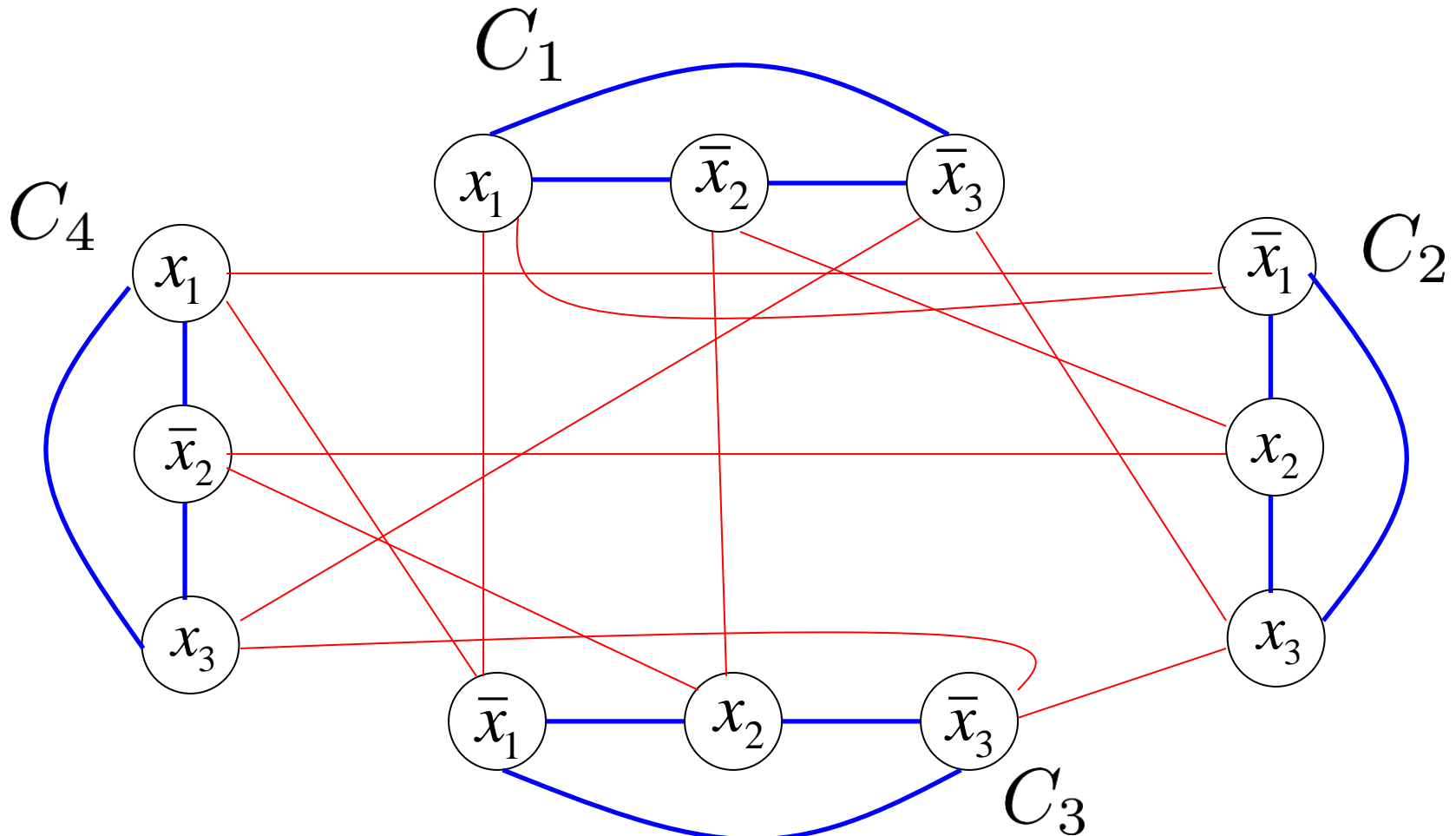
$$E = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$$



3-SAT reduction to IS

3-SAT instance: Can you assign True, False to the variables of the formula below so that the expression is True?

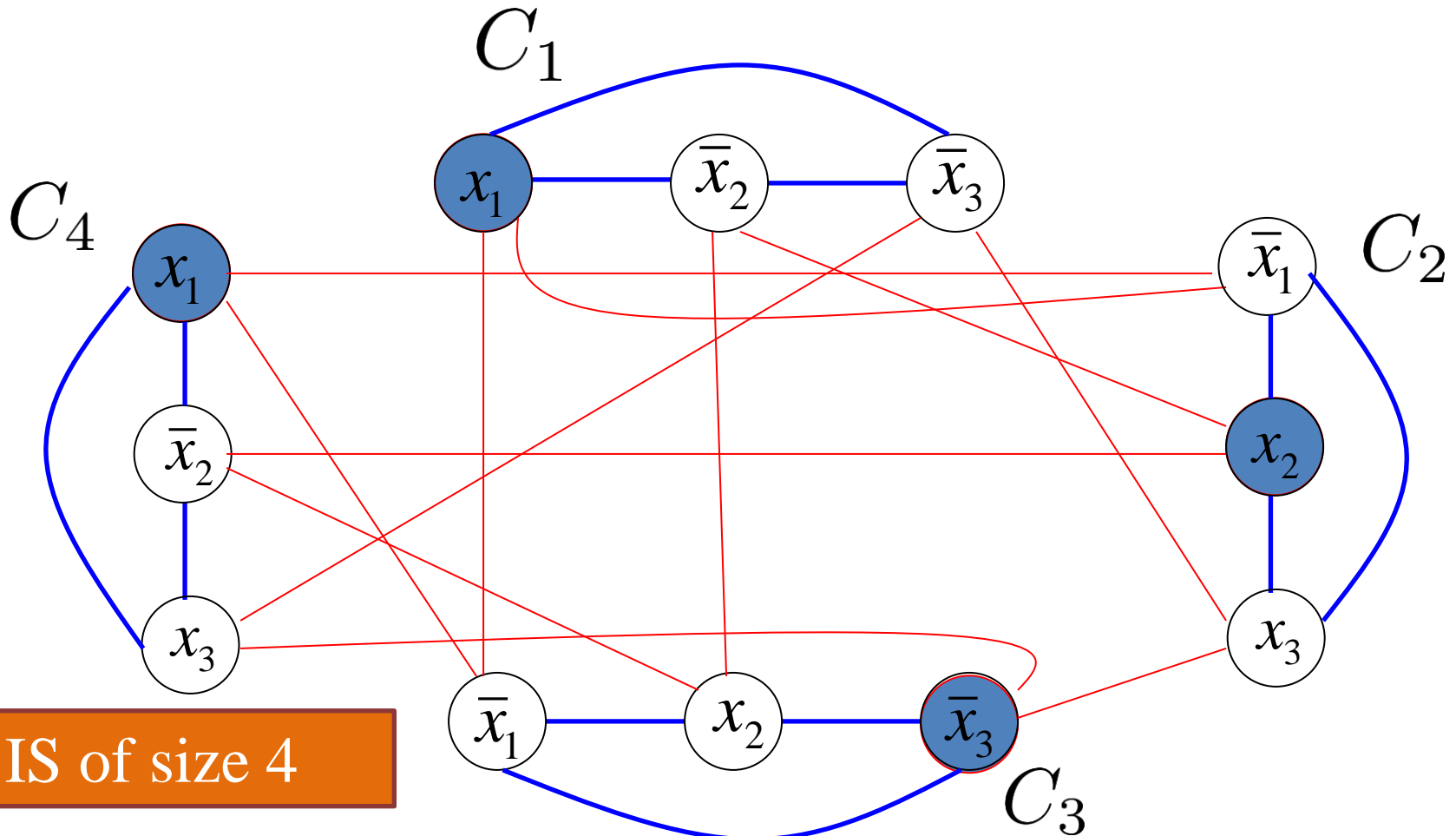
$$E = \overset{C_1}{(x_1 \vee \bar{x}_2 \vee \bar{x}_3)} \wedge \overset{C_2}{(\bar{x}_1 \vee x_2 \vee x_3)} \wedge \overset{C_3}{(\bar{x}_1 \vee x_2 \vee \bar{x}_3)} \wedge \overset{C_4}{(x_1 \vee \bar{x}_2 \vee x_3)}$$



3-SAT reduction to IS

3-SAT instance: Can you assign True, False to the variables of the formula below so that the expression is True?

$$E = \overset{C_1}{(x_1 \vee \bar{x}_2 \vee \bar{x}_3)} \wedge \overset{C_2}{(\bar{x}_1 \vee x_2 \vee x_3)} \wedge \overset{C_3}{(\bar{x}_1 \vee x_2 \vee \bar{x}_3)} \wedge \overset{C_4}{(x_1 \vee \bar{x}_2 \vee x_3)}$$



3-SAT reduction to IS

Claim: Expression E with k clauses is satisfiable if and only if the induced graph G has an IS of size k .

Therefore, given a **graph G and a k** , if we can identify in **poly-time** if there exists an **Independent Set of size at least k** , then we can solve in **poly-time 3-SAT**.

3-SAT reduction to IS

Claim: Expression E with k clauses is satisfiable if and only if the induced graph G has an IS of size k .

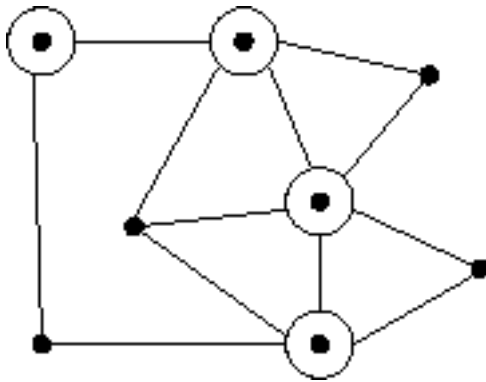
Therefore, given a **graph G and a k** , if we can identify in **poly-time** if there exists an **Independent Set of size at least k** , then we can solve in **poly-time 3-SAT**.

**$3\text{-SAT} \leq_p \text{INDEPENDENT SET} \Rightarrow$
 $\text{INDEPENDENT SET is NP-complete!}$**

Vertex Cover (VC)

Problem: Vertex Cover (VC):

Given a simple undirected graph $G(V, E)$ and k , is there an **vertex cover** in G of size $\geq k$? Vertex cover is called a set $I \subset V$ of vertices such that **all edges are “covered”**?

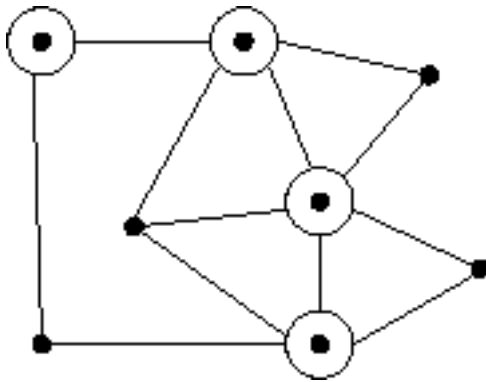


e.g., in this graph, 4 of the 8 vertices are enough to cover **all edges**.

Vertex Cover (VC)

Problem: Vertex Cover (VC):

Given a simple undirected graph $G(V, E)$ and k , is there an **vertex cover** in G of size $\geq k$? Vertex cover is called a set $I \subset V$ of vertices such that **all edges** are “covered”?



e.g., in this graph, 4 of the 8 vertices are enough to cover **all edges**.

Question: VC is NP-Complete? **Answer:** YES

- First, it belongs in NP (why?)
- Reduce 3-SAT to VC (or there is something **simpler**?)

Reduction of IS to Vertex Cover (VC)

- Given a graph $G(V, E)$, with $|V| = n$, we want to know if there exists an Independent Set of size k .

Reduction of IS to Vertex Cover (VC)

- Given a graph $G(V, E)$, with $|V| = n$, we want to know if there exists an Independent Set of **size k** .
- **Lemma:** Given $G(V, E)$, the set of vertices S is an *independent set* **if and only if** $V - S$ **(set of remaining vertices)** is a *vertex cover*.

Reduction of IS to Vertex Cover (VC)

- Given a graph $G(V, E)$, with $|V| = n$, we want to know if there exists an Independent Set of **size k** .
- **Lemma:** Given $G(V, E)$, the set of vertices S is an *independent set* **if and only if** $V - S$ is a *vertex cover*.

Reduction: Does G have a VC of size $n - k$?

Yes: Then it has an IS of size k .

No: Then it does not.

Reduction of IS to Vertex Cover (VC)

- Given a graph $G(V, E)$, with $|V| = n$, suppose there exists an Independent Set of size k .
- **Lemma:** Given $G(V, E)$, the set of vertices S is an independent set if and only if $V - S$ is a vertex cover.

Proof: Let S be an independent set, and $e = (u, v)$ be some edge. Only one of u, v can be in S . Hence, at least one of u, v is in $V - S$. So, $V - S$ is a vertex cover. The other direction is similar.