NP-complete problems

Heejin Park

Division of Computer Science and Engineering

Hanyang University

Contents

Background

- Almost all the algorithms we have studies thus far have been ***polynomial-time algorithms***: on inputs of size n, their worst-case running time is $O(n^k)$for some constant k.

- Do all problems can be solved in polynomial time?

- The answer is ***not yet been discovered***.

Background

- ***"NP-complete"*** problems.

- Not yet been discovered to have *polynomial-time* solvable algorithm

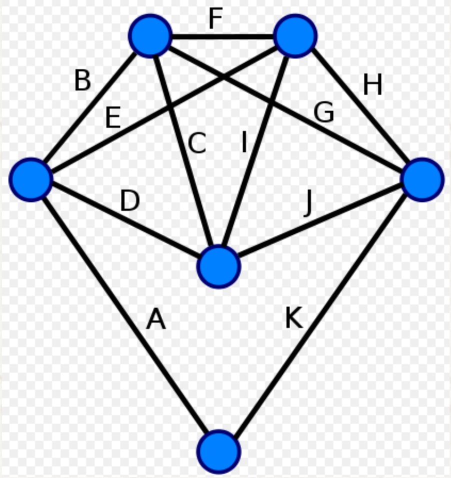
- Several *NP-complete* problems are particularly tantalizing
  - Shortest vs. longest simple paths
  - Euler tour vs. Hamiltonian cycle

Background

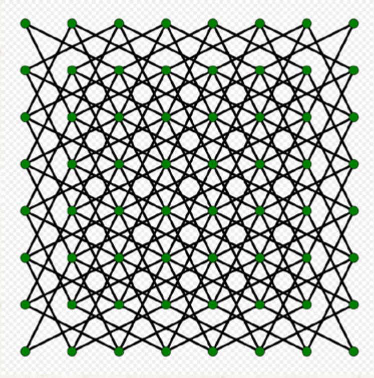
- Shortest vs. longest simple paths:
  - We can find shortest paths from a single source in a directed graph $G = (V, E)$ in $\mathcal{O}(VE)$time.
    Finding a longest simple path between two vertices is difficult, however. Merely determining whether a graph contains a simple path with at least a given number of edges is *NP-complete*.

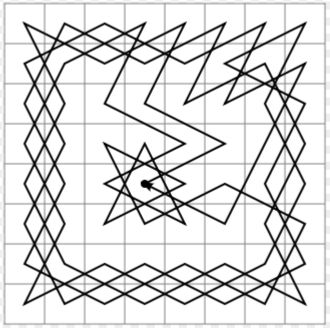
Background

- Euler tour vs. Hamiltonian cycle:



Euler tour





Hamiltonian Cycle

Contents

- Background
- NP-completeness and the classes P and NP
- Overview of showing problems to be NP-C
- The Clique Problems
- The vertex-cover problem

NP-completeness and the classes P and NP

- The class ***P*** consists of those problems that can be solved in time $\mathcal{O}(n^k)$ for some constant k, where n is the size of the input to the problem.

- Most of the problems examined in previous chapters are in P.

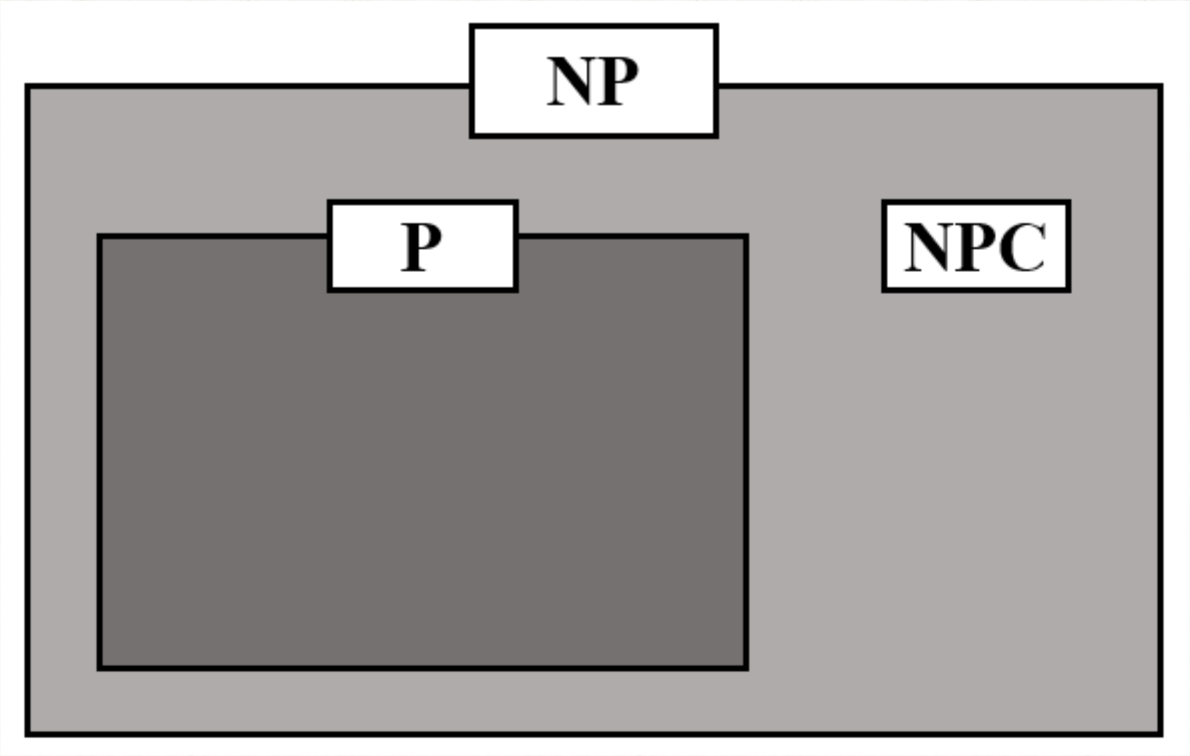
NP-completeness and the classes P and NP

- The class ***NP*** consists of those problems that are "*verifiable*" in polynomial time.

- If we were somehow given a "*certificate*" of a solution, then we could verify that the certificate is correct in time polynomial in the size of the input to the problem.
  - For example, in the Hamiltonian-cycle problem, given a directed graph $G = (V, E)$, a certificate would be a sequence $\langle v_1, v_2, v_3, \ldots, v_{|V|} \rangle$ of $|V|$ vertices. We could easily check in polynomial time that $(v_i, v_{i+1}) \in E$ for $i = 1, 2, 3, \ldots, |V|-1$ and that $(v_{|V|}, v_1) \in E$ as well.

NP-completeness and the classes P and NP

- Informally, a problem is in the class ***NPC*** if it is in *NP* and is as "hard" as any problem in *NP* .



Contents

- Background
- NP-completeness and the classes P and NP
- Overview of showing problems to be NP-C
- The Clique Problems
- The vertex-cover problem

Overview of showing problems to be NP-C

- When we demonstrate that a problem is *NP-complete*, we are making a statement about how ***hard*** it is, rather than about how *easy* it is.

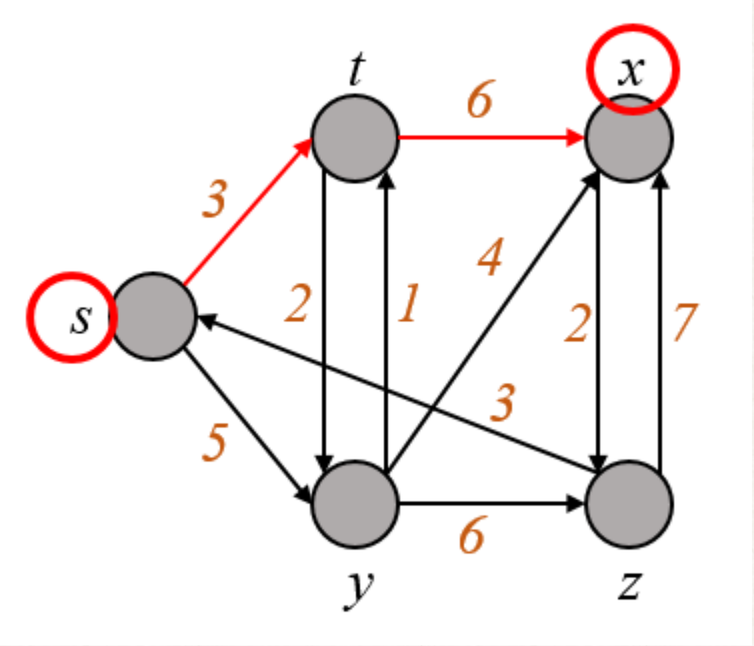
- We are not trying to prove the existence of an efficient algorithm, but instead that no efficient algorithm is likely to exist.

- We rely on three key concepts in showing a problem to be *NP-complete*:
  - Decision problems vs. optimization problems
  - Reductions
  - A first NP-complete problem

Overview of showing problems to be NP-C

- Many problems of interest are optimization problems.

- ***SHORTEST-PATH*** problem



Overview of showing problems to be NP-C

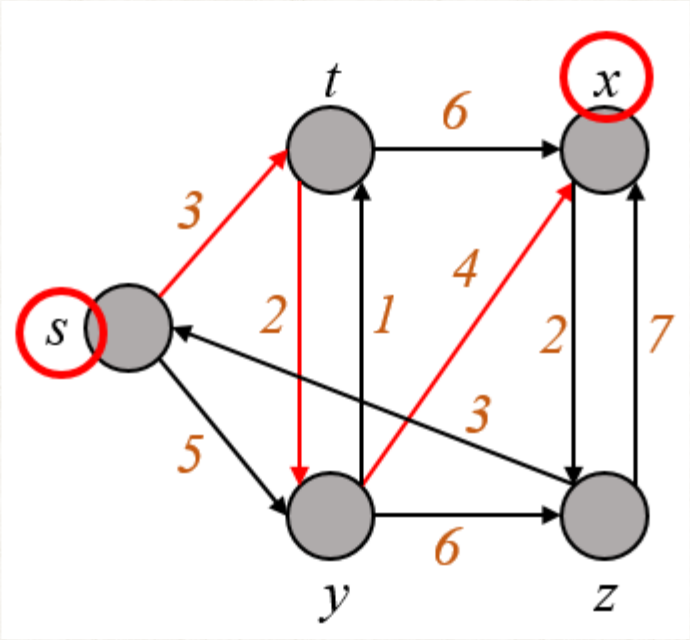
- *NP-completeness* applies directly not to optimization problems, however, but to decision problems, in which the answer is simply “yes” or “no” (or, more formally, “1” or “0”).

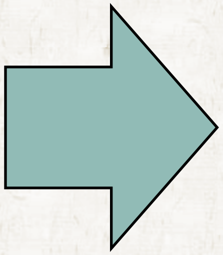
- We usually can ***cast*** a given optimization problem as a related decision problem by imposing a bound on the value to be optimized.

- ***SHORTEST-PATH*** problem

Overview of showing problems to be NP-C

- ***SHORTEST-PATH*** problem

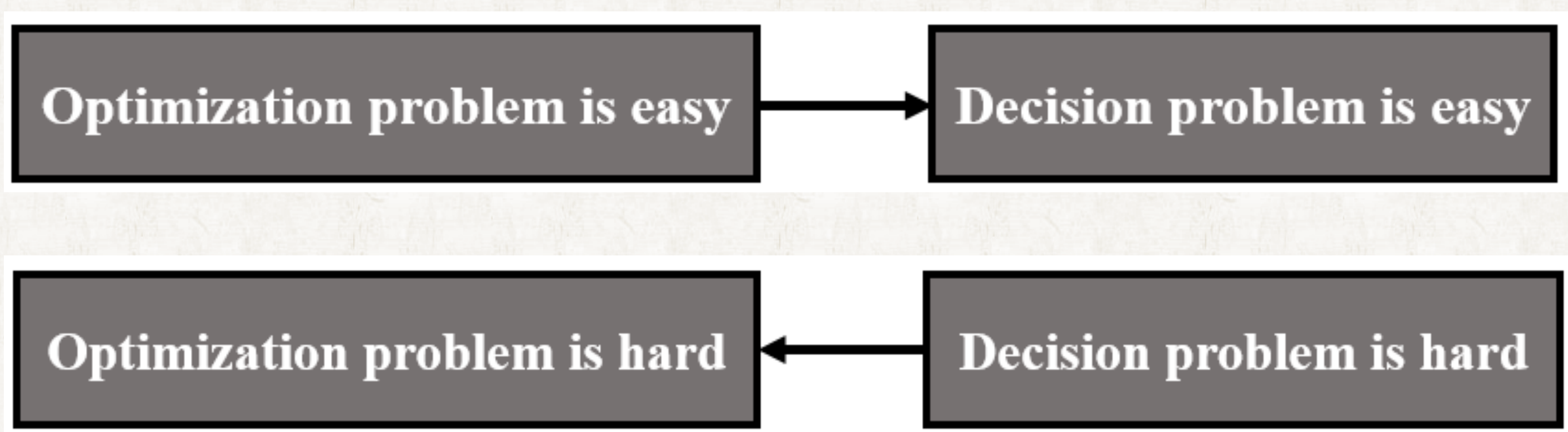




1(Yes)

Overview of showing problems to be NP-C

- We try to show that the optimization problem is "*hard*" with relationship between decision problem and optimization problem.
- The decision problem is "*easier*", or at least "*no harder*".



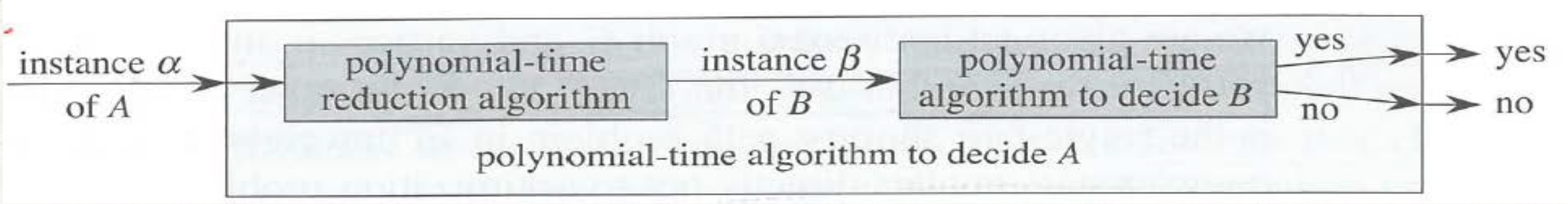
Overview of showing problems to be NP-C

- By "*reducing*" solving problem A to solving problem B, we use the "***easiness***" of B to prove the "***easiness***" of A.

- Let us consider a decision problem A to solve.

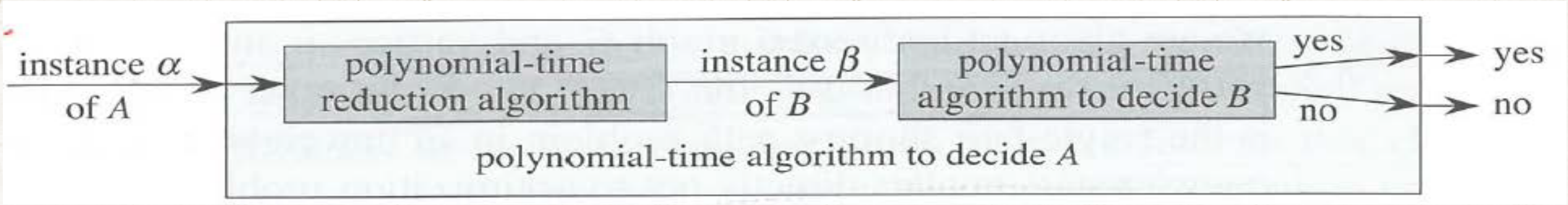
- Now suppose that we already know how to solve a different decision problem B in polynomial time.

- Finally, suppose that we have a procedure that transforms any instance α of A into some instance β of B



Overview of showing problems to be NP-C

- By "*reducing*" solving problem A to solving problem B, we use the "***difficulty***" of A to prove the "***difficulty***" of B.
- Suppose we have a decision problem A for which we already know that no polynomial-time algorithm can exist.
- Suppose further that we have a *polynomial-time* reduction transforming instances of A to instances of B.
- Suppose that B has a *polynomial-time* algorithm.



Overview of showing problems to be NP-C

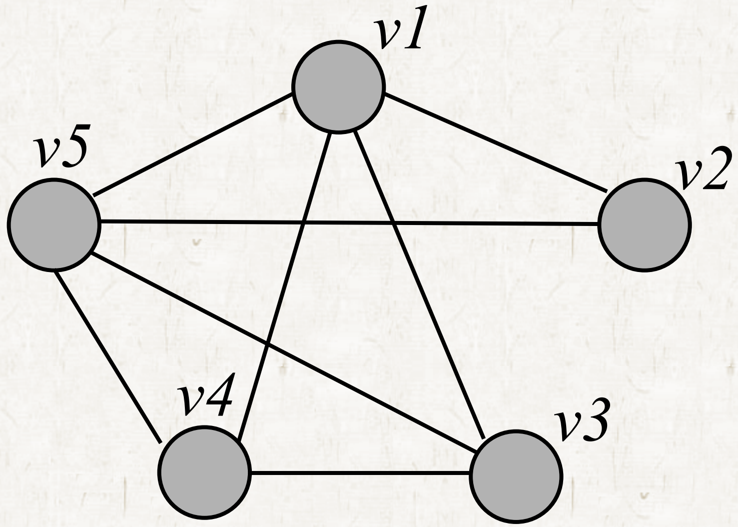
- Because the technique of reduction relies on having a problem already known to be *NP-complete* in order to prove a different problem *NP-complete*, we need a "*first*" *NP-complete* problem.

Contents

- Background
- NP-completeness and the classes P and NP
- Overview of showing problems to be NP-C
- The Clique Problems
- The vertex-cover problem

The Clique Problems

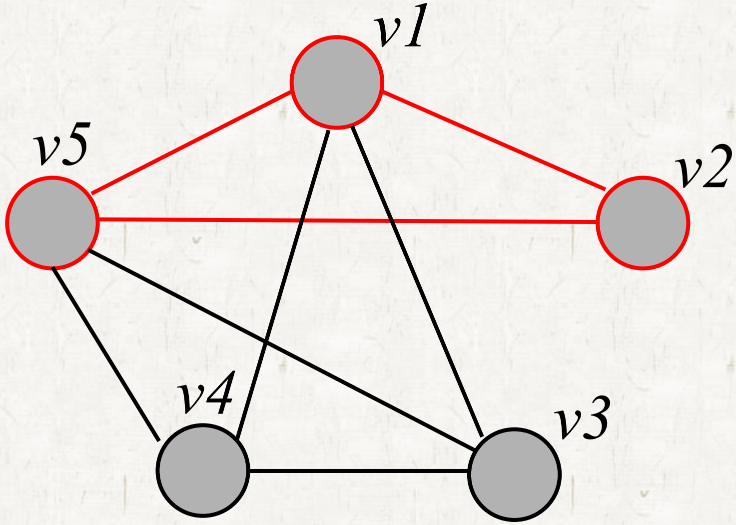
- **Cliques: Particular complete subgraphs in a graph**
- **CLIQUE = {<G,k>: G is a graph containing a clique of size k}**



G = (V, E)

The Clique Problems

- **Cliques: Particular complete subgraphs in a graph**
- **CLIQUE = {<G,k>: G is a graph containing a clique of size k}**



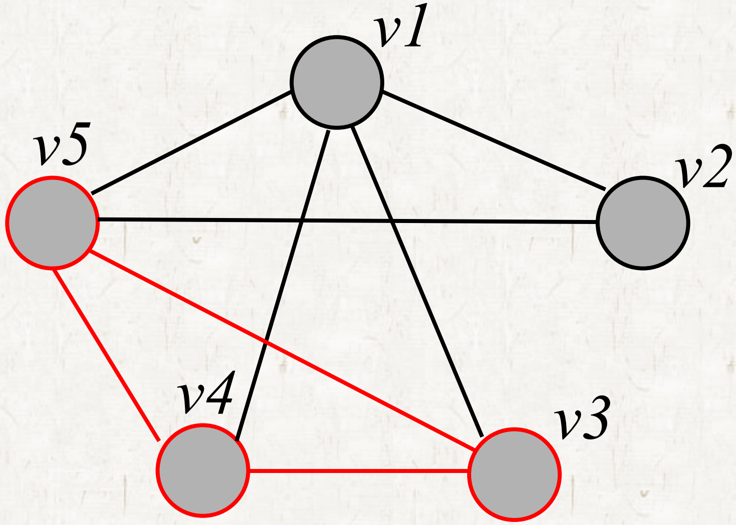
G = (V, E)

<G, 3>

V` = {v1, v2, v5}

The Clique Problems

- **Cliques: Particular complete subgraphs in a graph**
- **CLIQUE = {<G,k>: G is a graph containing a clique of size k}**



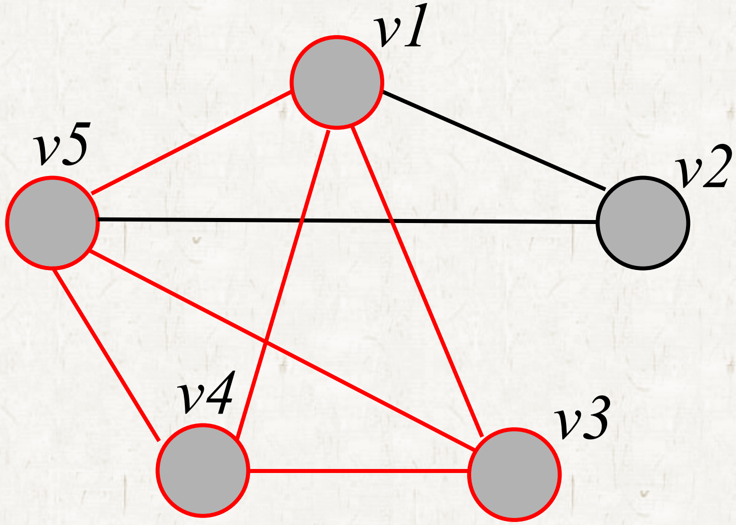
G = (V, E)

<G, 3>

V` = {v3, v4, v5}

The Clique Problems

- **Cliques: Particular complete subgraphs in a graph**
- **CLIQUE = {<G,k>: G is a graph containing a clique of size k}**



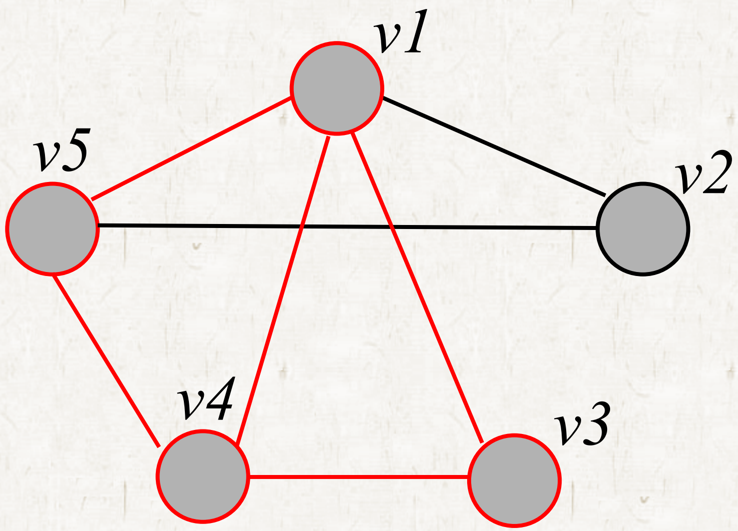
G = (V, E)

<G, 4>

V` = {v1, v3, v4, v5}

The Clique Problems

- **Cliques: Particular complete subgraphs in a graph**
- **CLIQUE = {<G,k>: G is a graph containing a clique of size k}**



G = (V, E)

~~<G, 4>~~

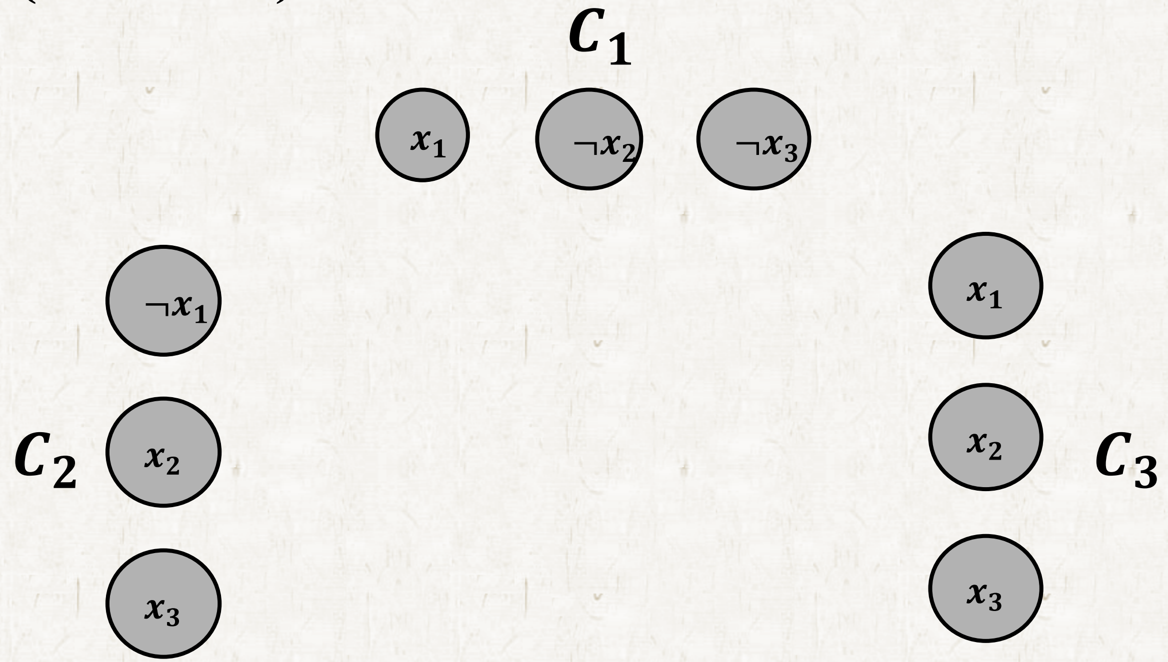
V` = {v1, v3, v4, v5}

The Clique Problems

- **The clique problem is NP-complete**
- **3-CNF-SAT $\leq_p$ CLIQUE**
- **A boolean formula: $\Phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ (k clauses)**
- $C_r = l_1^r \vee l_2^r \vee l_3^r$
- **Put an edge between two vertices v_i^r and v_j^s if**
  - **r != s**
  - **l_i^r is not the negation of l_j^s**

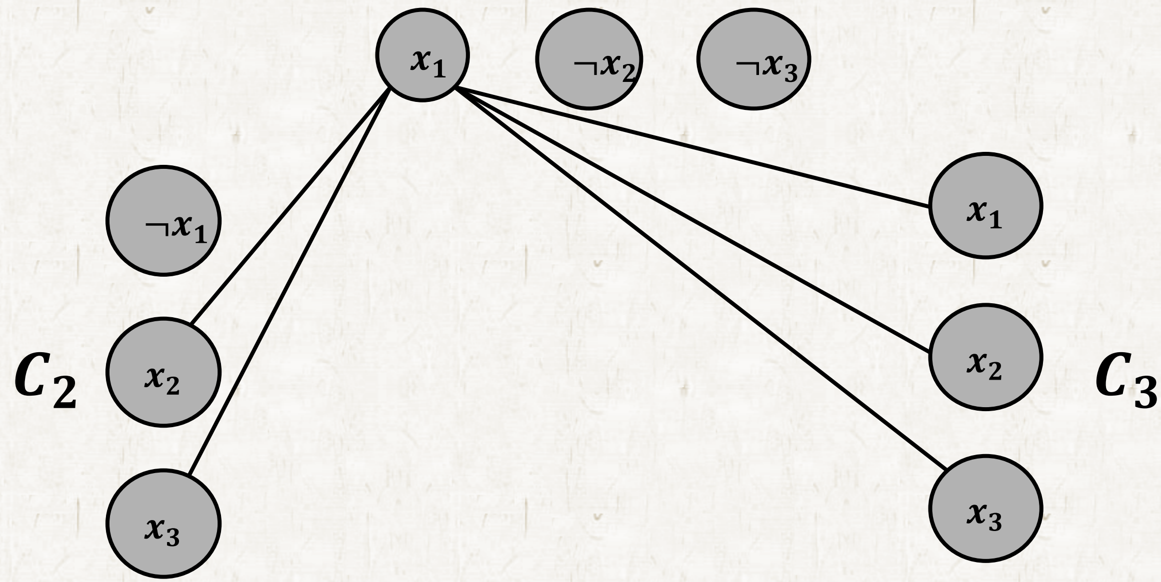
The Clique Problems

- $\boldsymbol{\Phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)}$ **(3 clauses)**



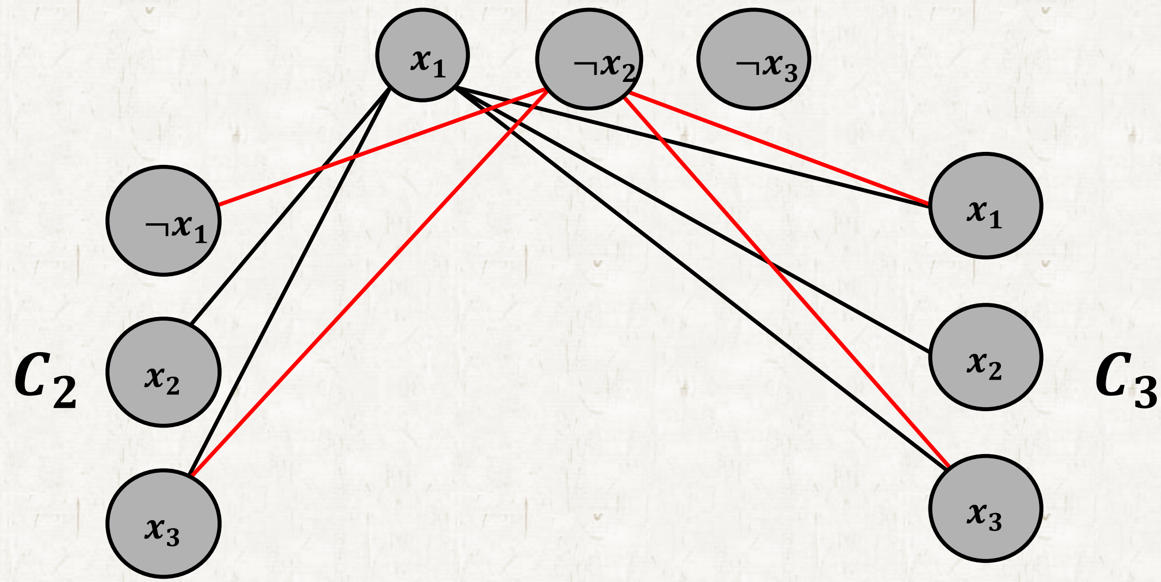
The Clique Problems

- $\boldsymbol{\Phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)}$ **(3 clauses)**



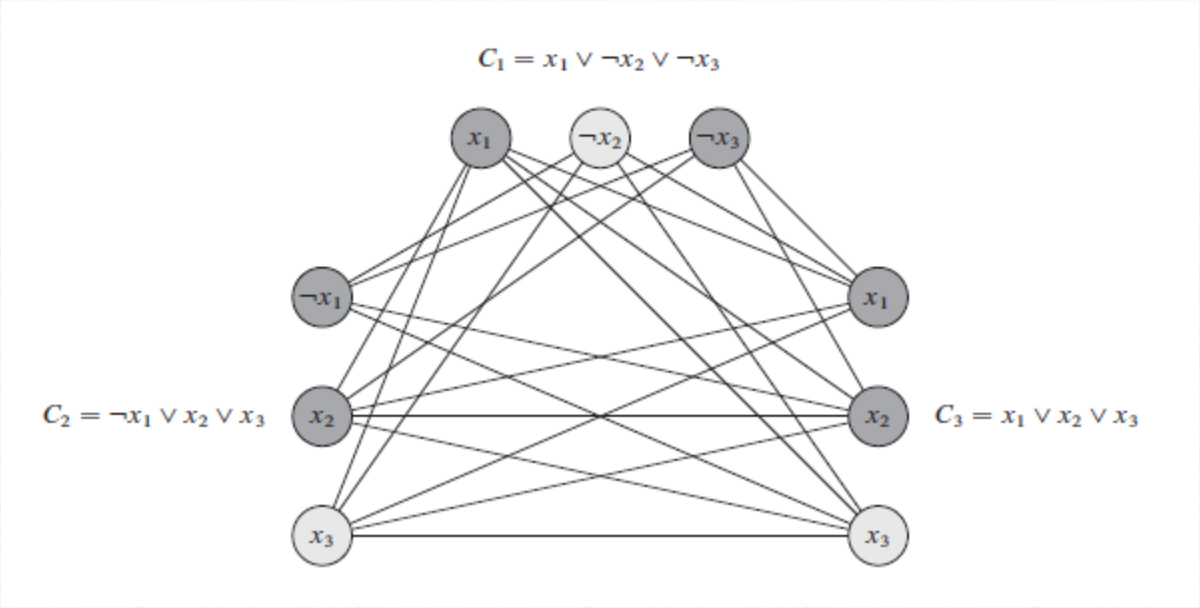
The Clique Problems

- $\boldsymbol{\Phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)}$ **(3 clauses)**



The Clique Problems

- $\boldsymbol{\Phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)}$ **(3 clauses)**



The Clique Problems

- $\boldsymbol{\Phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)}$ **(3 clauses)**
- **Φ has a satisfying assignment**

$$\boldsymbol{\Phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)}$$

The Clique Problems

- $\mathbf{\Phi} = (\boldsymbol{x_1} \vee \neg \boldsymbol{x_2} \vee \neg \boldsymbol{x_3}) \wedge (\neg \boldsymbol{x_1} \vee \boldsymbol{x_2} \vee \boldsymbol{x_3}) \wedge (\boldsymbol{x_1} \vee \boldsymbol{x_2} \vee \boldsymbol{x_3})$ **(3 clauses)**
- **$\mathbf{\Phi}$ has a satisfying assignment**

$$\mathbf{\Phi} = (\boldsymbol{x_1} \vee \neg \boldsymbol{x_2} \vee \neg \boldsymbol{x_3}) \wedge (\neg \boldsymbol{x_1} \vee \boldsymbol{x_2} \vee \boldsymbol{x_3}) \wedge (\boldsymbol{x_1} \vee \boldsymbol{x_2} \vee \boldsymbol{x_3})$$

  - **At least one literal is 1**

The Clique Problems

- $\Phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$ **(3 clauses)**
- **Φ has a satisfying assignment**

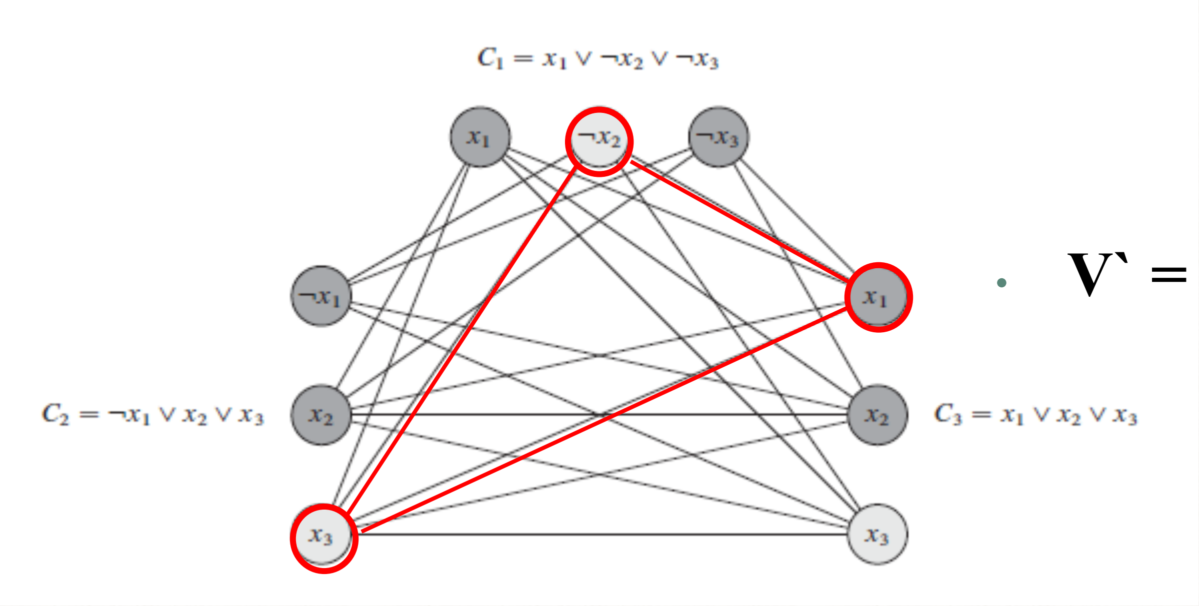
$\Phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$

- $\neg x_2 = 1$
- $x_3 = 1$

- $(\neg x_2, x_3) \in \mathrm{E}$
- $\mathrm{V}` = \{\neg x_2, x_3, x_1\}$
- $\mathrm{V}`` = \{\neg x_2, x_3, x_3\}$

The Clique Problems

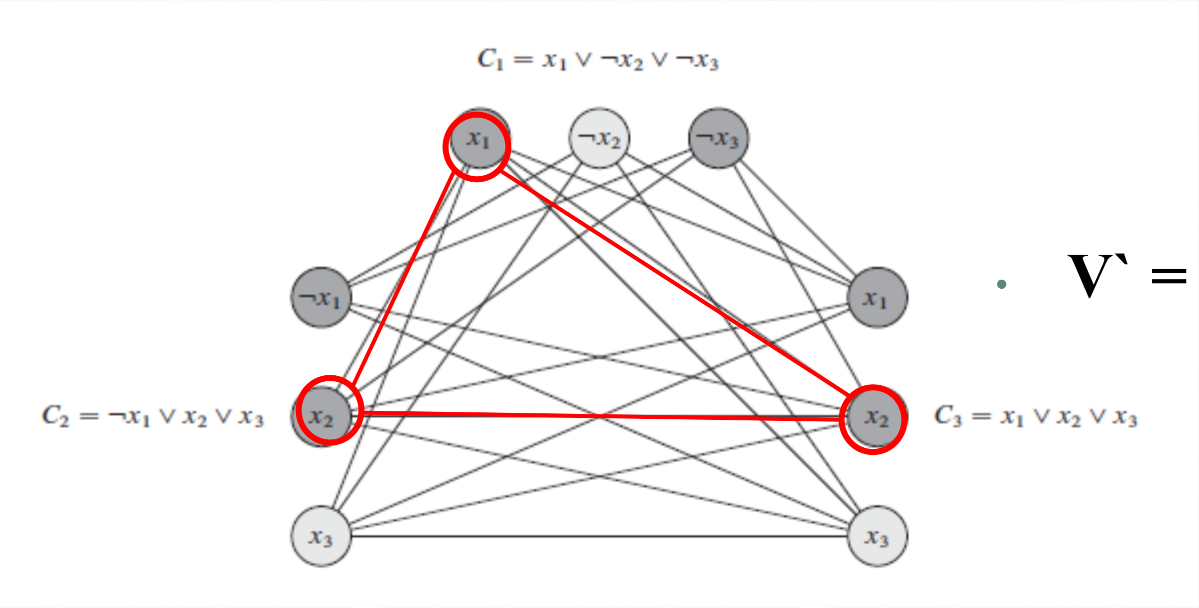
$\boldsymbol{\Phi = (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)}$



- **V`** $= \{\neg x_2, x_3, x_1\}$

The Clique Problems

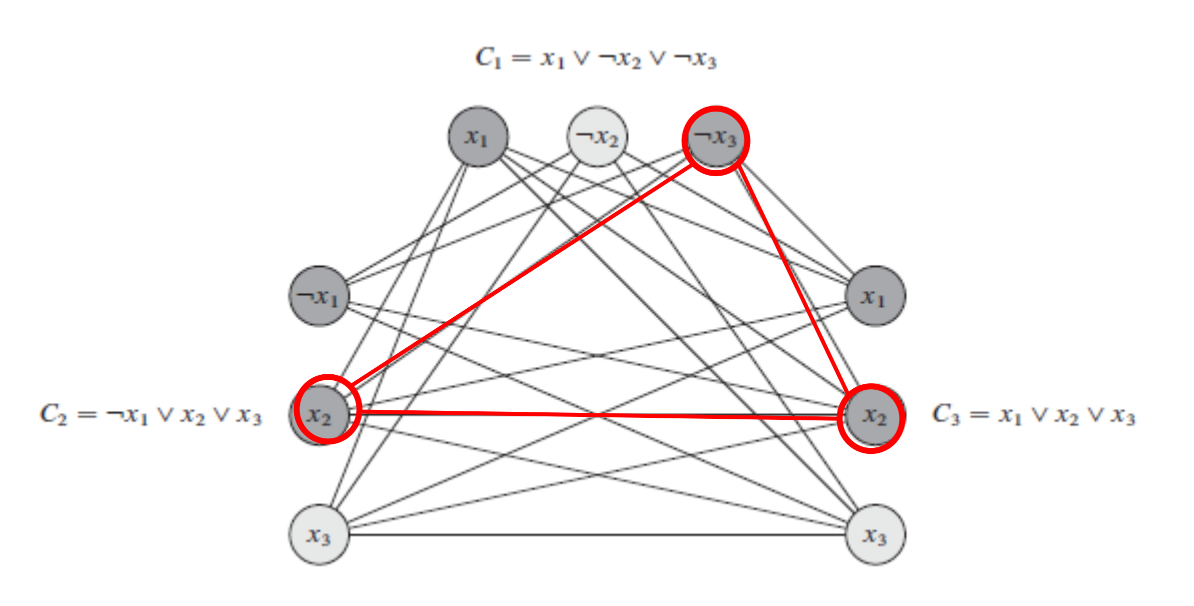
$$\Phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$



- $V` = \{x_1, x_2, x_2\}$

The Clique Problems

$$\Phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$



- $V` = \{\neg x_3, x_2, x_2\}$

The Clique Problems

- $\mathbf{\Phi} = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$ **(3 clauses)**
- **$\mathbf{\Phi}$ has a satisfying assignment**

$$\Phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) = 1$$

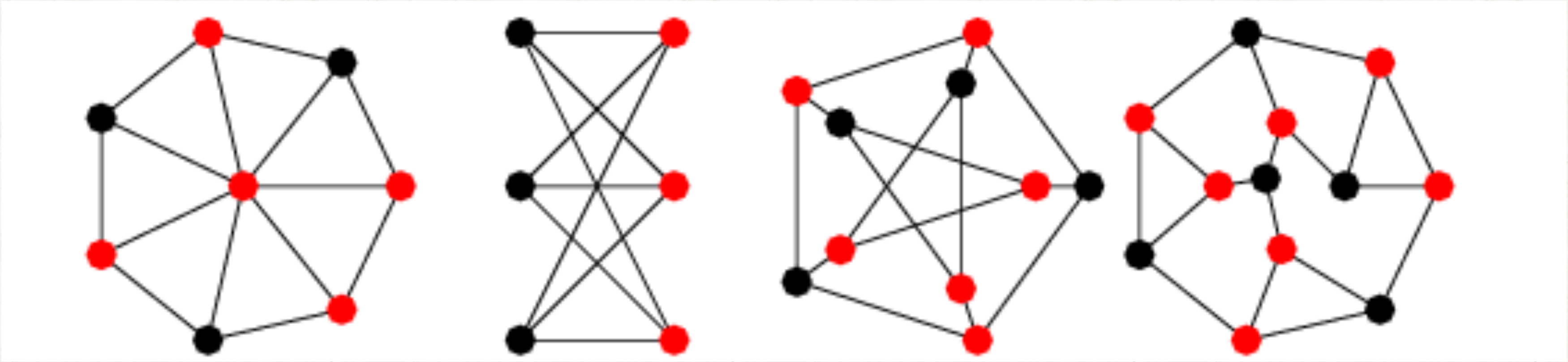
- $\neg x_3 = 1$
- $x_2 = 1$
- $x_2 = 1$

- **V\` =** $\{\neg x_3, x_2, x_2\}$

Contents

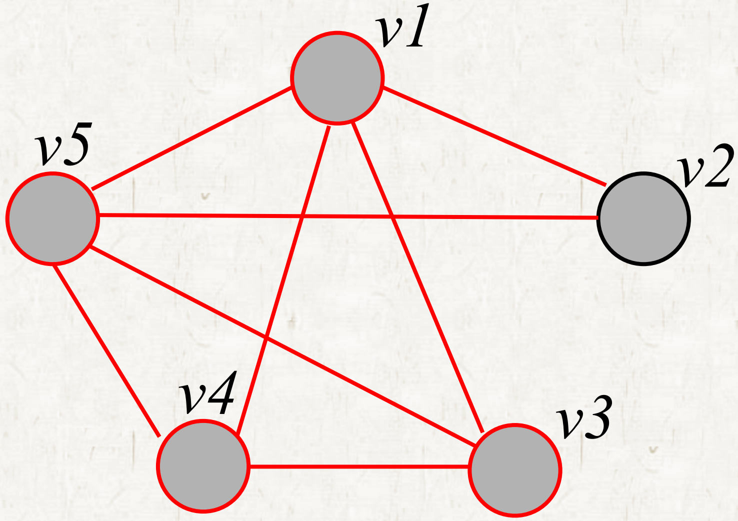
The vertex-cover problem

- **Finding a vertex cover in a given graph**
- **Vertex cover: set of vertices that each edge of the graph is incident to at least one vertex of the set**



The vertex-cover problem

- **Finding a vertex cover in a given graph**
- **Vertex cover: set of vertices that each edge of the graph is incident to at least one vertex of the set**

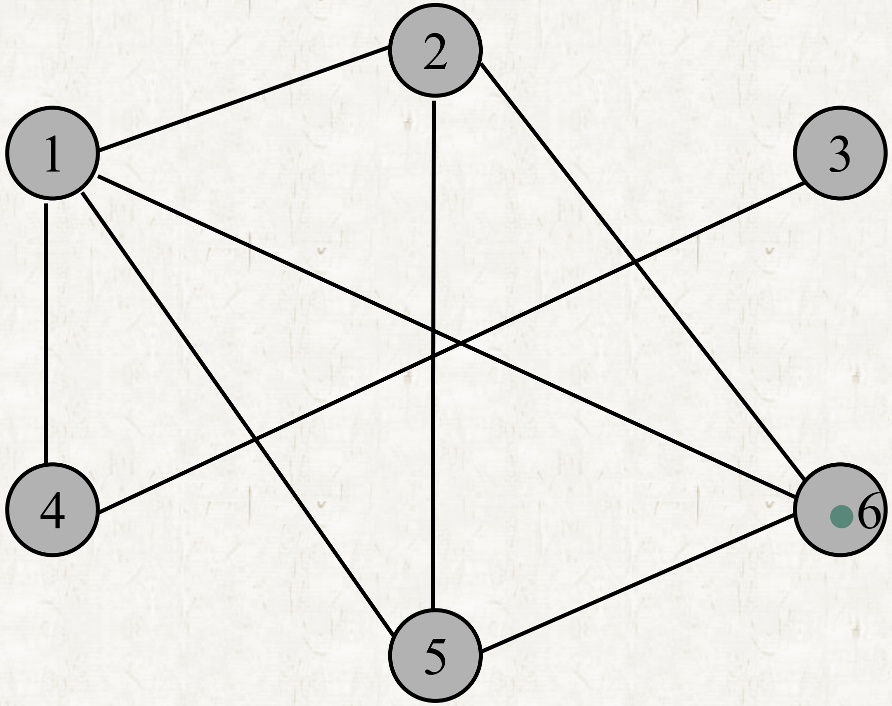


The vertex-cover problem

- **CLIQUE $\leq_p$ VERTEX-COVER**
- **Using complement graph**
- **G = (V, E) -> $\overline{G}$ = (V, $\overline{E}$)**
- **$\overline{E}$ = {(u, v): u, v $\in$ V, u $\neq$ v, (u, v) $\notin$ E)**

The vertex-cover problem

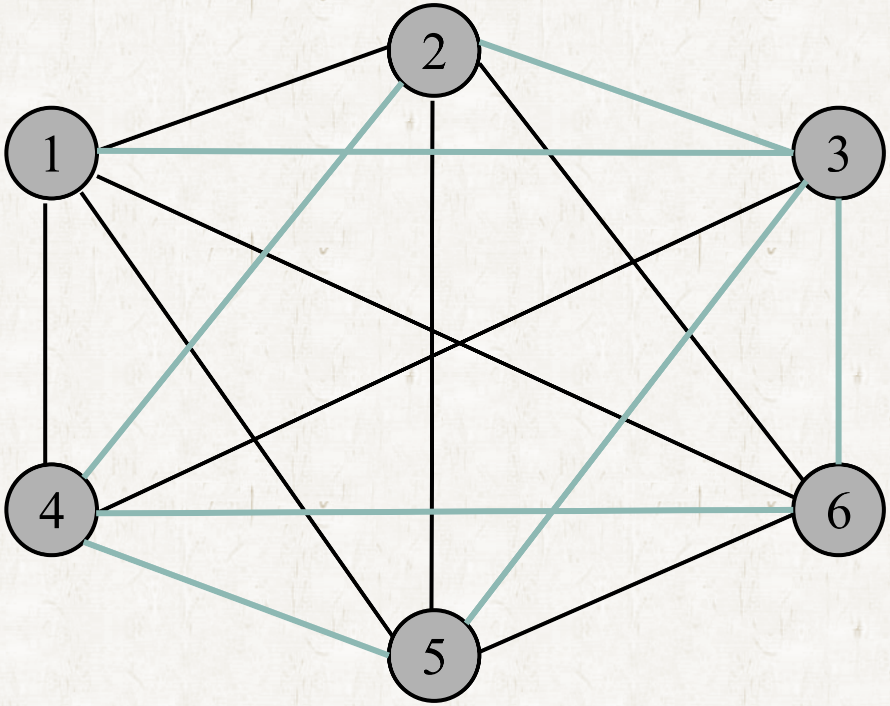
- **Complement graph**
  - **G = (V, E) -> $\overline{G}$ = (V, $\overline{E}$)**
  - **$\overline{E}$ = {(u, v): u, v ∈ V, u ≠ v, (u, v) ∉ E)**



G = (V, E)

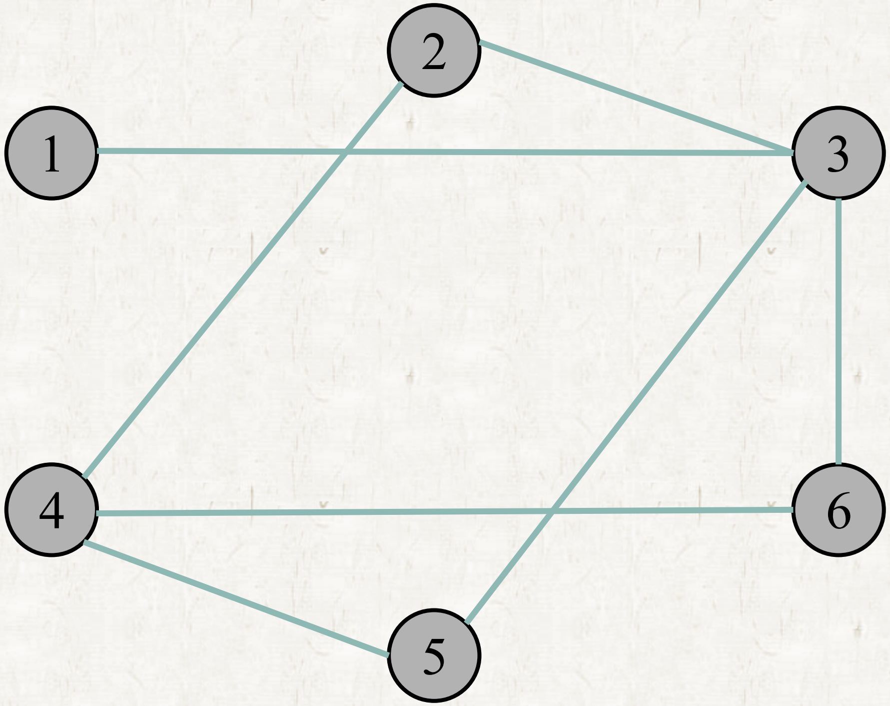
The vertex-cover problem

- **Complement graph**
  - **G = (V, E) -> $\overline{G}$ = (V, $\overline{E}$)**
  - **$\overline{E}$ = {(u, v): u, v $\in$ V, u $\neq$ v, (u, v) $\notin$ E)**



The vertex-cover problem

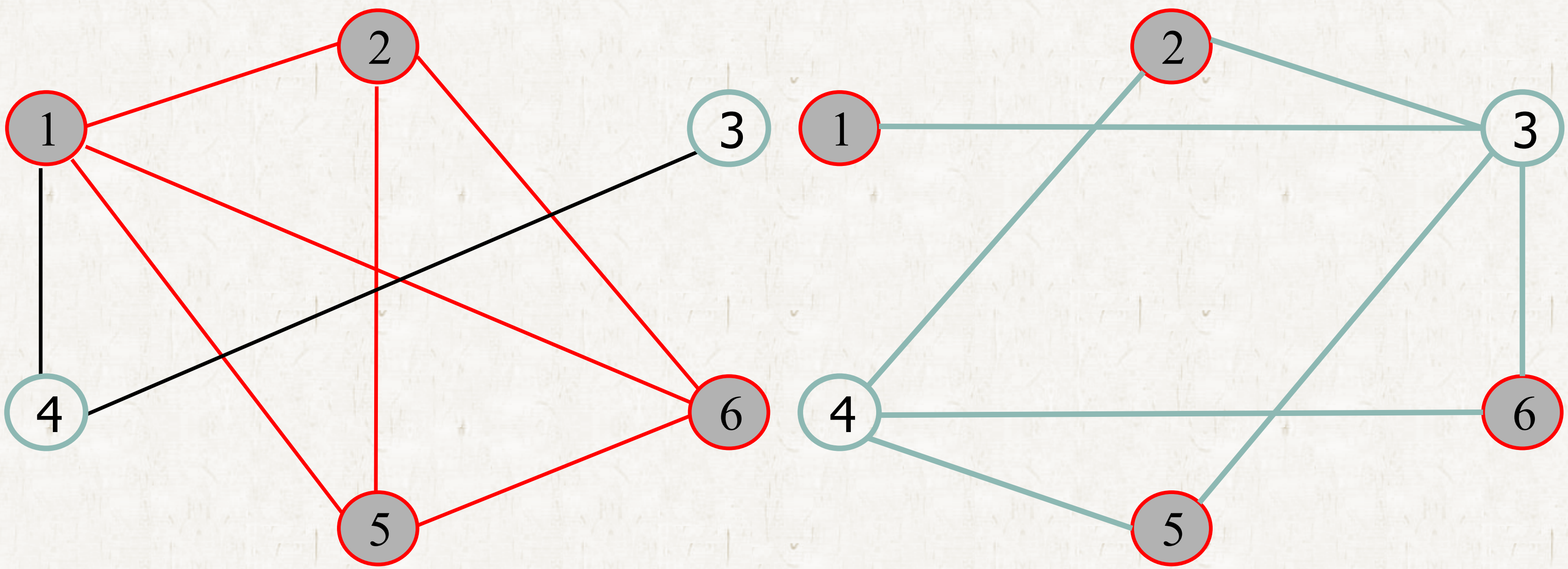
- **Complement graph**
  - **G = (V, E) -> $\overline{G}$ = (V, $\overline{E}$)**
  - **$\overline{E}$ = {(u, v): u, v ∈ V, u ≠ v, (u, v) ∉ E)**



$\overline{G}$ = (V, $\overline{E}$)

The vertex-cover problem

- **CLIQUE <G, k> $\leq_p$ VERTEX-COVER <$\overline{G}$, |V| - k>**



G = (V, E)

$\overline{G}$ = (V, $\overline{E}$)