جامعة الملك سعود
King Saud University

Theory of Computation
CSC 339 – Spring 2021

Chapter-7: part5
NP-Complete Problems

King Saud University

Department of Computer Science

Dr. Azzam Alsudais

Recap

- **Problem A is NP-complete if:**

  1)A is in NP (it can be verified in polynomial time), and

Recap

- **Problem A is NP-complete if:**

  1) **A is in NP (it can be verified in polynomial time), and**
  2) **Every problem in NP reduces to A.**

Recap

- **Problem A is NP-complete if:**

  1) **A is in NP (it can be verified in polynomial time), and**
  2) **Every problem in NP reduces to A.**

- **In this part, we will look at more NP-complete problems and how we use reduction to prove they are NP-complete.**

Introduction

- **Most problems in NP are known either to be in P or to be NP-complete.**

Introduction

- **Most problems in NP are known either to be in P or to be NP-complete.**

- **Again, why do we focus on NP-complete?**

Introduction

- Most problems in NP are known either to be in P or to be NP-complete.

- Again, why do we focus on NP-complete?

  - Assume we encountered a new NP-problem. Instead of spending lots of time seeking a polynomial time algorithm, we might want to try to prove that it is NP-complete.

Introduction

- **Most problems in NP are known either to be in P or to be NP-complete.**

- **Again, why do we focus on NP-complete?**

  - **Assume we encountered a new NP-problem. Instead of spending lots of time seeking a polynomial time algorithm, we might want to try to prove that it is NP-complete.**

  - **Proving that a problem is NP-complete saves us time (instead of wasting too much time seeking a polynomial time solution that doesn’t exist).**

More on Polynomial Time Reduction (p-reduction)

- **When constructing polynomial time reduction from 3SAT to another language, we look for <u>structures (or gadgets)</u> in that language that can simulate the variables and clauses in Boolean formulas.**

More on Polynomial Time Reduction (p-reduction)

- **When constructing polynomial time reduction from 3SAT to another language, we look for <u>structures (or gadgets)</u> in that language that can simulate the variables and clauses in Boolean formulas.**

- **e.g.,**

3SAT	$\xrightarrow{\text{Simulated by}}$ CLIQUE
variables	nodes
clauses	triples
a true variable	node part of clique
each clause must contain at least one true literal	each triple must contain one node in clique

VERTEX-COVER (yet, another NP-complete problem)

- **Description**
  - **If G is undirected graph, a *vertex cover* of G is a subset of nodes where every edge of G touches one of those nodes.**

VERTEX-COVER (yet, another NP-complete problem)

- **Description**

  - **If *G* is undirected graph, a *vertex cover* of *G* is a subset of nodes where every edge of *G* touches one of those nodes.**
  - **The *vertex cover* problem asks whether a graph contains a vertex cover of a specified size:**

VERTEX-COVER (yet, another NP-complete problem)

- **Description**

  - **If *G* is undirected graph, a *vertex cover* of *G* is a subset of nodes where every edge of *G* touches one of those nodes.**
  - **The *vertex cover* problem asks whether a graph contains a vertex cover of a specified size:**

VERTEX-COVER = {⟨*G*, *k*⟩| *G is an undirected graph that has a k-node vertex cover*}**.**

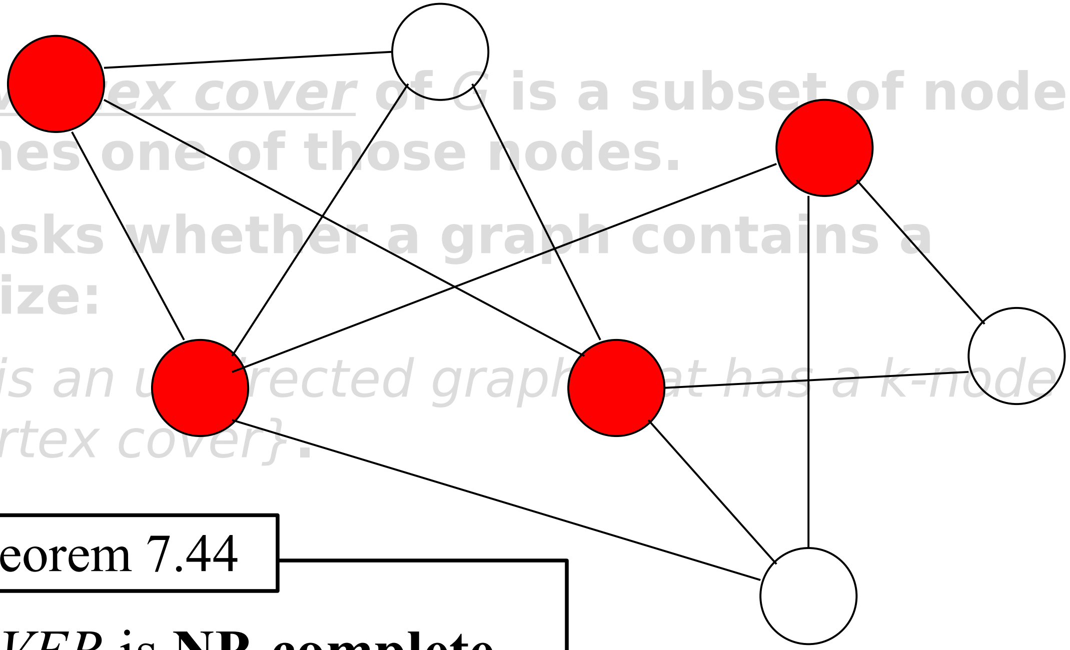
Theorem 7.44

VERTEX-COVER is **NP-complete**

VERTEX-COVER (yet, another NP-complete problem)

- **Description**
  - **If *G* is undirected graph, a *vertex cover* of *G* is a subset of nodes where every edge of *G* touches one of those nodes.**
  - **The *vertex cover* problem asks whether a graph contains a vertex cover of a specified size:**

VERTEX-COVER = {⟨*G*, *k*⟩| *G* is an undirected graph that has a *k*-node vertex cover}.



Theorem 7.44

VERTEX-COVER is **NP-complete**

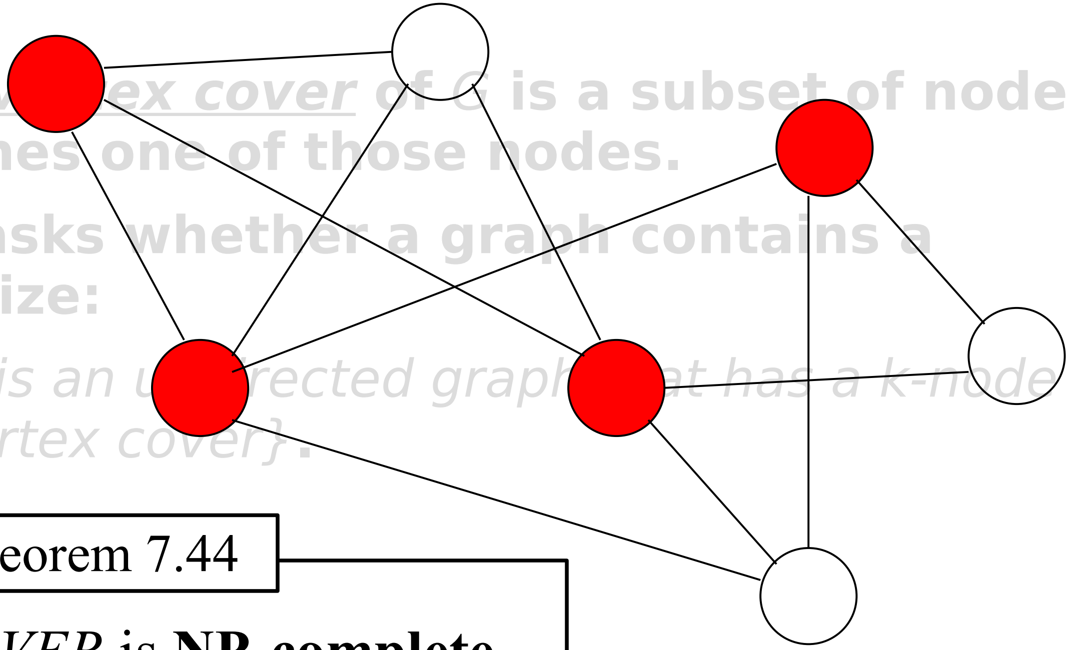
VERTEX-COVER (yet, another NP-complete problem)

- Description
  - If G is undirected graph, a *vertex cover* of G is a subset of nodes where every edge of G touches one of those nodes.
  - The vertex cover problem asks whether a graph contains a vertex cover of a specified size:

VERTEX-COVER = {⟨*G*, *k*⟩| *G* is an undirected graph that has a *k*-node vertex cover}.

S **is a vertex cover for this graph**

$|S| = 4$



Theorem 7.44

VERTEX-COVER is **NP-complete**

VERTEX-COVER (yet, another NP-complete problem)

- Description
  - If G is undirected graph, a *vertex cover* of G is a subset of nodes where e … touches one of those nodes.
  - The *ve* … blem asks whether a graph contains a vertex c … ified size:

    VERTEX-COVER = {⟨G, k⟩| G is an undirected graph that has a k-node vertex cover}.

How can we prove this?

Theorem 7.44

VERTEX-COVER is **NP-complete**

VERTEX-COVER (yet, another NP-complete problem)

- Description
  - If G is undirected graph, a *ve* … les where e … touches
  - The *ve* … blem asks whether a graph contains a vertex c … ified size:

    VERTEX-COVER = {⟨G, k⟩| G is an undirected graph that has a k-node vertex cover}.

1 *Show that a vertex-cover solution can be verified in polynomial time*

How can we prove this?

Theorem 7.44

VERTEX-COVER is **NP-complete**

VERTEX-COVER (yet, another NP-complete problem)

- Description
  - If *G* is undirected graph, a *ve*... les where e... touches
  - The *ve*... blem as... whether a graph contains a vertex c... ified size

    VERTEX-COVER = {⟨G, k⟩| G is a ... e vertex cover}.

How can we prove this?

1. *Show that a vertex-cover solution can be verified in polynomial time*
2. *Show that all NP problems are reducible to it*

Theorem 7.44

VERTEX-COVER is **NP-complete**

VERTEX-COVER (yet, another NP-complete problem)

- 1) Verifying a certificate
  - A certificate is just a vertex cover of size k.

VERTEX-COVER (yet, another NP-complete problem)

- **1) Verifying a certificate**
  - **A certificate is just a vertex cover of size k.**
  - **A Vertex-cover verifier can clearly verify whether the certificate is valid or not in polynomial time.**

VERTEX-COVER (yet, another NP-complete problem)

- **1) Verifying a certificate**
  - **A certificate is just a vertex cover of size k.**
  - **A Vertex-cover verifier can clearly verify whether the certificate is valid or not in polynomial time.**
    1) **Check if cover size is k. If not, <u>reject</u>.**

VERTEX-COVER (yet, another NP-complete problem)

- **1) Verifying a certificate**
  - **A certificate is just a vertex cover of size** k**.**
  - **A Vertex-cover verifier can clearly verify whether the certificate is valid or not in polynomial time.**
    1) **Check if cover size is** k**. If not, <u>reject</u>.**
    2) **Iterate over the edges of the graph and check whether one of the nodes (at the two ends of the edge) exists in the vertex cover presented in the certificate.**

VERTEX-COVER (yet, another NP-complete problem)

- **1) Verifying a certificate**

  - **A certificate is just a vertex cover of size k.**
  - **A Vertex-cover verifier can clearly verify whether the certificate is valid or not in polynomial time.**

    1)Check if cover size is k. If not, <u>reject</u>.

    2)Iterate over the edges of the graph and check whether one of the nodes (at the two ends of the edge) exists in the vertex cover presented in the certificate.

    3)If it finds one edge whose both nodes are not in the certificate, then it <u>rejects</u> the solution. Otherwise, it <u>accepts</u>.

VERTEX-COVER (yet, another NP-complete problem)

- **2) Polynomial time reduction (p-reduction)**
  - **Show that *3SAT* is p-reducible to *VERTEX-COVER*.**

VERTEX-COVER (yet, another NP-complete problem)

- **2) Polynomial time reduction (p-reduction)**
  - **Show that *3SAT* is p-reducible to *VERTEX-COVER*.**
  - **This entails converting a 3cnf-formula into a graph *G* and a number *k*.**

VERTEX-COVER (yet, another NP-complete problem)

- **2) Polynomial time reduction (p-reduction)**
  - **Show that *3SAT* is p-reducible to *VERTEX-COVER*.**
  - **This entails converting a 3cnf-formula into a graph G and a number k.**
  - **The formula φ is satisfiable whenever G has a vertex cover with k nodes.**

VERTEX-COVER (yet, another NP-complete problem)

- **2) Polynomial time reduction (p-reduction)**
  - **Show that *3SAT* is p-reducible to *VERTEX-COVER*.**
  - **This entails converting a 3cnf-formula into a graph G and a number k.**
  - **The formula φ is satisfiable whenever G has a vertex cover with k nodes.**
  - **Simply put, G should simulate φ.**

VERTEX-COVER (yet, another NP-complete problem)

- **2) Polynomial time reduction (p-reduction)**

Let φ be a 3CNF formula with $\boldsymbol{m}$ variables and $\boldsymbol{l}$ clauses

$$\varphi = \underbrace{(x_1 \vee x_2 \vee x_3)}_{\text{Clause 1}} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee \overline{x_4})}_{\text{Clause 2}} \wedge \underbrace{(\overline{x_1} \vee x_3 \vee x_4)}_{\text{Clause 3}}$$

$\boldsymbol{m} = 4$
$\boldsymbol{l} = 3$

VERTEX-COVER (yet, another NP-complete problem)

- **2) Polynomial time reduction (p-reduction)**

Formula φ can be converted to a graph G such that:

φ is satisfiable ***iff***

G has a vertex cover
of size $\mathbf{k} = \boldsymbol{m} + \boldsymbol{2l}$

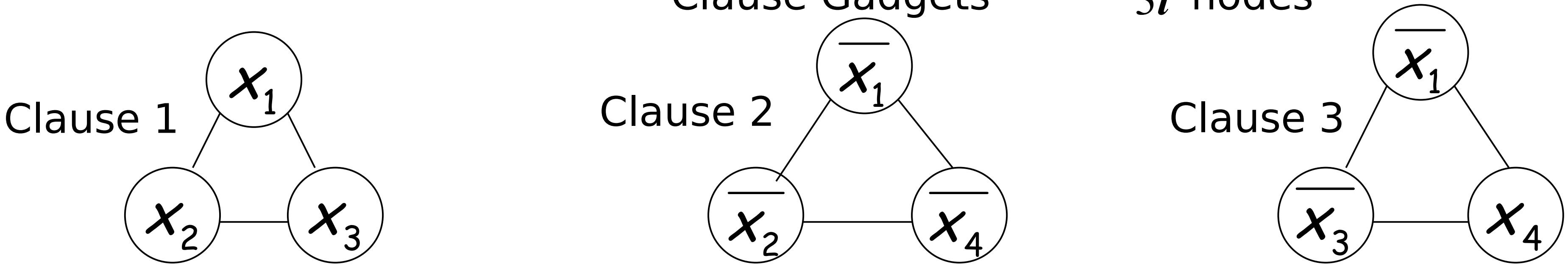
VERTEX-COVER (yet, another NP-complete problem)

- **2) Polynomial time reduction (p-reduction)**

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4)$$

Variable Gadgets $\quad 2m$ nodes

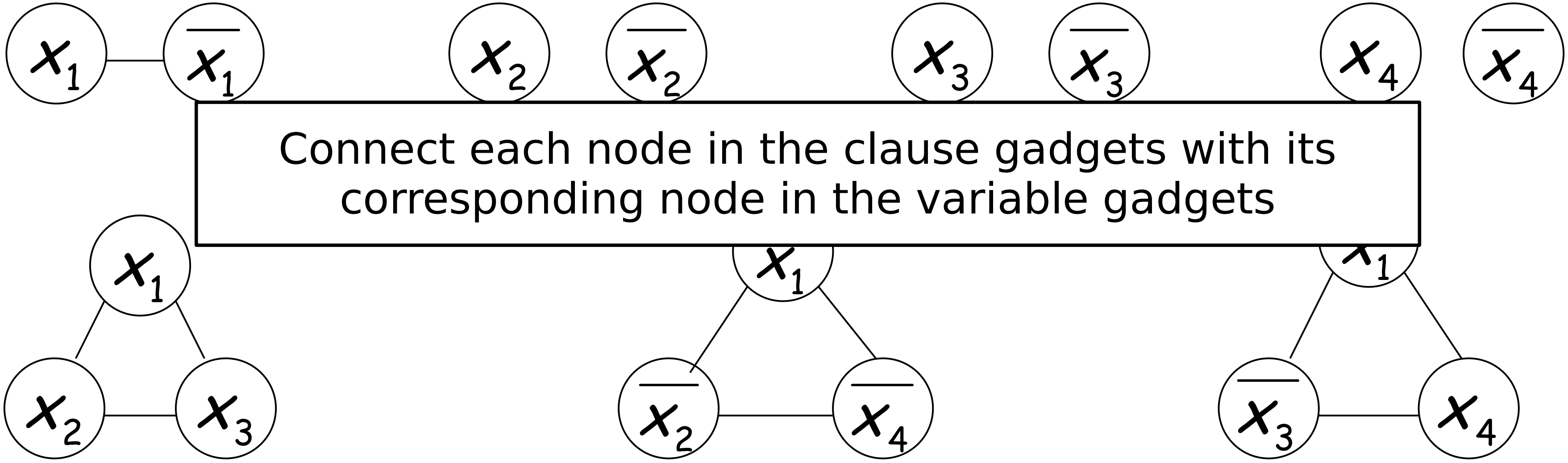
Clause Gadgets $\quad 3l$ nodes



VERTEX-COVER (yet, another NP-complete problem)

- **2) Polynomial time reduction (p-reduction)**

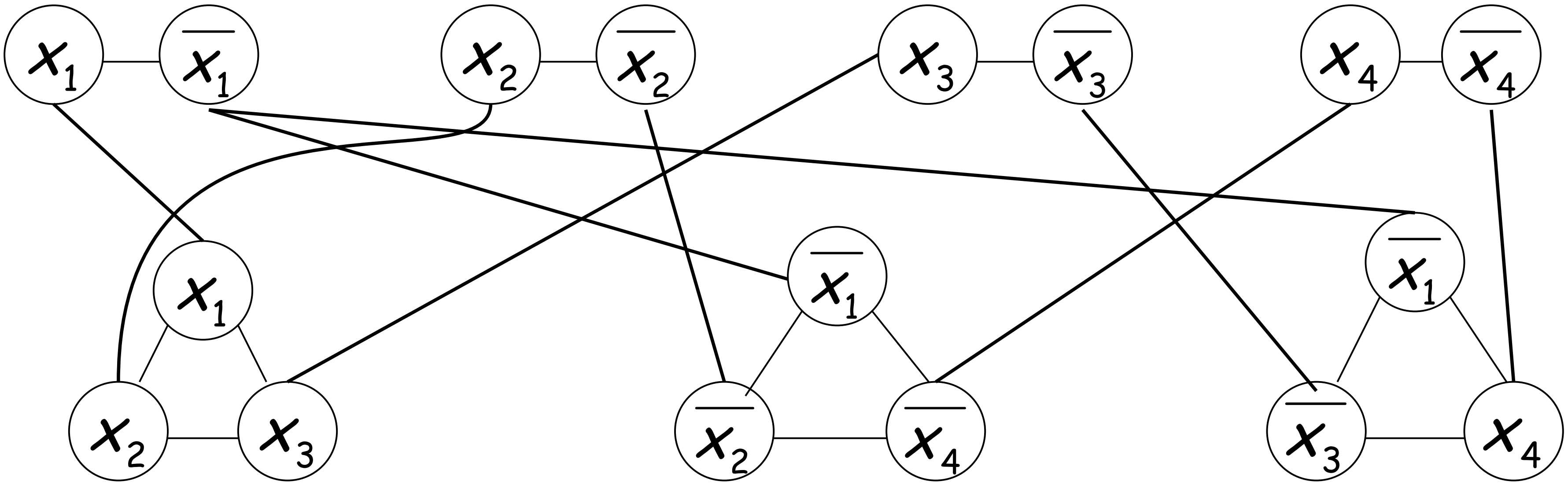
$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4)$$



VERTEX-COVER (yet, another NP-complete problem)

- **2) Polynomial time reduction (p-reduction)**

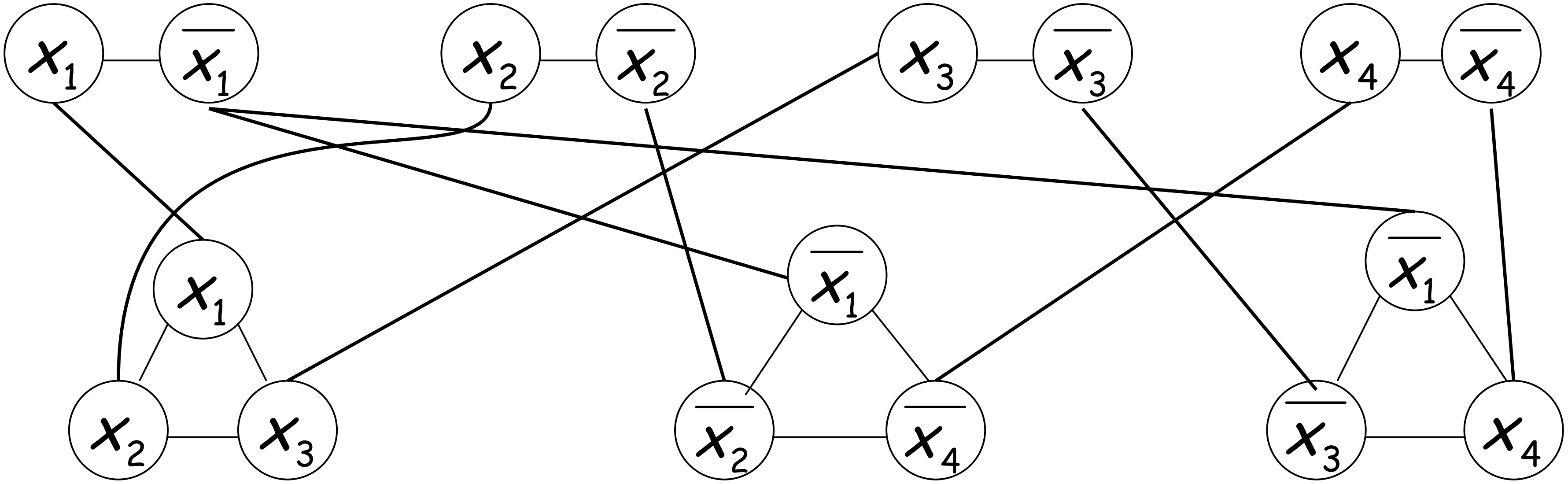
$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4)$$



VERTEX-COVER (yet, another NP-complete problem)

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4)$$

Start with a satisfying assignment $x_1 = 1$ $x_2 = 0$ $x_3 = 0$ $x_4 = 1$

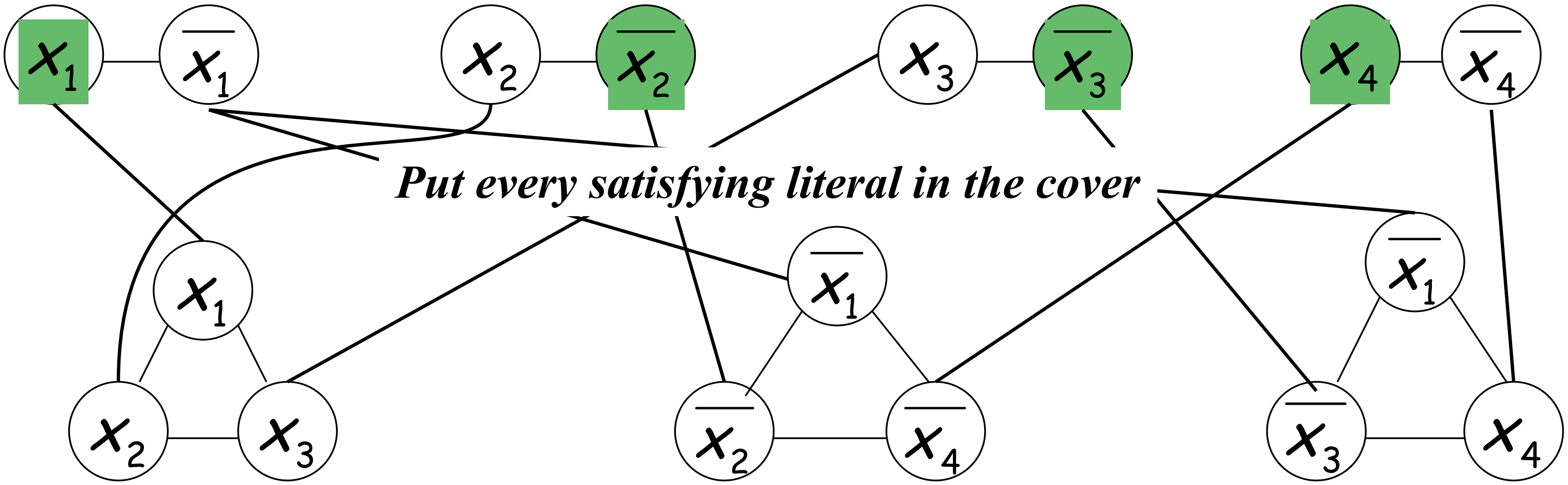


VERTEX-COVER (yet, another NP-complete problem)

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4)$$

Start with a satisfying assignment

$x_1 = 1$ $x_2 = 0$ $x_3 = 0$ $x_4 = 1$

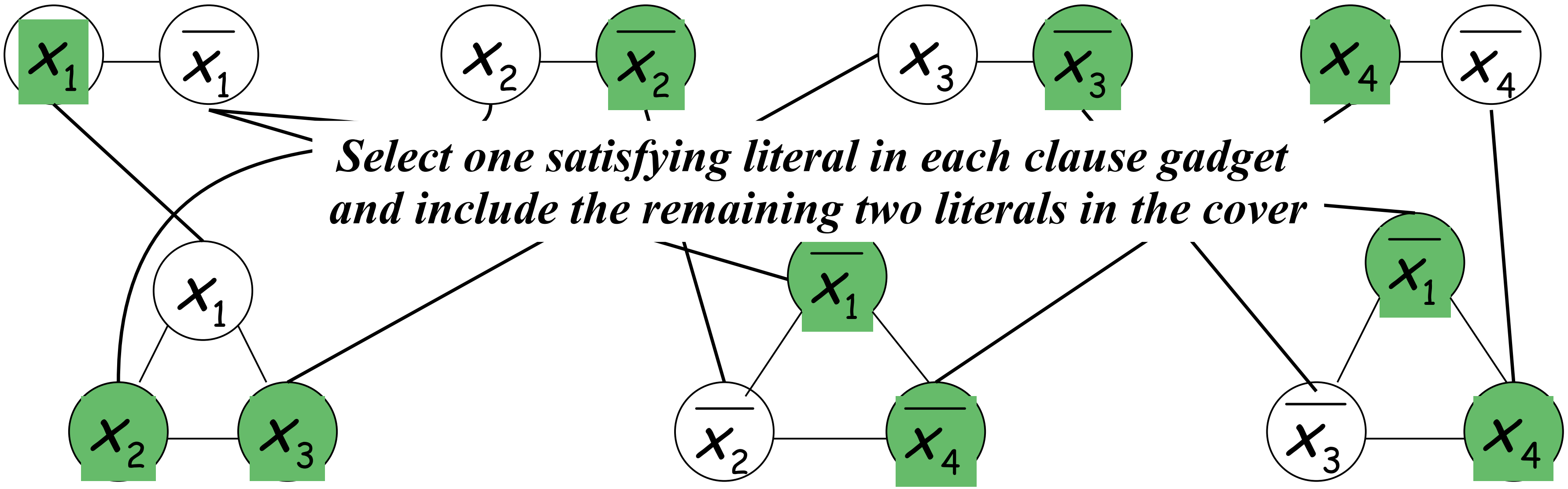


VERTEX-COVER (yet, another NP-complete problem)

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4)$$

Start with a satisfying assignment

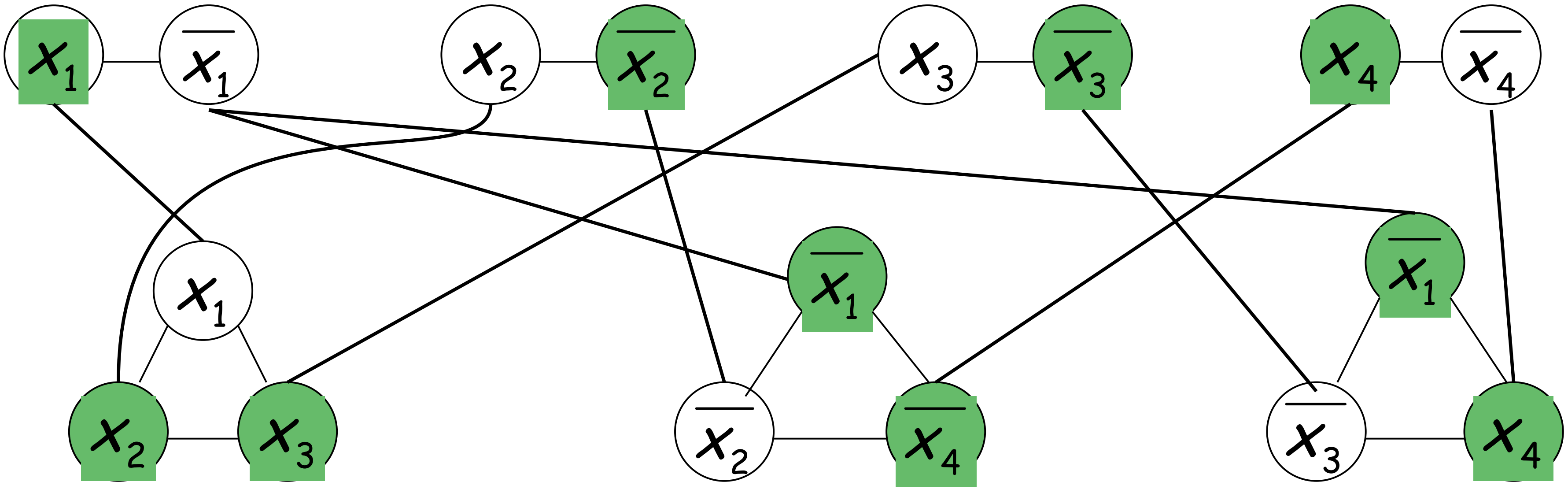
$x_1 = 1$ $x_2 = 0$ $x_3 = 0$ $x_4 = 1$



VERTEX-COVER (yet, another NP-complete problem)

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4)$$

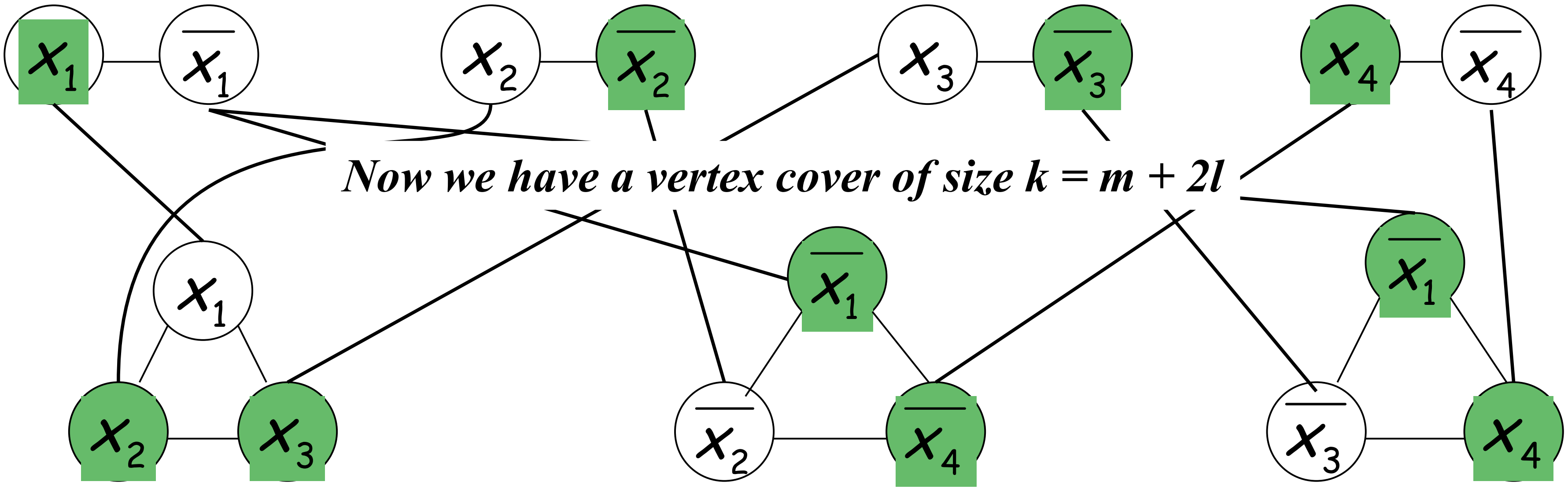
Start with a satisfying assignment $x_1 = 1$ $x_2 = 0$ $x_3 = 0$ $x_4 = 1$



VERTEX-COVER (yet, another NP-complete problem)

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4)$$

Start with a satisfying assignment $x_1 = 1$ $x_2 = 0$ $x_3 = 0$ $x_4 = 1$



Now we have a vertex cover of size k = m + 2l

VERTEX-COVER **(yet, another NP-complete problem)**

We've shown that we can covert an instance of ***3SAT*** *into an instance of* ***VERTEX-COVER*** *(in polynomial time)*

VERTEX-COVER (yet, another NP-complete problem)

We've shown that we can covert an instance of ***3SAT*** *into an instance of* ***VERTEX-COVER*** *(in polynomial time)*

Therefore, we've reduced in polynomial time 3SAT to VERTEX-COVER

VERTEX-COVER (yet, another NP-complete problem)

We've shown that we can covert an instance of ***3SAT*** *into an instance of* ***VERTEX-COVER*** *(in polynomial time)*

Therefore, we've reduced in polynomial time 3SAT to VERTEX-COVER

This concludes the proof for Theorem 7.44