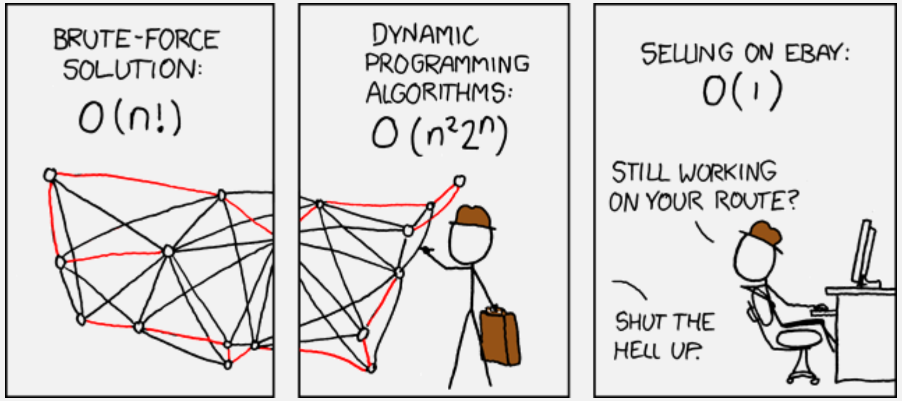
More More NP-Complete Problems

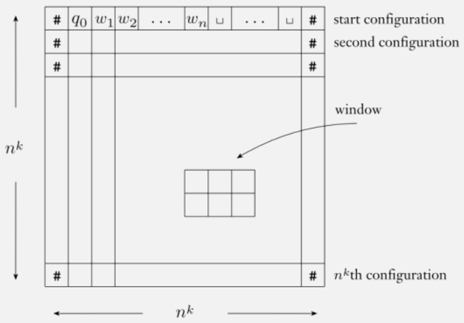
Monday, May 9, 2022



Announcements

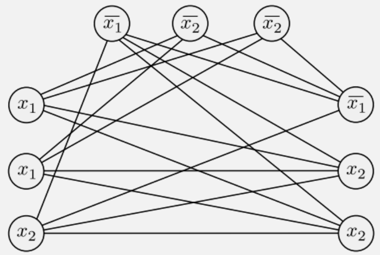
- HW 11 in
  - ~~Due Tues 5/3 11:59pm EST~~
- HW 12 out
  - Due Wed 5/11 11:59pm EST
  - Last HW!
- Last week: 2 lectures left!
- Course evals today

NP-Complete problems, so far

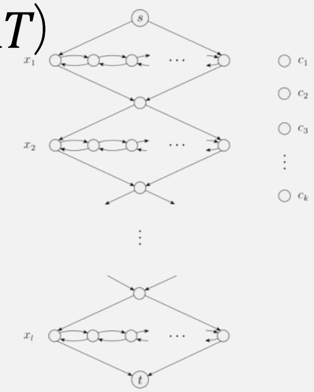


- $SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula}\}$ (Cook-Levin Theorem)

- $3SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable 3cnf-formula}\}$ (reduce from *SAT*)



- $CLIQUE = \{\langle G, k \rangle \mid G \text{ is an undirected graph with a } k\text{-clique}\}$ (reduce from 3*SAT*)



- $HAMPATH = \{\langle G, s, t \rangle \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t\}$ (reduce from 3*SAT*)

- $UHAMPATH = \{\langle G, s, t \rangle \mid G \text{ is a undirected graph with a Hamiltonian path from } s \text{ to } t\}$ (HW 12)

Today: More **NP**-Complete problems

- $SUBSET\text{-}SUM = \{\langle S, t\rangle \mid S = \{x_1, \ldots, x_k\}, \text{ and for some } \{y_1, \ldots, y_l\} \subseteq \{x_1, \ldots, x_k\}, \text{ we have } \Sigma y_i = t\}$
  - (reduce from *3SAT*)

- $VERTEX\text{-}COVER = \{\langle G, k\rangle \mid G \text{ is an undirected graph that has a } k\text{-node vertex cover}\}$
  - (reduce from *3SAT*)

Theorem: *SUBSET-SUM* is NP-complete

$$\textit{SUBSET-SUM} = \{\langle S, t\rangle \mid S = \{x_1, \ldots, x_k\}, \text{ and for some } \{y_1, \ldots, y_l\} \subseteq \{x_1, \ldots, x_k\}, \text{ we have } \Sigma y_i = t\}$$

50 kg??

THEOREM

Using: If B is NP-complete and $B \leq_{\mathrm{P}} C$ for C in NP, then C is NP-complete.

3 steps to prove a language is **NP**-complete:

1. Show C is in **NP**
2. Choose B, the known **NP**-complete problem to reduce from
3. Show a poly time mapping reduction from B to C

Theorem: *SUBSET-SUM* is NP-complete

$SUBSET\text{-}SUM = \{\langle S,t\rangle \mid S = \{x_1,\ldots,x_k\}, \text{ and for some } \{y_1,\ldots,y_l\} \subseteq \{x_1,\ldots,x_k\}, \text{ we have } \Sigma y_i = t\}$

3 steps to prove *SUBSET-SUM* is **NP**-complete:

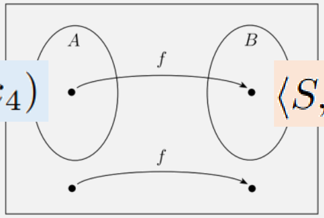
☑ 1. Show *SUBSET-SUM* is in **NP**

☑ 2. Choose the **NP**-complete problem to reduce from: *3SAT*

3. Show a poly time mapping reduction from *3SAT* to *SUBSET-SUM*

To show poly time mapping reducibility:
1. create **computable fn**,
2. show that it **runs in poly time**,
3. then show **forward direction** of mapping red.,
4. and **reverse direction** (or **contrapositive** of **reverse direction**)

$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4)$

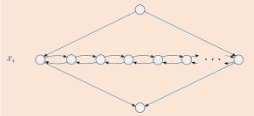


$\langle S,t\rangle \mid S = \{x_1,\ldots,x_k\}$

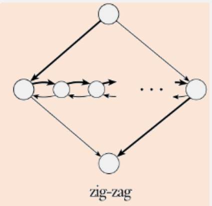
Review: Reducing from *3SAT*

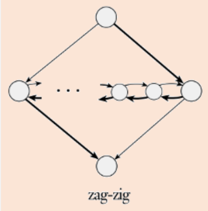
Create a **computable function** mapping formula to “gadgets”:

- Clause → some “gadget”, e.g., 

- Variable → another “gadget”, e.g., 

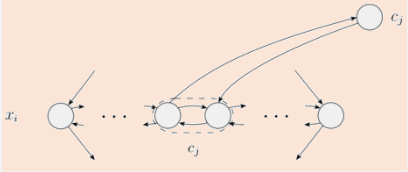
  Gadget is typically used in two “opposite” ways, e.g.:
  - ZIG when var is assigned TRUE, or
  - ZAG when var is assigned FALSE



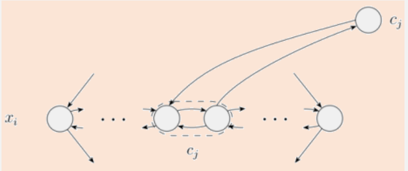


<u>NOTE</u>: “gadgets” are not always graphs

Then connect “gadgets” according to clause literals:



- Literal x_i in clause c_j → gadget x_i “detours” to c_j
- Literal $\overline{x_i}$ in clause c_j → gadget x_i “reverse detours” to c_j



Computable Fn: 3cnf → $\langle S, t\rangle$

E.g., $(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3 \vee \cdots) \wedge \cdots \wedge (\overline{x_3} \vee \cdots \vee \cdots)$ ➡

- Assume formula has:
  - ***l*** variables $x_1, \ldots, x_l$
  - ***k*** clauses $c_1, \ldots, c_k$
- Computable function *f* maps:
  - Variable x_i → two numbers y_i and z_i
  - Clause c_j → two numbers g_j and h_j
  - Digits arranged as rows in a table …
- Each number has max ***l+k*** **digits:**
  - Literal x_i in clause c_j → y_i: l+j^{th} digit = 1
  - Literal $\overline{x_i}$ in clause c_j → z_i: l+j^{th} digit = 1
- Sum is ***l*** 1s followed by ***k*** 3s

y_i and z_i: i^{th} digit = 1

y_i: l+j^{th} digit = 1 if c_j has x_i

z_i: l+j^{th} digit = 1 if c_j has $\overline{x_i}$

	1	2	3	4	⋯	l	c_1	c_2	⋯	c_k
y_1	1	0	0	0	⋯	0	1	0	⋯	0
z_1	1	0	0	0	⋯	0	0	0	⋯	0
y_2		1	0	0	⋯	0	0	1	⋯	0
z_2		1	0	0	⋯	0	1	0	⋯	0
y_3			1	0	⋯	0	1	1	⋯	0
z_3			1	0	⋯	0	0	0	⋯	1
⋮					⋱	⋮	⋮		⋮	⋮
y_l						1	0	0	⋯	0
z_l						1	0	0	⋯	0
g_1							1	0	⋯	0
h_1							1	0	⋯	0
g_2								1	⋯	0
h_2								1	⋯	0
⋮									⋱	⋮
g_k										1
h_k										1
t	1	1	1	1	⋯	1	3	3	⋯	3

g_j and h_j: l+j^{th} digit = 1, To help get correct sum

The sum

Theorem: *SUBSET-SUM* is NP-complete

$$SUBSET\text{-}SUM = \{\langle S,t\rangle \mid S = \{x_1,\ldots,x_k\}, \text{ and for some } \{y_1,\ldots,y_l\} \subseteq \{x_1,\ldots,x_k\}, \text{ we have } \Sigma y_i = t\}$$

3 steps to prove *SUBSET-SUM* is **NP**-complete:

☑ 1. Show *SUBSET-SUM* is in **NP**

☑ 2. Choose the **NP**-complete problem to reduce from: *3SAT*

3. Show a poly time mapping reduction from *3SAT* to *SUBSET-SUM*

To show poly time mapping reducibility:

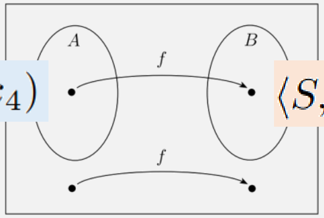
☑ 1. create **computable fn**,

➡ 2. show that it **runs in poly time**,

3. then show **forward direction** of mapping red.,

4. and **reverse direction** (or **contrapositive** of **reverse direction**)

$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4)$



$\langle S,t\rangle \mid S = \{x_1,\ldots,x_k\}$

Polynomial Time?

E.g., $(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3 \vee \cdots) \wedge \cdots \wedge (\overline{x_3} \vee \cdots \vee \cdots)$ ➡

- Assume formula has:
  - ***l*** variables $x_1, \ldots, x_l$
  - ***k*** clauses $c_1, \ldots, c_k$
- Table size: $(l + k) * (2l + 2k)$
  - Creating it requires constant number of passes over the table
  - Num variables l = at most $3k$
- Total: $O(k^2)$

	1	2	3	4	$\cdots$	l	c_1	c_2	$\cdots$	c_k
y_1	1	0	0	0	$\cdots$	0	1	0	$\cdots$	0
z_1	1	0	0	0	$\cdots$	0	0	0	$\cdots$	0
y_2		1	0	0	$\cdots$	0	0	1	$\cdots$	0
z_2		1	0	0	$\cdots$	0	1	0	$\cdots$	0
y_3			1	0	$\cdots$	0	1	1	$\cdots$	0
z_3			1	0	$\cdots$	0	0	0	$\cdots$	1
$\vdots$					$\ddots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
y_l						1	0	0	$\cdots$	0
z_l						1	0	0	$\cdots$	0
g_1							1	0	$\cdots$	0
h_1							1	0	$\cdots$	0
g_2								1	$\cdots$	0
h_2								1	$\cdots$	0
$\vdots$									$\ddots$	$\vdots$
g_k										1
h_k										1
t	1	1	1	1	$\cdots$	1	3	3	$\cdots$	3

Theorem: *SUBSET-SUM* is NP-complete

SUBSET-SUM $= \{\langle S,t\rangle \mid S=\{x_1,\ldots,x_k\}$, and for some $\{y_1,\ldots,y_l\}\subseteq\{x_1,\ldots,x_k\}$, we have $\Sigma y_i = t\}$

3 steps to prove *SUBSET-SUM* is **NP**-complete:

☑ 1. Show *SUBSET-SUM* is in **NP**

☑ 2. Choose the **NP**-complete problem to reduce from: *3SAT*

3. Show a poly time mapping reduction from *3SAT* to *SUBSET-SUM*

To show poly time mapping reducibility:

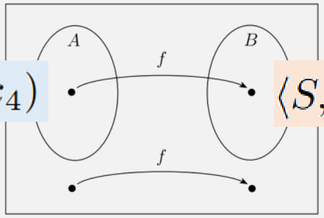
☑ 1. create **computable fn**,

☑ 2. show that it **runs in poly time**,

3. then show **forward direction** of mapping red.,

4. and **reverse direction**
   (or **contrapositive** of **reverse direction**)

$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4)$



$\langle S,t\rangle \mid S=\{x_1,\ldots,x_k\}$

ϕ is a satisfiable 3cnf-formula $\iff$ $f(\langle\phi\rangle) = \langle S, t\rangle$ where some subset of S sums to t

⇒ If formula is satisfiable …

- Sum t = $\boldsymbol{l}$ 1s followed by $\boldsymbol{k}$ 3s
- Choose for the subset …
  - y_i if x_i = TRUE
  - z_i if x_i = FALSE
  - and some of g_i and h_i to make the sum t
- … Then this subset of S must sum to t bc:
  - Left digits:
    - only one of y_i or z_i is in S
  - Right digits:
    - Top right: Each column sums to 1, 2, or 3
      - Because each clause has 3 literals
    - Bottom right:
      - Can always use g_i and/or h_i to make column sum to 3

	1	2	3	4	⋯	l	c_1	c_2	⋯	c_k
y_1	1	0	0	0	⋯	0	1	0	⋯	0
z_1	1	0	0	0	⋯	0	0	0	⋯	0
y_2		1	0	0	⋯	0	0	1	⋯	0
z_2		1	0	0	⋯	0	1	0	⋯	0
y_3			1	0	⋯	0	1	1	⋯	0
z_3			1	0	⋯	0	0	0	⋯	1
⋮					⋱	⋮	⋮		⋮	⋮
y_l						1	0	0	⋯	0
z_l						1	0	0	⋯	0
g_1							1	0	⋯	0
h_1							1	0	⋯	0
g_2								1	⋯	0
h_2								1	⋯	0
⋮									⋱	⋮
g_k										1
h_k										1
t	1	1	1	1	⋯	1	3	3	⋯	3

Each column:
- At least one 1
- At most 3 1s

S only includes one of these

g_j and h_j: help get the correct sum

So each column sum (for left digits) is 1

ϕ is a satisfiable 3cnf-formula $\iff$ $f(\langle\phi\rangle) = \langle S, t\rangle$ where some sub… o t

$\Leftarrow$ If a subset of S sums to t …

The only way to do it is as prev described:

- It can only include either y_i or z_i
  - Because each left digit column must sum to 1
  - And no carrying is possible
- Also, since each right digit column must sum to 3:
  - And only 2 can come from g_i and h_i
  - Then for every right column, some y_i or z_i in the subset has a 1 in that column
- … Then table must have been created from a sat. ϕ:
  - x_i = TRUE if y_i in the subset
  - x_i = FALSE if z_i in the subset
- This is satisfying because:
  - Table was constructed so 1 in column c_j for y_i or z_i means that variable x_i satisfies clause c_j
  - We already determined, for every right column, some number in the subset has a 1 in the column
  - So all clauses are satisfied

	1	2	3	4	$\cdots$	l	c_1	c_2	$\cdots$	c_k
y_1	1	0	0	0	$\cdots$	0	1	0	$\cdots$	0
z_1	1	0	0	0	$\cdots$	0	0	0	$\cdots$	0
y_2		1	0	0	$\cdots$	0	0	1	$\cdots$	0
z_2		1	0	0	$\cdots$	0	1	0	$\cdots$	0
y_3			1	0	$\cdots$	0	1	1	$\cdots$	0
z_3			1	0	$\cdots$	0	0	0	$\cdots$	1
$\vdots$					$\ddots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
						1	0	0	$\cdots$	0
						1	0	0	$\cdots$	0
							1	0	$\cdots$	0
							1	0	$\cdots$	0
								1	$\cdots$	0
h_2								1	$\cdots$	0
$\vdots$									$\ddots$	$\vdots$
g_k										1
h_k										1
t	1	1	1	1	$\cdots$	1	3	3	$\cdots$	3

Subset must have some number with 1 in each right column

S only includes y_i or z_i

In each right column, g_i and h_i can account for at most 2

Because each column sum (for left digits) is 1

More **NP**-Complete problems

☑ • $\mathit{SUBSET\text{-}SUM} = \{\langle S, t\rangle \mid S = \{x_1, \ldots, x_k\}, \text{ and for some } \{y_1, \ldots, y_l\} \subseteq \{x_1, \ldots, x_k\}, \text{ we have } \Sigma y_i = t\}$

  - (reduce from *3SAT*)

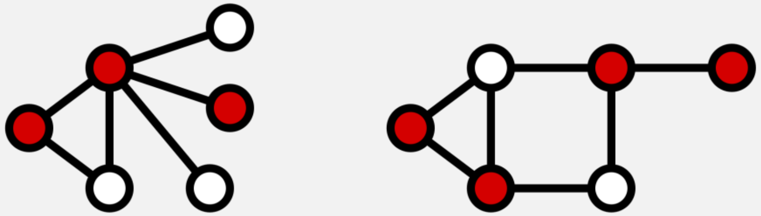
- $\mathit{VERTEX\text{-}COVER} = \{\langle G, k\rangle \mid G \text{ is an undirected graph that has a } k\text{-node vertex cover}\}$

  - (reduce from *3SAT*)

Theorem: *VERTEX-COVER* is NP-complete

$$VERTEX\text{-}COVER = \{\langle G, k\rangle \mid G \text{ is an undirected graph that has a } k\text{-node vertex cover}\}$$

- A **vertex cover** of a graph is …
  - … a subset of its nodes where every edge touches one of those nodes



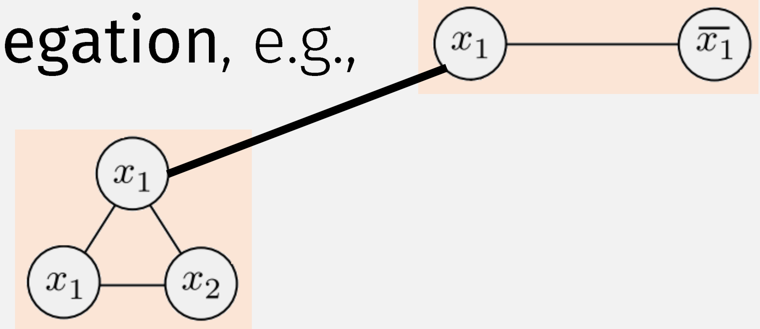
<u>Theorem</u>: *VERTEX-COVER* is NP-complete

VERTEX-COVER = $\{\langle G, k\rangle |$ G is an undirected graph that has a k-node vertex cover$\}$

- A **vertex cover** of a graph is …
  - … a subset of its nodes where every edge touches one of those nodes

Proof Sketch: Reduce *3SAT* to *VERTEX-COVER*

- The <u>reduction</u> maps:
- Variable x_i → 2 connected nodes
  - corresponding to the var and its negation, e.g.,
- Clause → 3 connected nodes
  - corresponding to its literals, e.g.,

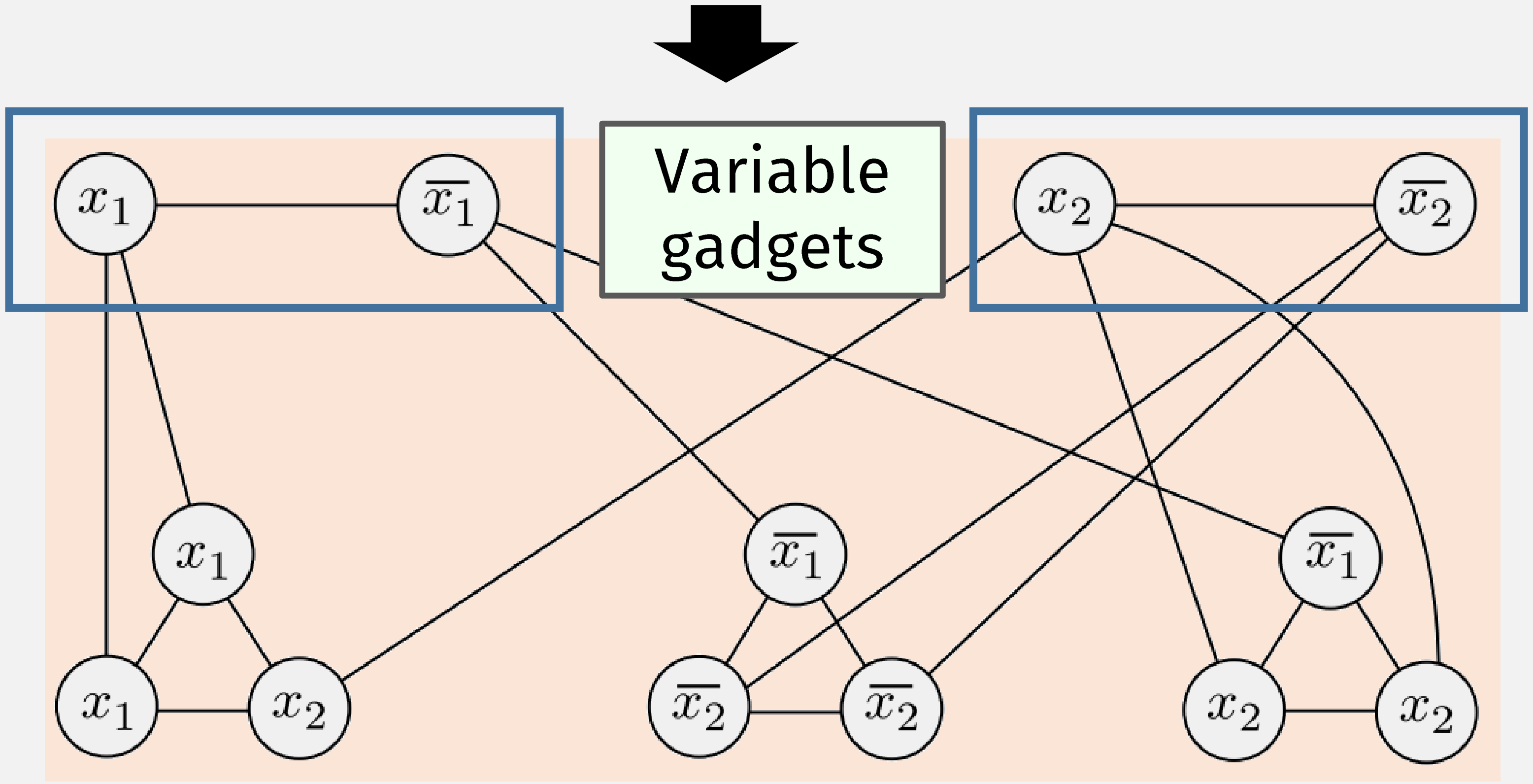


- Additionally,
  - <u>connect</u> var and clause gadgets by …
  - … <u>connecting</u> nodes that correspond to the same literal

VERTEX-COVER example

VERTEX-COVER = $\{\langle G, k\rangle |$ G is an undirected graph that has a k-node vertex cover$\}$

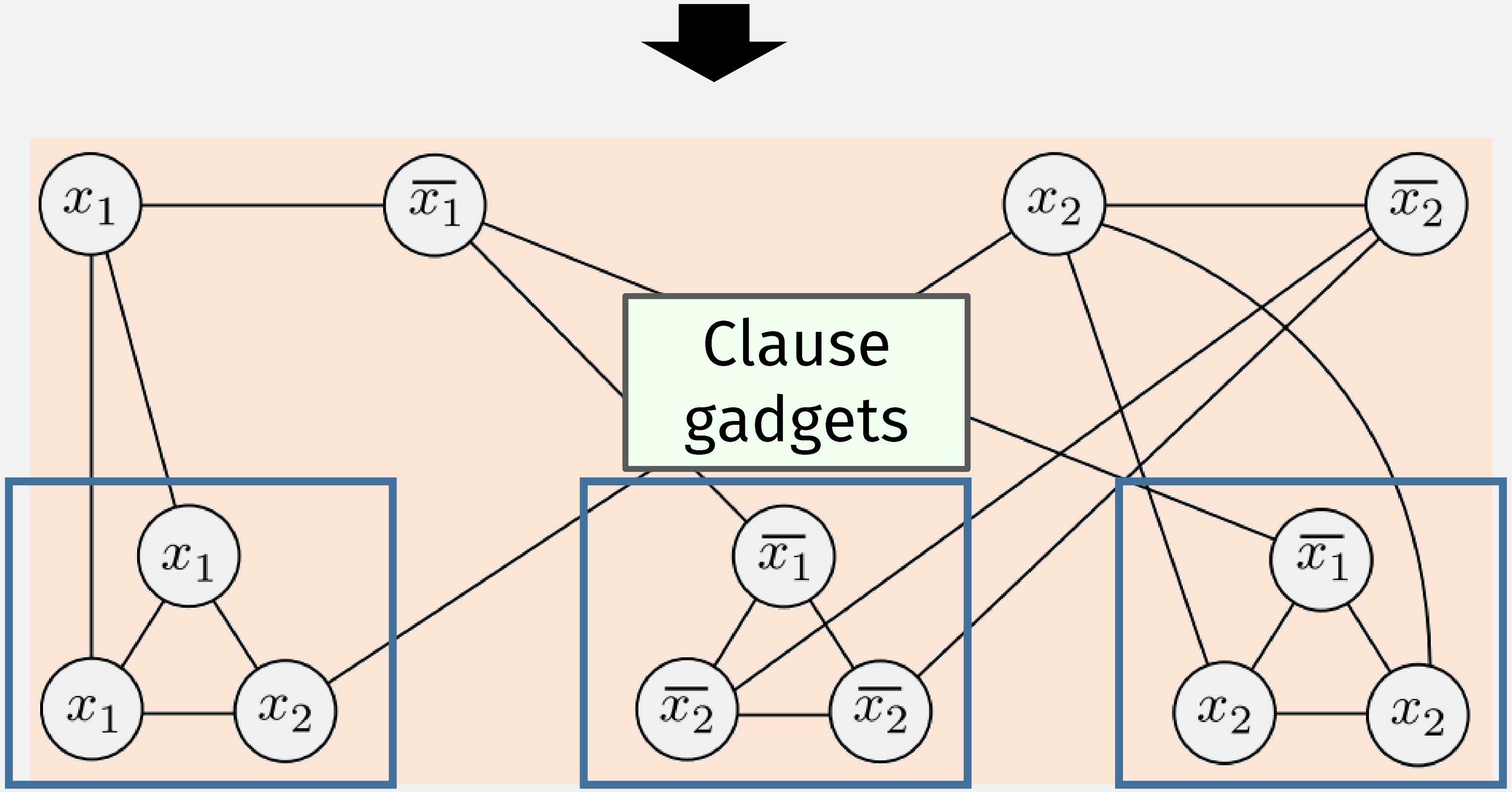
$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



VERTEX-COVER example

VERTEX-COVER = $\{\langle G, k\rangle \mid G$ is an undirected graph that has a k-node vertex cover$\}$

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

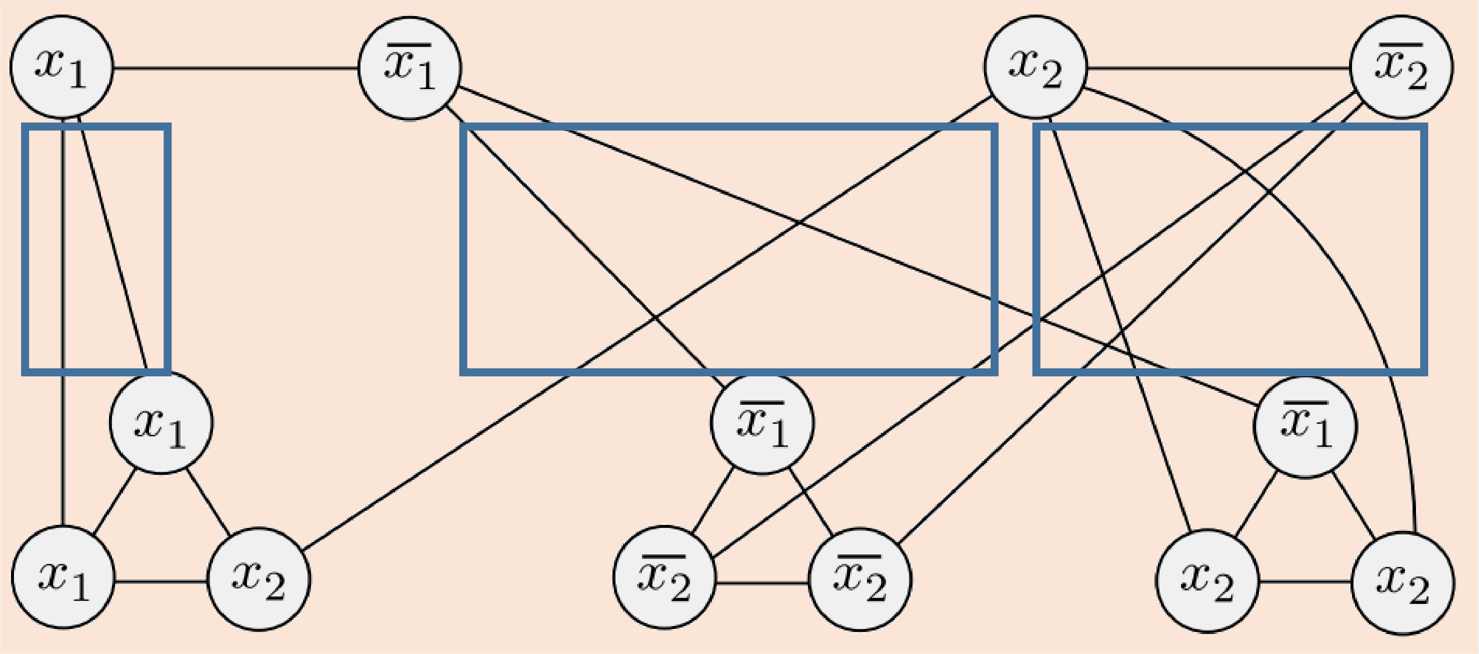


VERTEX-COVER example

VERTEX-COVER = $\{\langle G, k\rangle |$ G is an undirected graph that has a k-node vertex cover$\}$

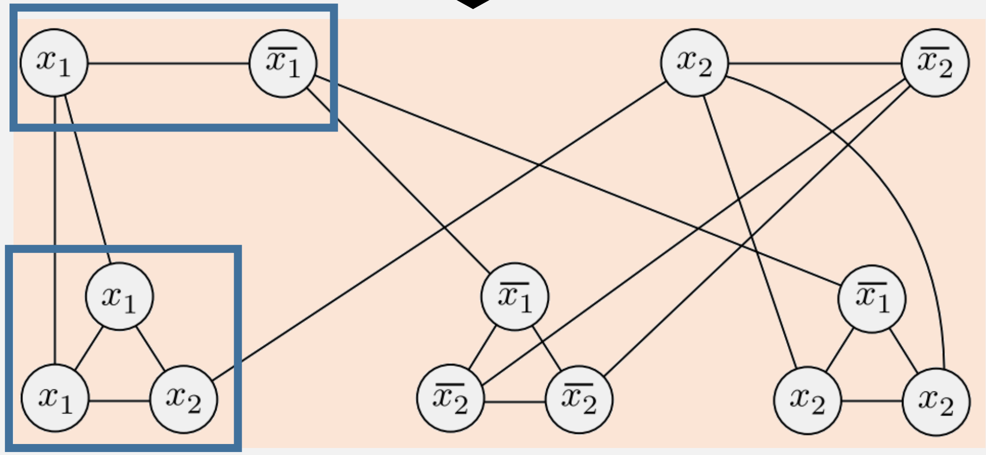
$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

Extra edges connecting variable and clause gadgets together



$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

VERTEX-COVER example



- If formula has …
  - ***m*** = # variables
  - ***l*** = # clauses
- Then graph has …
  - # nodes = 2 × #vars + 3 × #clauses = **2*m* + 3*l***

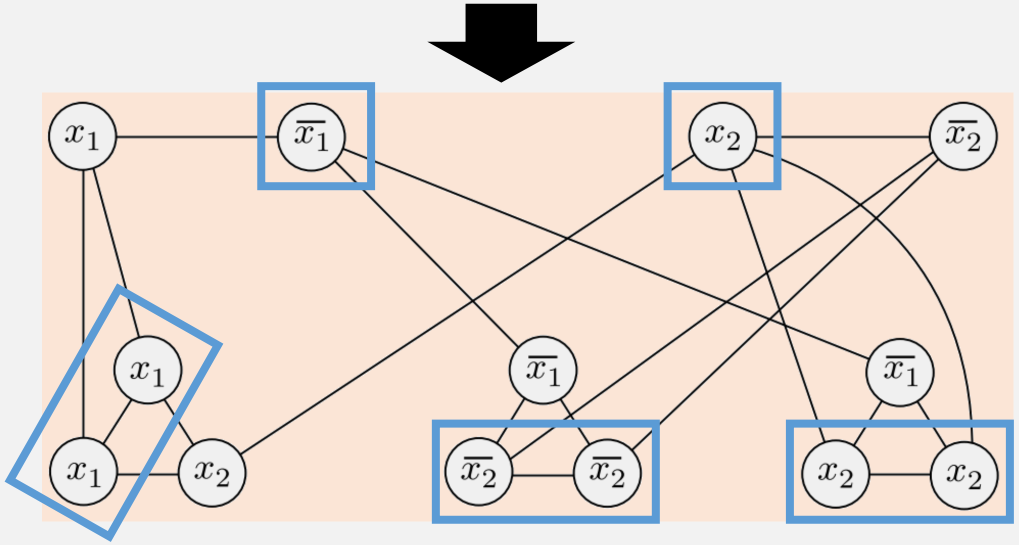
⇒ If satisfying assignment, then there is a ***k***-cover, where ***k* = *m* + 2*l***

- Nodes in the cover are:
  - In each of ***m*** var gadgets, choose 1 node corresponding to TRUE literal
  - For each of ***l*** clause gadgets, ignore 1 TRUE literal and choose other 2
  - Since there is satisfying assignment, each clause has a TRUE literal
  - Total nodes in cover = ***m* + 2*l***

VERTEX-COVER $= \{\langle G, k\rangle |$ G is an undirected graph that has a k-node vertex cover$\}$

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

VERTEX-COVER example



- If formula has …
  - $\boldsymbol{m}$ = # variables
  - $\boldsymbol{l}$ = # clauses
- Then graph has …
  - # nodes = $2\boldsymbol{m} + 3\boldsymbol{l}$

Example:
x_1 = FALSE
x_2 = TRUE

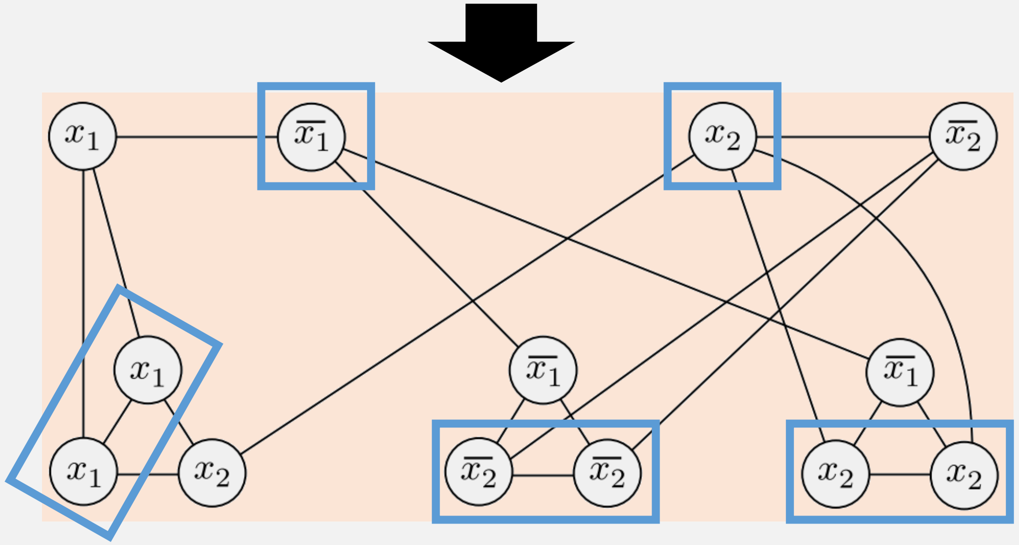
⇒ If satisfying assignment, then there is a $\boldsymbol{k}$-cover, where $\boldsymbol{k} = \boldsymbol{m} + \boldsymbol{2l}$

- Nodes in the cover are:
  - In each of $\boldsymbol{m}$ var gadgets, choose 1 node corresponding to TRUE literal
  - For each of $\boldsymbol{l}$ clause gadgets, ignore 1 TRUE literal and choose other 2
  - Since there is satisfying assignment, each clause has a TRUE literal
  - Total nodes in cover = $\boldsymbol{m} + \boldsymbol{2l}$

VERTEX-COVER = $\{\langle G, k\rangle \mid G$ is an undirected graph that has a k-node vertex cover$\}$

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

VERTEX-COVER example



- If formula has …
  - ***m*** = # variables
  - ***l*** = # clauses
- Then graph has …
  - # nodes = 2***m*** + 3***l***

Example:
x_1 = FALSE
x_2 = TRUE

⇐ If there is a ***k*** = ***m* + 2*l*** cover,

- Then it can only be a ***k***-cover as described on the last slide …
  - 1 node (and only 1) from each of "var" gadgets
  - 2 nodes (and only 2) from each "clause" gadget
  - Any other set of ***k*** nodes is not a cover
- Which means that input has satisfying assignment:
  - x_i = TRUE if node x_i is in cover, else x_i = FALSE

VERTEX-COVER $= \{\langle G, k\rangle \mid G$ is an undirected graph that has a k-node vertex cover$\}$

More **NP**-Complete problems

- ☑ $\textit{SUBSET-SUM} = \{\langle S, t\rangle \mid S = \{x_1, \ldots, x_k\}, \text{ and for some } \{y_1, \ldots, y_l\} \subseteq \{x_1, \ldots, x_k\}, \text{ we have } \Sigma y_i = t\}$
  - (reduce from *3SAT*)
- ☑ $\textit{VERTEX-COVER} = \{\langle G, k\rangle \mid G \text{ is an undirected graph that has a } k\text{-node vertex cover}\}$
  - (reduce from *3SAT*)

Course Evaluations 5/9
(no quiz)