NP–Complete Problems

The clique problem

- Clique in an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices, each pair of which is connected by an edge in E
- a clique is a complete subgraph of G
- Size of a clique – number of vertices it contains
- Clique problem – Optimization problem of finding a clique of maximum size in a graph

Decision Problem – Clique

- Whether a clique of a given size k exists in the graph
- Formal definition is
- CLIQUE = {<G, k> : G is a graph containing a clique of size k}

Naive Algorithm

- Determine whether a graph G = (V, E) with |V| vertices has a clique of size k
- List all k–subsets of V
- Check each one to see whether it forms a clique
- running time of this algorithm is $\Omega(k^2\binom{|V|}{k})$ which is polynomial if k is a constant

Naive Algorithm

- however, k could be near |V|/2, in which case the algorithm runs in superpolynomial time
- an efficient algorithm for the clique problem is unlikely to exist

Clique problem is NP–complete

- To show that CLIQUE $\in$ NP, for a given graph $G = (V, E)$, we use the set $V' \subseteq V$ of vertices in the clique as a certificate for G
- We can check whether V' is a clique in polynomial time by checking whether, for each pair $u,v \in V'$, the edge (u, v) belongs to E

Clique problem is NP–complete

- We prove that 3–CNF–SAT $\leq_P$ CLIQUE, which shows that the clique problem is NP–hard.

Instance of 3-CNF-SAT with k clauses → Reduction Algorithm → Graph G such that is Φ is statisfiable only if G has a clique of size k

Clique problem is NP–complete

- Let $\Phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ be a boolean formula in 3–CNF with k clauses
- For r = 1, 2, …, k, each clause C_r has exactly three distinct literals l_1^r, l_2^r, and l_3^r
- construct a graph G such that Φ is satisfiable if and only if G has a clique of size k.

Algorithm to Construct Graph

- For each clause $C_r = (l_1^r \vee l_2^r \vee$ and $l_3^r)$ in Φ, we place a triple of vertices , v_1^r, v_2^r, and v_3^r into V
- We put an edge between two vertices v_i^r and v_j^s if both of the following hold:
- v_i^r and v_j^s are in different triples, that is $r \neq s$
- their corresponding literals are consistent, that is, l_1^r is not the negation of l_j^s

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

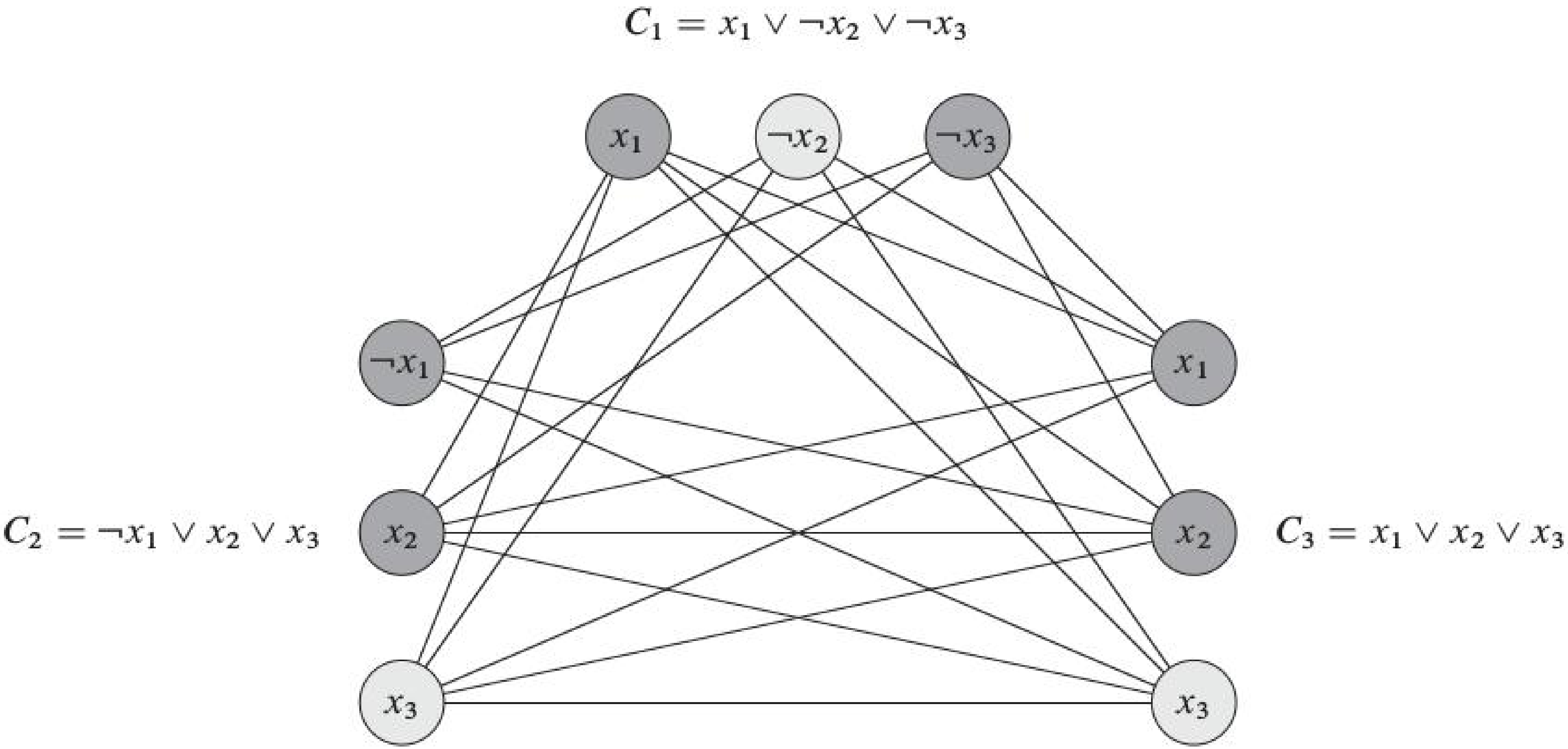


Figure 34.14 The graph G derived from the 3-CNF formula $\phi = C_1 \wedge C_2 \wedge C_3$, where $C_1 = (x_1 \vee \neg x_2 \vee \neg x_3)$, $C_2 = (\neg x_1 \vee x_2 \vee x_3)$, and $C_3 = (x_1 \vee x_2 \vee x_3)$, in reducing 3-CNF-SAT to CLIQUE. A satisfying assignment of the formula has $x_2 = 0$, $x_3 = 1$, and x_1 either 0 or 1. This

This transformation is a reduction

- We can easily build this graph from Φ in polynomial time
- If Φ has an satisfying assignment, then each clause C_r contains at least one literal l_i^r that is assigned 1 and each such literal corresponds to a vertex v_i^r
- Picking one such "true" literal from each clause yields a set V' of k vertices
- We claim that V' is a clique

This transformation is a reduction

- For any two vertices v_i^r, $v_j^s \in V'$, where $r \neq s$, both corresponding literals l_i^r and l_j^s map to 1 by the given satisfying assignment, and thus the literals cannot be complements
- Thus, by the construction of G, the edge (v_i^r and v_j^s) belongs to E

Converse is also True

- If G has a clique V' of size k then the boolean formula is satisfiable
- No edges in G connect vertices in the same triple, and so V' contains exactly one vertex per triple
- We can assign 1 to each literal l_i^r such that $v_i^r \in V'$ without fear of assigning 1 to both a literal and its complement, since G contains no edges between inconsistent literals.
- Each clause is satisfied, and Φ so is satisfied

Converse is also True

- A satisfying assignment of Φ has $x_2 = 0$ and $x_3 = 1$
- A corresponding clique of size $k = 3$ consists of the vertices corresponding to $!x_2$ from the first clause, x_3 from the second clause, and x_3 from the third clause
- Because the clique contains no vertices corresponding to either x_1 or $\neg x_1$
- we can set x_1 to either 0 or 1 in this satisfying assignment

The vertex-cover problem

- Vertex cover of an undirected graph G = (V, E) is a subset V ' ⊆ V such that if (u, v) ∈ E, then u ∈ V' or v ∈ V' (or both)
- That is, each vertex "covers" its incident edges, and a vertex cover for G is a set of vertices that covers all the edges in E
- size of a vertex cover is the number of vertices in it
- For example, graph in Figure 34.15(b) has a vertex cover {w, z} of size 2

The vertex–cover problem

- Vertex–cover problem is to find a vertex cover of minimum size in a given graph
- Restating this optimization problem as a decision problem, we wish to determine whether a graph has a vertex cover of a given size k
- As a language, we define VERTEX–COVER = {<G, k> : graph G has a vertex cover of size k}

Vertex–cover problem is NP–complete

- VERTEX–COVER $\in$ NP
- We are given a graph $G = (V, E)$ and an integer k
- Certificate we choose is the vertex cover $V' \subseteq V$ itself
- The verification algorithm affirms that $|V'| = k$ and then it checks, for each edge $(u, v) \in E$, that $u \in V'$ or $v \in V'$
- Can easily verify the certificate in polynomial time

Vertex–cover problem is NP–hard

- Showing that CLIQUE $\leq_P$ VERTEX–COVER
- This reduction relies on "complement" of a graph
- Given an undirected graph $G = (V, E)$, we define the complement of G as $G' = (V, E')$, where $E' = \{(u,v) : u, v \in V; u \neq v, \text{ and } (u, v) \notin E\}$
- In other words, G' is the graph containing exactly those edges that are not in G

The vertex–cover problem

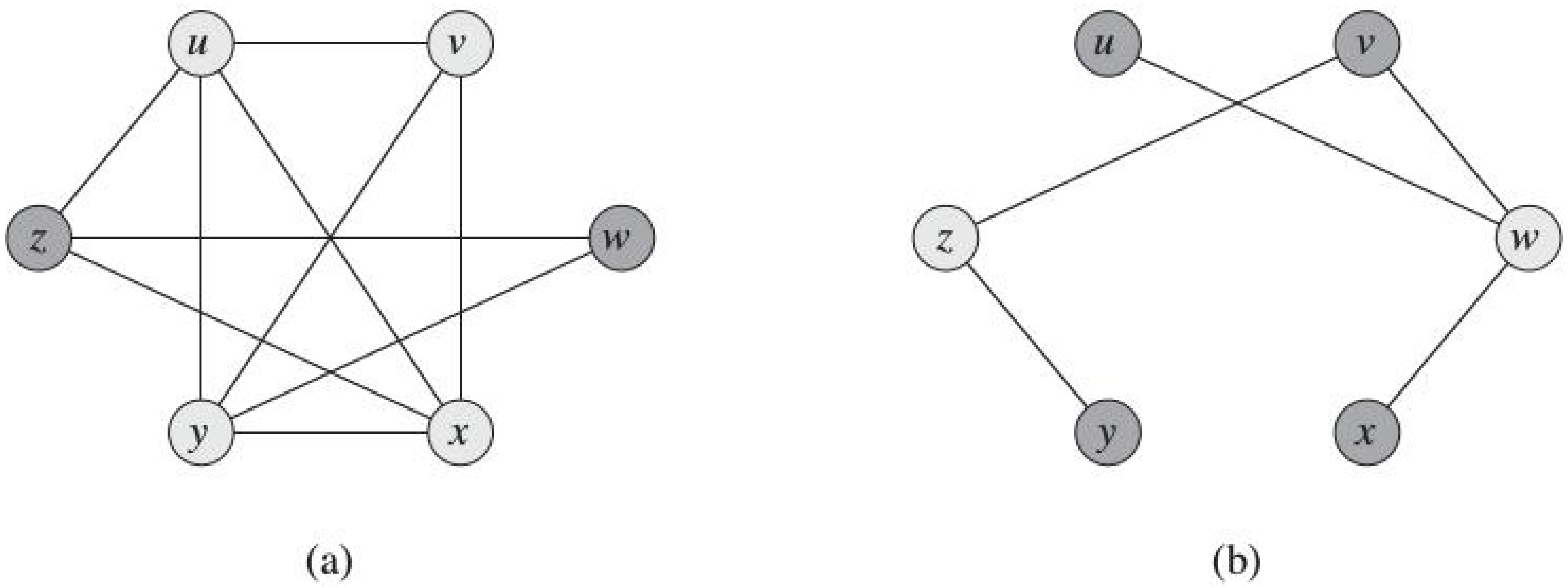
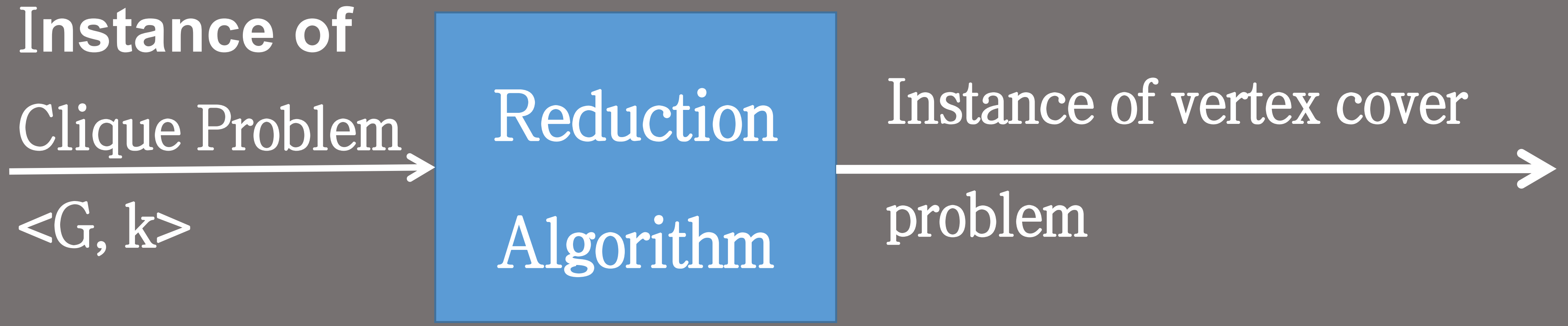


Figure 34.15 Reducing CLIQUE to VERTEX-COVER. **(a)** An undirected graph $G = (V, E)$ with clique $V' = \{u, v, x, y\}$. **(b)** The graph $\overline{G}$ produced by the reduction algorithm that has vertex cover $V - V' = \{w, z\}$.

Reducing Clique to Vertex cover

- Reduction algorithm takes as input an instance <G, k> of the clique problem
- It computes complement G', which we can easily do in polynomial time



Vertex–cover problem is NP–hard

- Output of reduction algorithm is the instance <G', |V| – k> of vertex–cover problem
- To complete we show that this transformation is indeed a reduction
- the graph G has a clique of size k if and only if the graph G has a vertex cover of size |V| – k

Vertex–cover problem is NP–hard

If G has a clique $V' \subseteq V$ with $|V'| = k$ then $V - V'$ is a vertex cover in G'

- If $(u,v) \in E'$ –> $(u,v) \notin E$, which implies
- If $(u,v) \notin E$ then at least one of u or v does not belong to V'
- Since every pair of vertices in V' is connected by an edge of E
- Equivalently, at least one of u or v is in $V - V'$, which means that edge (u, v) is covered by $V - V'$

Vertex–cover problem is NP–hard

- Since (u, v) was chosen arbitrarily from E', every edge of E' is covered by a vertex in V – V'
- Hence, the set V – V', which has size |V| – k, forms a vertex cover for G

Vertex–cover problem is NP–hard

- Conversely, suppose that G has a vertex cover $V' \subseteq V$, where $|V'| = |V| - k$
- Then, for all $u, v \in V$, if $(u, v) \in E'$, then $u \in V'$ or $v \in V'$ or both
- Contrapositive of this implication is that for all $u, v \in V$, if $u \notin V'$ and $v \notin V'$, then $(u, v) \in E$.
- In other words, $V - V'$ is a clique, and it has size $|V| - |V'| = k$