

Chapter 8

NP and Computational Intractability



Slides by Kevin Wayne.
Copyright © 2005 Pearson-Addison Wesley.
All rights reserved.

Algorithm Design Patterns and Anti-Patterns

Algorithm design patterns.

- Greed.
- Divide-and-conquer.
- Dynamic programming.
- Duality.
- Reductions.
- Local search.
- Randomization.

Ex.

$O(n \log n)$ interval scheduling.

$O(n \log n)$ FFT.

$O(n^2)$ edit distance.

$O(n^3)$ bipartite matching.

Algorithm design anti-patterns.

- NP-completeness.
- PSPACE-completeness.
- Undecidability.

$O(n^k)$ algorithm unlikely.

$O(n^k)$ certification algorithm unlikely.

No algorithm possible.

8.1 Polynomial-Time Reductions

Classify Problems According to Computational Requirements

Q. Which problems will we be able to solve in practice?

A working definition. [Cobham 1964, Edmonds 1965, Rabin 1966]
Those with polynomial-time algorithms.

Yes	Probably no
Shortest path	Longest path
Matching	3D-matching
Min cut	Max cut
2-SAT	3-SAT
Planar 4-color	Planar 3-color
Bipartite vertex cover	Vertex cover
Primality testing	Factoring

Classify Problems

Desiderata. Classify problems according to those that can be solved in polynomial-time and those that cannot.

Provably requires exponential-time.

- Given a Turing machine, does it halt in at most k steps?
- Given a board position in an n -by- n generalization of chess, can black guarantee a win?

Frustrating news. Huge number of fundamental problems have defied classification for decades.

This chapter. Show that these fundamental problems are "computationally equivalent" and appear to be different manifestations of one **really hard** problem.

Polynomial-Time Reduction

Desiderata'. Suppose we could solve X in polynomial-time. What else could we solve in polynomial time?

don't confuse with reduces from

Reduction. Problem X **polynomially reduces to** problem Y if arbitrary instances of problem X can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem Y .



Notation. $X \leq_p Y$.

computational model supplemented by special piece of hardware that solves instances of Y in a single step

Remarks.

- We pay for time to write down instances sent to black box \Rightarrow instances of Y must be of polynomial size.
- Note: Turing reducibility.

in contrast to Karp reductions

Polynomial-Time Reduction

Purpose. Classify problems according to **relative** difficulty.

Design algorithms. If $X \leq_p Y$ and Y can be solved in polynomial-time, then X **can** also be solved in polynomial time.

Establish intractability. If $X \leq_p Y$ and X cannot be solved in polynomial-time, then Y **cannot** be solved in polynomial time.

Establish equivalence. If $X \leq_p Y$ and $Y \leq_p X$, we use notation $X \equiv_p Y$.

↑
up to cost of reduction

Reduction By Simple Equivalence

Basic reduction strategies.

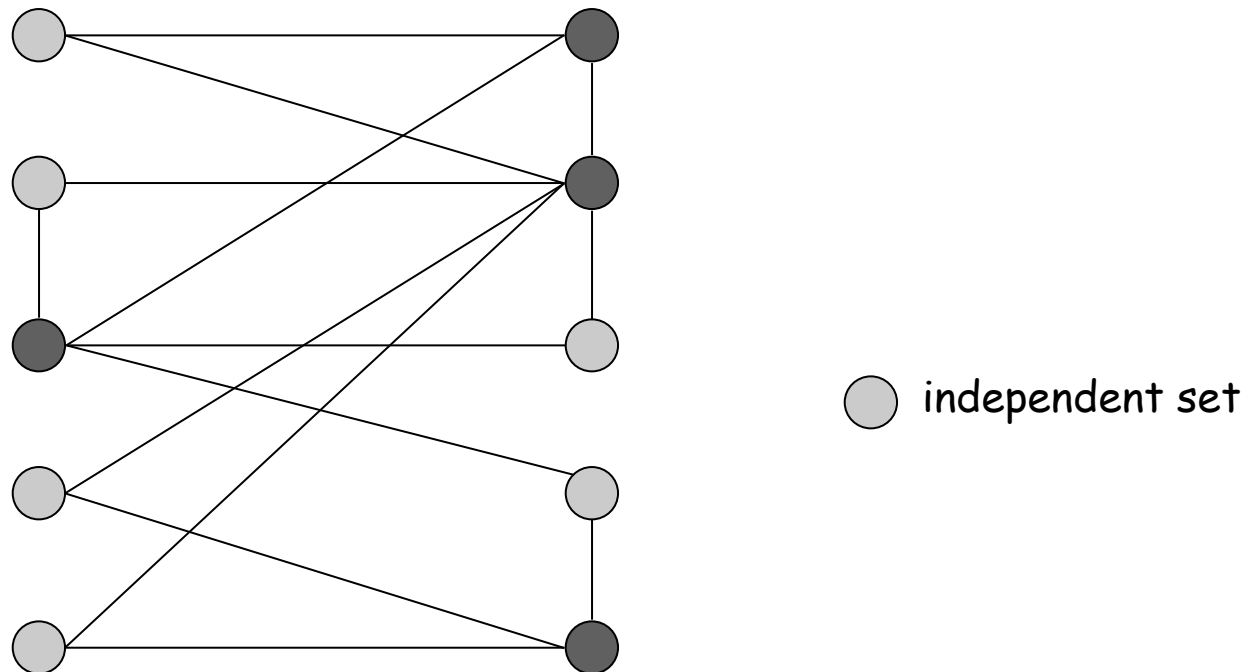
- **Reduction by simple equivalence.**
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Independent Set

INDEPENDENT SET: Given a graph $G = (V, E)$ and an integer k , is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in S ?

Ex. Is there an independent set of size ≥ 6 ? Yes.

Ex. Is there an independent set of size ≥ 7 ? No.

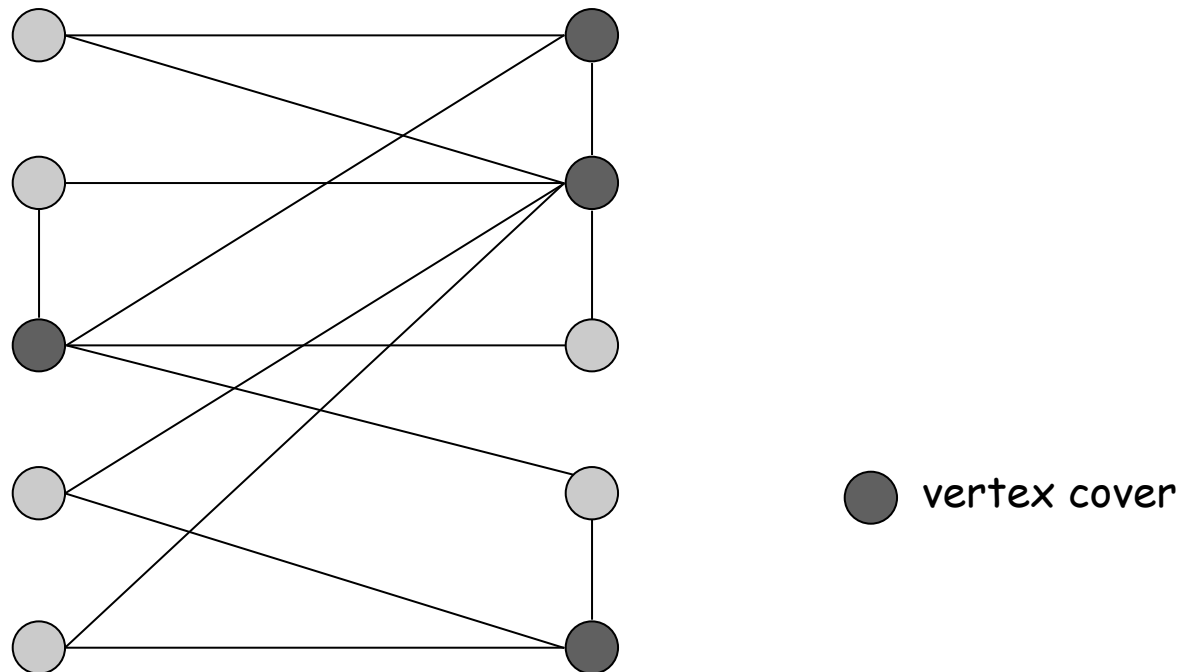


Vertex Cover

VERTEX COVER: Given a graph $G = (V, E)$ and an integer k , is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, at least one of its endpoints is in S ?

Ex. Is there a vertex cover of size ≤ 4 ? Yes.

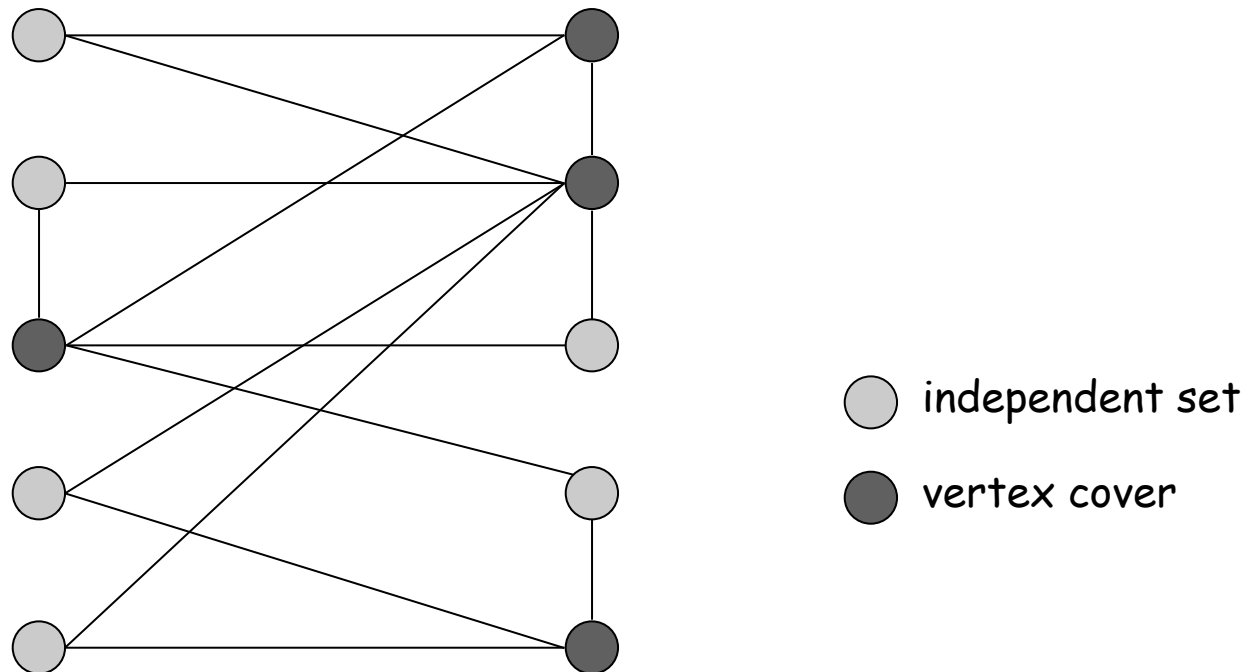
Ex. Is there a vertex cover of size ≤ 3 ? No.



Vertex Cover and Independent Set

Claim. VERTEX-COVER \equiv_p INDEPENDENT-SET.

Pf. We show S is an independent set iff $V - S$ is a vertex cover.



Vertex Cover and Independent Set

Claim. VERTEX-COVER \equiv_p INDEPENDENT-SET.

Pf. We show S is an independent set iff $V - S$ is a vertex cover.

\Rightarrow

- Let S be any independent set.
- Consider an arbitrary edge (u, v) .
- S independent $\Rightarrow u \notin S$ or $v \notin S \Rightarrow u \in V - S$ or $v \in V - S$.
- Thus, $V - S$ covers (u, v) .

\Leftarrow

- Let $V - S$ be any vertex cover.
- Consider two nodes $u \in S$ and $v \in S$.
- Observe that $(u, v) \notin E$ since $V - S$ is a vertex cover.
- Thus, no two nodes in S are joined by an edge $\Rightarrow S$ independent set. ■

Reduction from Special Case to General Case

Basic reduction strategies.

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Set Cover

SET COVER: Given a set U of elements, a collection S_1, S_2, \dots, S_m of subsets of U , and an integer k , does there exist a collection of $\leq k$ of these sets whose union is equal to U ?

Sample application.

- m available pieces of software.
- Set U of n capabilities that we would like our system to have.
- The i th piece of software provides the set $S_i \subseteq U$ of capabilities.
- Goal: achieve all n capabilities using fewest pieces of software.

Ex:

$$U = \{1, 2, 3, 4, 5, 6, 7\}$$

$$k = 2$$

$$S_1 = \{3, 7\}$$

$$S_4 = \{2, 4\}$$

$$S_2 = \{3, 4, 5, 6\}$$

$$S_5 = \{5\}$$

$$S_3 = \{1\}$$

$$S_6 = \{1, 2, 6, 7\}$$

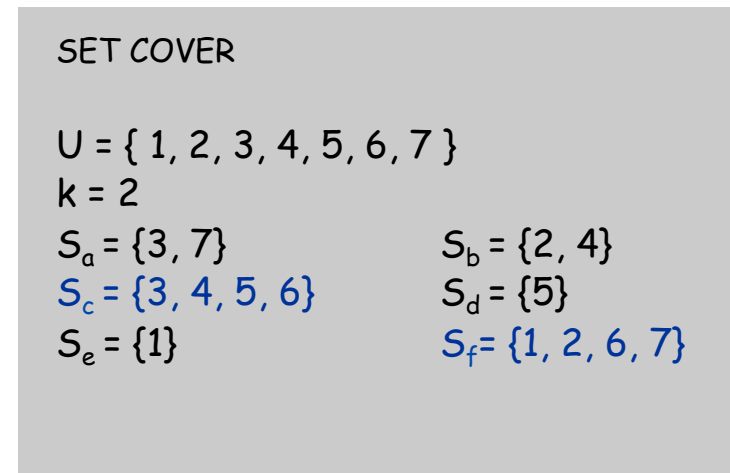
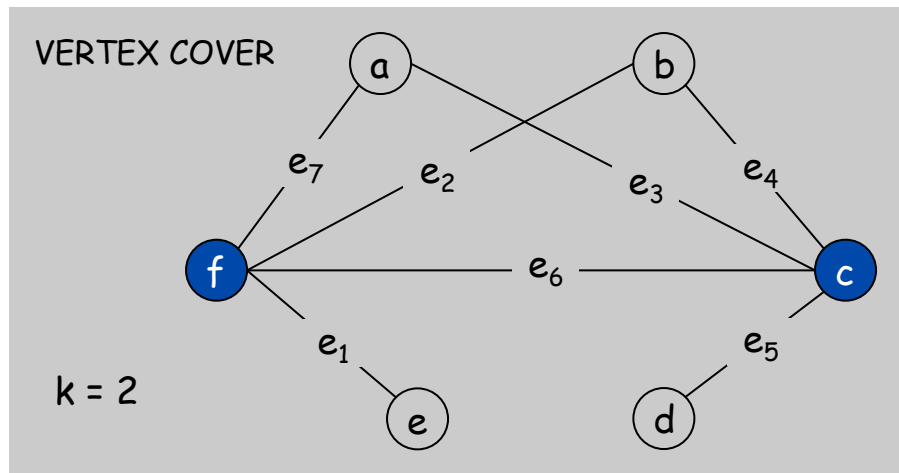
Vertex Cover Reduces to Set Cover

Claim. VERTEX-COVER \leq_p SET-COVER.

Pf. Given a VERTEX-COVER instance $G = (V, E)$, k , we construct a set cover instance whose size equals the size of the vertex cover instance.

Construction.

- Create SET-COVER instance:
 - $k = k$, $U = E$, $S_v = \{e \in E : e \text{ incident to } v\}$
- Set-cover of size $\leq k$ iff vertex cover of size $\leq k$. ▪



8.2 Reductions via "Gadgets"

Basic reduction strategies.

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction via "gadgets."

Satisfiability

Literal: A Boolean variable or its negation.

$$x_i \text{ or } \overline{x_i}$$

Clause: A disjunction of literals.

$$C_j = x_1 \vee \overline{x_2} \vee x_3$$

Conjunctive normal form: A propositional formula Φ that is the conjunction of clauses.

$$\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$$

SAT: Given CNF formula Φ , does it have a satisfying truth assignment?

3-SAT: SAT where each clause contains exactly 3 literals.

↑
each corresponds to a different variable

$$\text{Ex: } (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$

Yes: $x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{false}.$

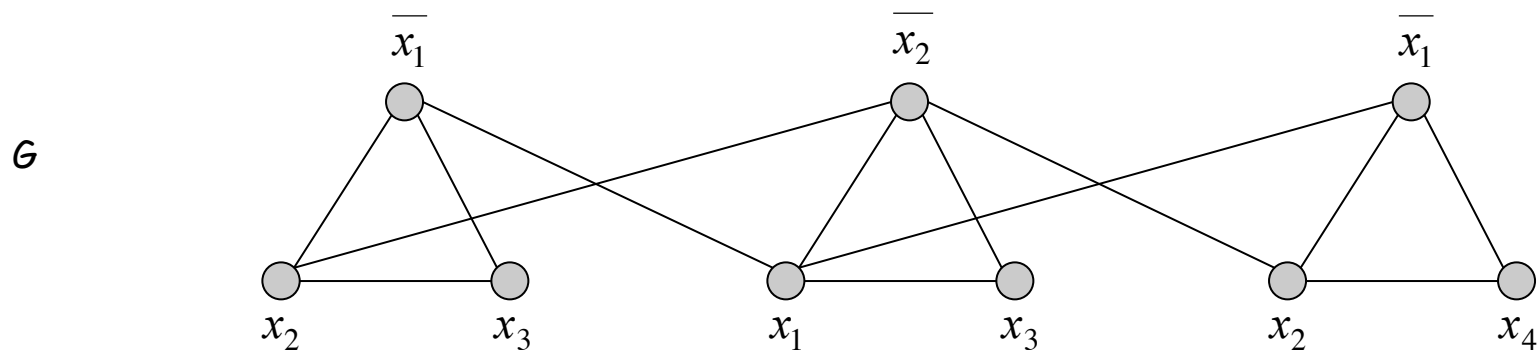
3 Satisfiability Reduces to Independent Set

Claim. $3\text{-SAT} \leq_p \text{INDEPENDENT-SET}$.

Pf. Given an instance Φ of 3-SAT, we construct an instance (G, k) of INDEPENDENT-SET that has an independent set of size k iff Φ is satisfiable.

Construction.

- G contains 3 vertices for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect literal to each of its negations.



$k = 3$

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

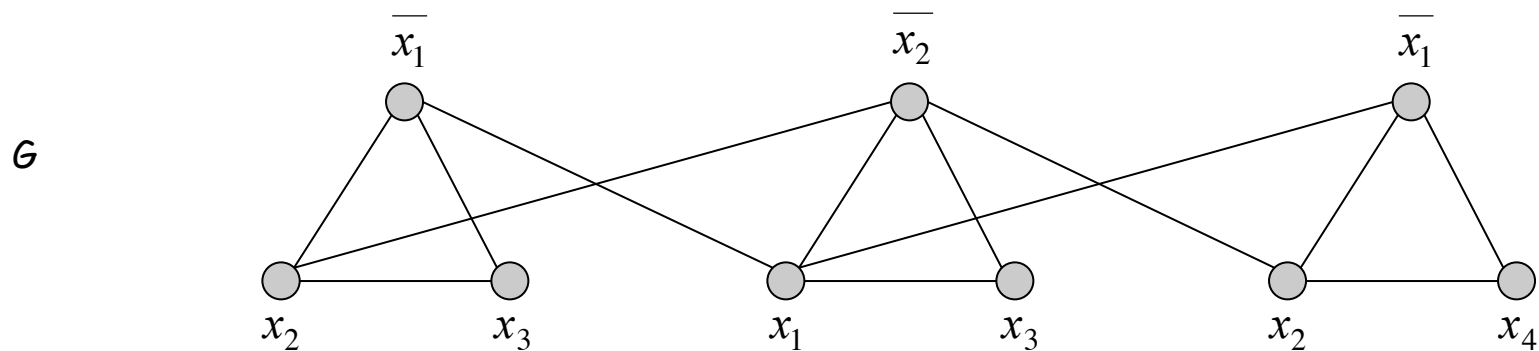
3 Satisfiability Reduces to Independent Set

Claim. G contains independent set of size $k = |\Phi|$ iff Φ is satisfiable.

Pf. \Rightarrow Let S be independent set of size k .

- S must contain exactly one vertex in each triangle.
- Set these literals to true. \leftarrow and any other variables in a consistent way
- Truth assignment is consistent and all clauses are satisfied.

Pf \Leftarrow Given satisfying assignment, select one true literal from each triangle. This is an independent set of size k . ■



$k = 3$

$$\Phi = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4)$$

Review

Basic reduction strategies.

- Simple equivalence: $\text{INDEPENDENT-SET} \equiv_p \text{VERTEX-COVER}$.
- Special case to general case: $\text{VERTEX-COVER} \leq_p \text{SET-COVER}$.
- Encoding with gadgets: $3\text{-SAT} \leq_p \text{INDEPENDENT-SET}$.

Transitivity. If $X \leq_p Y$ and $Y \leq_p Z$, then $X \leq_p Z$.

Pf idea. Compose the two algorithms.

Ex: $3\text{-SAT} \leq_p \text{INDEPENDENT-SET} \leq_p \text{VERTEX-COVER} \leq_p \text{SET-COVER}$.

Self-Reducibility

Decision problem. Does there **exist** a vertex cover of size $\leq k$?

Search problem. **Find** vertex cover of minimum cardinality.

Self-reducibility. Search problem \leq_p decision version.

- Applies to all (NP-complete) problems in this chapter.
- Justifies our focus on decision problems.

Ex: to find min cardinality vertex cover.

- (Binary) search for cardinality k^* of min vertex cover.
- Find a vertex v such that $G - \{v\}$ has a vertex cover of size $\leq k^* - 1$.
 - any vertex in any min vertex cover will have this property
- Include v in the vertex cover.
- Recursively find a min vertex cover in $G - \{v\}$.

↑
delete v and all incident edges

Self-Reducibility

Decision problem. Does there **exist** a vertex cover of size $\leq k$?

Search problem. **Find** vertex cover of minimum cardinality.

Self-reducibility. Search problem \leq_p decision version.

- Applies to all (NP-complete) problems in this chapter.
- Justifies our focus on decision problems.

Ex: to find min cardinality vertex cover.

- (Binary) search for cardinality k^* of min vertex cover.
- Find a vertex v such that $G - \{v\}$ has a vertex cover of size $\leq k^* - 1$.
 - any vertex in any min vertex cover will have this property
- Include v in the vertex cover.
- Recursively find a min vertex cover in $G - \{v\}$.

↑
delete v and all incident edges