Topic 8:
Approximation Algorithms

Outline

- Overview
- The Vertex-cover problem



Overview



NP-Complete Problems in Practice

- NP-complete problems are too important to abandon.
- Even if a problem is NP-complete, there may be hope.
- We have at least three ways to get around NP-completeness.
  - 1. If the actual inputs are small, an algorithm with exponential running time may be perfectly satisfactory.
  - 2. We may be able to isolate important special cases that we can solve in polynomial time.
  - 3. we might come up with approaches to find *near-optimal* solutions in polynomial time (either in the worst case or the expected case).
    - We call an algorithm that returns near-optimal solutions an ***approximation algorithm***.



Performance Ratios

- An algorithm for a problem has an ***approximation ratio*** of *ρ(n)* if, for any input of size *n*, the cost ***C*** of the solution produced by the algorithm is within a factor of *ρ(n)* of the cost ***C**** of an optimal solution:

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \le \rho(n)$$

- If an algorithm achieves an approximation ratio of ρ(n), we call it a ***ρ(n)-approximation algorithm***.
  - For a maximization problem, $0<C\le C^*$ and ratio = C^*/C
  - For a minimization problem, $0<C^*\le C$ and ratio = C/C^*
- The approximation ratio of an approximation algorithm is *never less than 1*.



Approximation Scheme

- An ***approximation scheme*** for an optimization problem:
  - An approximation algorithm that takes as input not only an instance of the problem, but also a value $\varepsilon > 0$ such that for any fixed ε, the scheme is a ***(1+ε)-approximation algorithm***.
    - ***Polynomial-time approximation scheme (PTAS)***
      - If for any fixed $\varepsilon > 0$, the scheme runs in time polynomial in the size n of its input instance. E.g., $O(n^{1/\varepsilon})$, $O(n^{2/\varepsilon})$
    - ***Fully Polynomial-time approximation scheme (FPTAS)***
      - If the scheme is an approximation scheme and its running time is polynomial in both $1/\varepsilon$ and the size n of the input instance. E.g., $O((1/e)n)$, $O(1/\varepsilon)^2 n^3)$.
      - Any constant-factor decrease in ε comes with a corresponding constant-factor increase in the running time.



The Vertex-Cover Problem



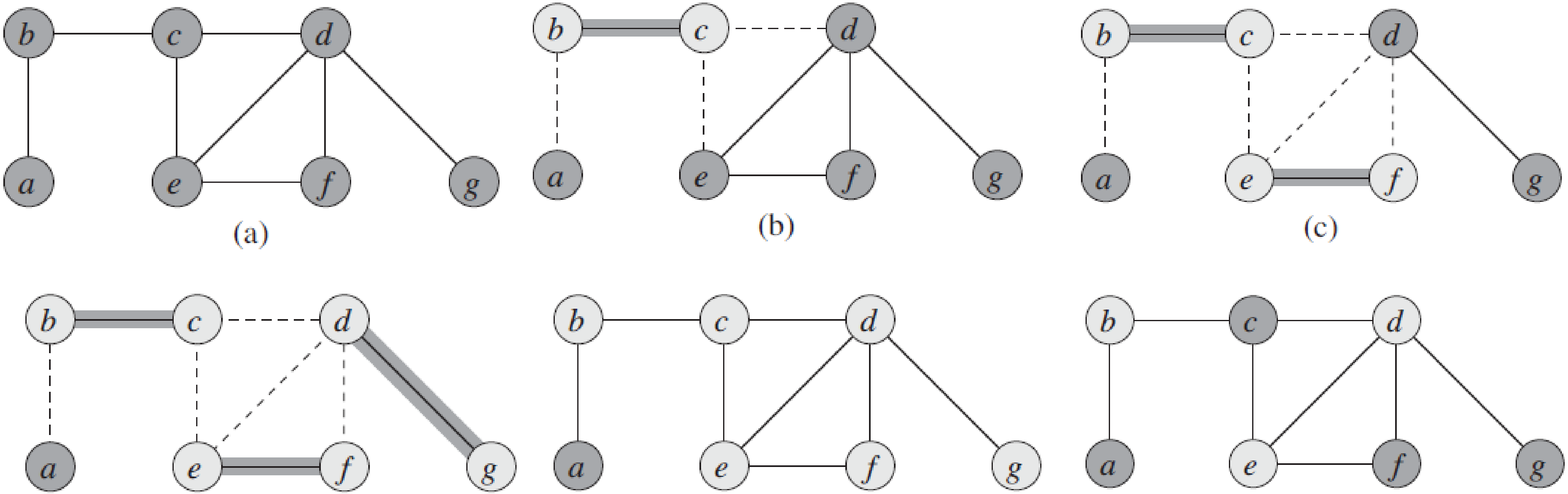
Vertex-Cover Problem

- A ***vertex cover*** of an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ such that if (u, v) is an edge of G, then either $u \in V'$ or $v \in V'$ (or both).
- The ***size*** of a vertex cover is the number of vertices in it.
- The ***vertex-cover problem*** is to find a vertex cover of minimum size in a given undirected graph. We call such a vertex cover an ***optimal vertex cover***.
- This problem is the ***optimization version*** of an NP-complete decision problem.

A 2-Approximation Algorithm

Using *adjacency lists* to represent E', the running time of this algorithm is O(V+E).

```
APPROX-VERTEX-COVER(G)
C = ∅
E' = G.E
while E' ≠ ∅
    let (u, v) be an arbitrary edge of E'
    C = C ∪ {u, v}
    remove from E' every edge incident on either u or v
return C
```



(a) (b) (c)

(d) (e) (f)



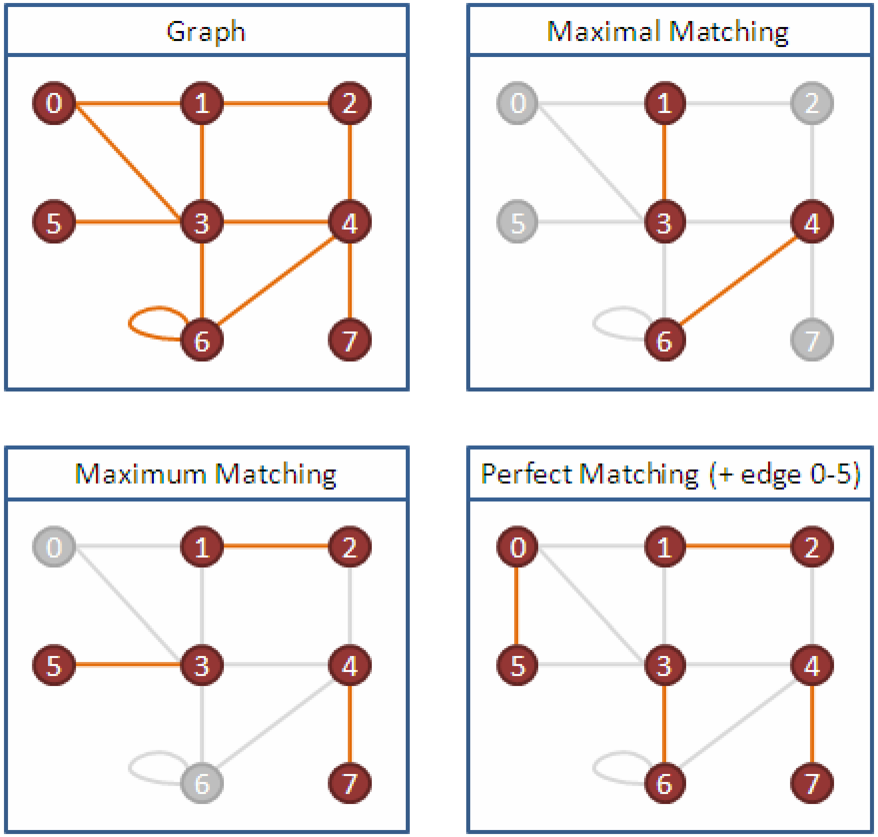
A 2-Approximation Algorithm (Cont.)

```
APPROX-VERTEX-COVER(G)
1  C = ∅
2  E' = G.E
3  while E' ≠ ∅
4      let (u, v) be an arbitrary edge of E'
5      C = C ∪ {u, v}
6      remove from E' every edge incident on either u or v
7  return C
```

- ***Theorem 35.1***
  - APPROX-VERTEX-COVER is a polynomial-time 2-approximation algorithm.
- ***Proof***
  - APPROX-VERTEX-COVER runs in polynomial time: O(V+E)
  - Let A denote the set of edges that line 4 of APPROX-VERTEX-COVER picked.
    - In order to cover the edges in A, any vertex cover (including an optimal cover ***C****) must include at least one endpoint of each edge in A.
    - No two edges in A share an endpoint, since once an edge is picked in line 4, all other edges that are incident on its endpoints are deleted from E' in line 6. → $|C^*| \geq |A|$
  - Each execution of line 4 picks an edge for which neither of its endpoints is already in C. → $|C| = 2|A|$
  - Therefore, $|C| = 2|A| \leq 2|C^*|$

A 2-Approximation Algorithm (Cont.)

- How to prove without knowing the optimal vertex cover?
  →Rely on the ***lower bound***.
- The set ***A*** of edges is a a *maximal matching* that is a *lower bound* on the size of an optimal vertex cover.



maximal matching: a matching that is not a proper subset of any other matching
maximum matching：A matching that contains the largest possible number of edges.
Also a maximal matching。
perfect matching：A matching that matches all vertices of the graph. Also a maximum matching