CSE408
Vertex Cover and Bin Packing

Lecture # 37

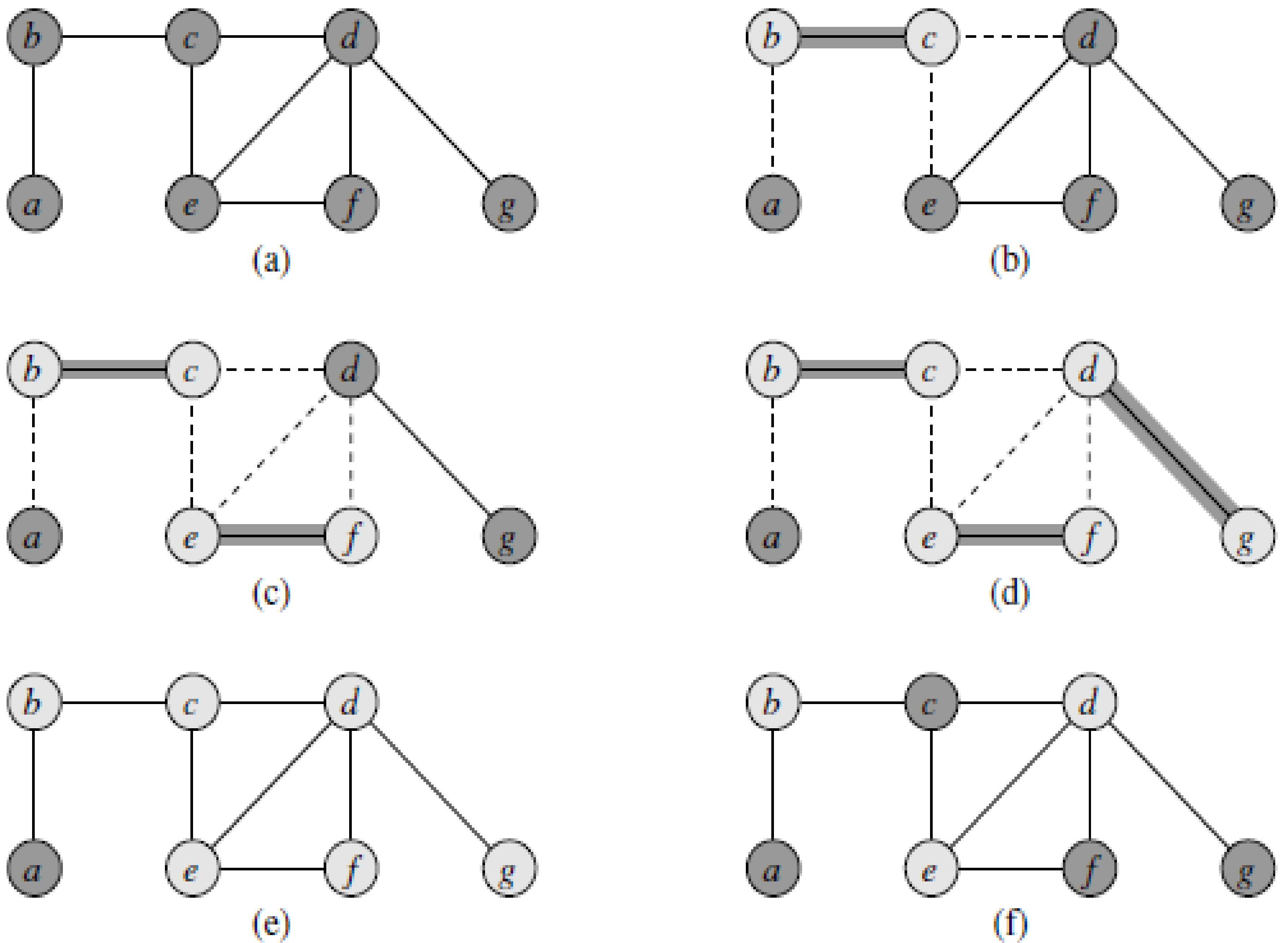
The vertex-cover problem

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.

```
APPROX-VERTEX-COVER(G)

C ← Ø
E′ ← E[G]
while E′ ≠ Ø
      do 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
```

Example



The set cover problem

An instance $(X, \mathcal{F})$ of the *set-covering problem* consists of a finite set X and a family $\mathcal{F}$ of subsets of X, such that every element of X belongs to at least one subset in $\mathcal{F}$:

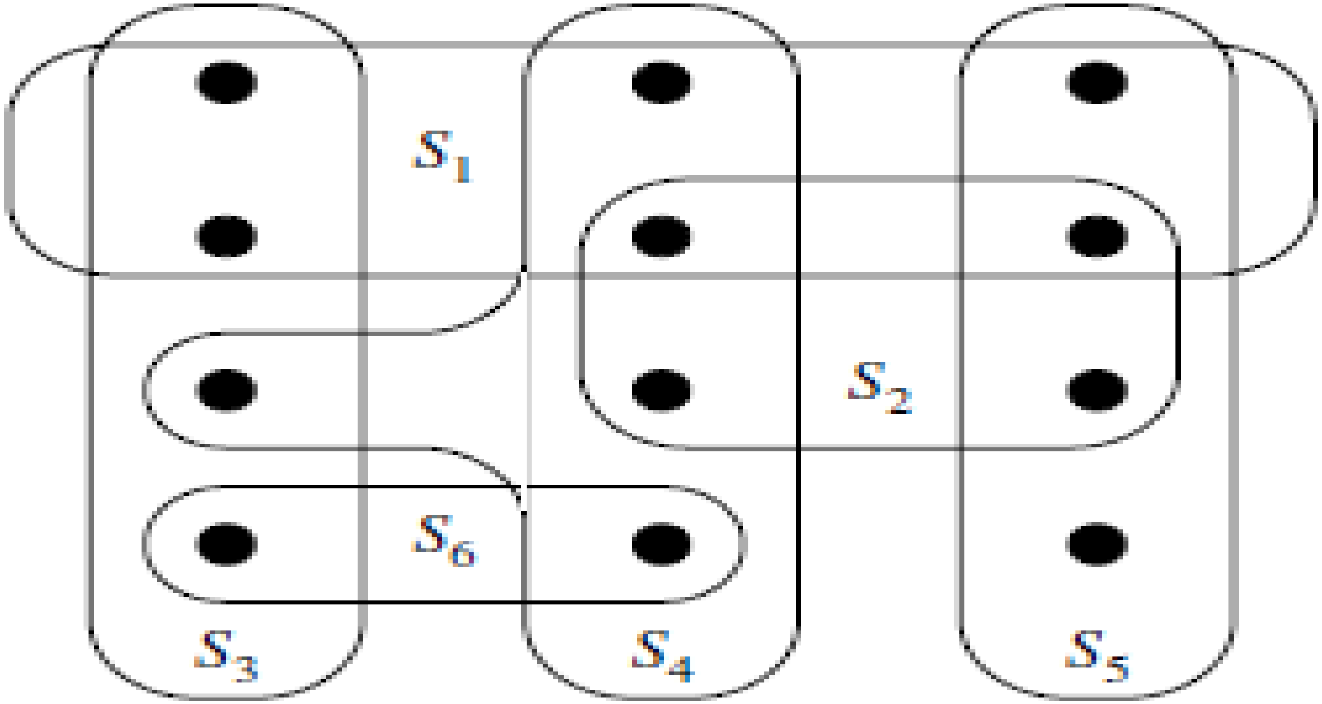


Figure 35.3 An instance $(X, \mathcal{F})$ of the set-covering problem, where X consists of the 12 black points and $\mathcal{F} = \{S_1, S_2, S_3, S_4, S_5, S_6\}$. A minimum-size set cover is $\mathcal{C} = \{S_3, S_4, S_5\}$. The greedy algorithm produces a cover of size 4 by selecting the sets S_1, S_4, S_5, and S_3 in order.

A greedy approximation algorithm

The greedy method works by picking, at each stage, the set S that covers the greatest number of remaining elements that are uncovered.

Set cover algo

GREEDY-SET-COVER$(X, \mathcal{F})$

```
U ← X
𝒞 ← ∅
while U ≠ ∅
      do select an S ∈ ℱ that maximizes |S ∩ U|
            U ← U − S
            𝒞 ← 𝒞 ∪ {S}
return 𝒞
```

Bin Packing

- In the **bin packing problem**, objects of different volumes must be packed into a finite number of bins or containers each of volume *V* in a way that minimizes the number of bins used. In computational complexity theory, it is a combinatorial NP-hard problem.
- There are many variations of this problem, such as 2D packing, linear packing, packing by weight, packing by cost, and so on.
- They have many applications, such as filling up containers, loading trucks with weight capacity constraints, creating file backups in removable media and technology mapping in Field-programmable gate array semiconductor chip design.

- The bin packing problem can also be seen as a special case of the cutting stock problem.
- When the number of bins is restricted to 1 and each item is characterised by both a volume and a value, the problem of maximising the value of items that can fit in the bin is known as the knapsack problem.
- Despite the fact that the bin packing problem has an NP-hard computational complexity, optimal solutions to very large instances of the problem can be produced with sophisticated algorithms.
- In addition, many heuristics have been developed: for example, the **first fit algorithm** provides a fast but often non-optimal solution, involving placing each item into the first bin in which it will fit.
- It requires $\Theta(n \log n)$ time, where n is the number of elements to be packed.

Bin packing

- The algorithm can be made much more effective by first sorting the list of elements into decreasing order (sometimes known as the first-fit decreasing algorithm), although this still does not guarantee an optimal solution, and for longer lists may increase the running time of the algorithm.
- It is known, however, that there always exists at least one ordering of items that allows first-fit to produce an optimal solution.[1]
- An interesting variant of bin packing that occurs in practice is when items can share space when packed into a bin.
- Specifically, a set of items could occupy less space when packed together than the sum of their individual sizes.

- This variant is known as VM packing[2] since when virtual machines (VMs) are packed in a server, their total memory requirement could decrease due to pages shared by the VMs that need only be stored once.
- If items can share space in arbitrary ways, the bin packing problem is hard to even approximate.
- However, if the space sharing fits into a hierarchy, as is the case with memory sharing in virtual machines, the bin packing problem can be efficiently approximated.

Thank You !!!