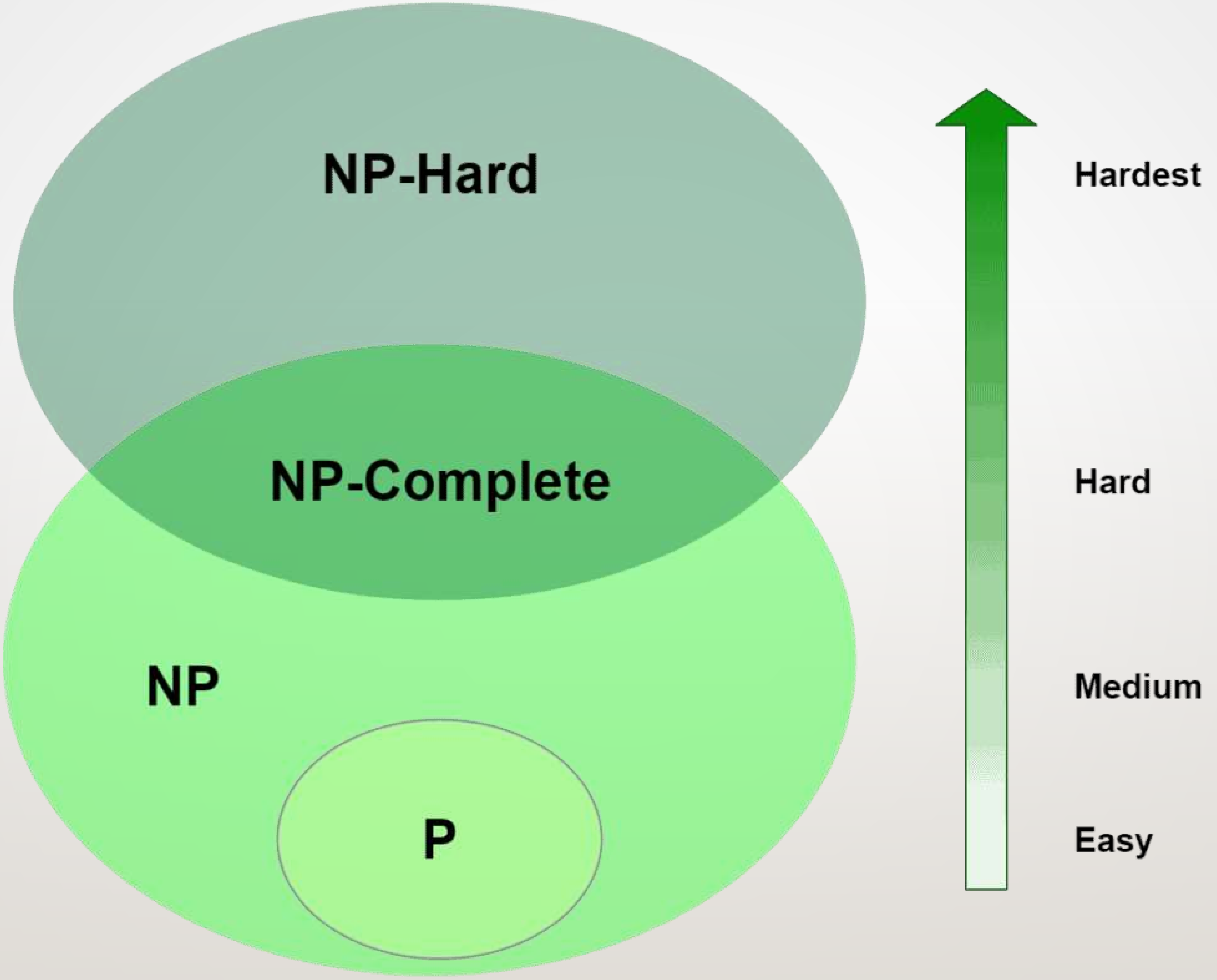
Approximation Algorithms


NP-Hard
NP-Complete
NP
P
Hardest
Hard
Medium
Easy

Approximation Algorithms

Optimization Problem

1. In mathematics and computer science, an **optimization problem is the problem of finding the best solution from all feasible solutions**.

2. The objective may be **either min. or max.** depending on the problem considered.

3. A **large number of optimization problems** which are required to be solved in practice are **NP-hard**.

4. For such problems, it is not possible to design algorithms that can find exactly **optimal solution** to all instances of the problem in polynomial time in the size of the input, unless P = NP

Approximation Algorithms

Overview :

✓An approximation algorithm is a way of dealing with NP-completeness for an optimization problem.

✓This technique does not guarantee the best solution.

✓The goal of the approximation algorithm is to come as close as possible to the optimal solution in polynomial time Such algorithms are called APPROXIMATION ALGORITHMS or HEURISTIC ALGORITHMS.

•Features of Approximation Algorithm :

The features of the Approximation Algorithm as follows.

- An approximation algorithm guarantees to run in polynomial time though it does not guarantee the most effective solution.
- An approximation algorithm guarantees to seek out high accuracy and top quality solution(say within 1% of optimum)
- An Approximation algorithms are used to get an answer near the (optimal) solution of an optimization problem in polynomial time

Approximation Algorithms

•Performance Ratios for approximation algorithms :

The performance ratios of the Approximation Algorithm as follows:

Scenario-1 :

•Suppose that we are working on an **optimization problem** in which each potential solution has a cost, and we wish to find a near-optimal solution. Depending on the problem, we may define an **optimal solution** as one with **maximum possible cost or one with minimum possible cost**,i.e, the problem can either be a maximization or minimization problem.

•We say that an algorithm for a problem has an approximation ratio of P(n) if, for any **input** size **n**, the **cost C** of the solution produced by the algorithm is within a factor of P(n) of the **cost C*** of an optimal solution as follows.

$$\max(C/C^*, C^*/C) \le P(n)$$

Approximation Algorithms

Formula = max(C/C*,C*/C)<=P(n)

- **C**: Represents the cost or solution provided by the approximation algorithm.
- **C***: Represents the cost or solution of the optimal algorithm.
- **C/C***: The ratio of the approximation cost to the optimal cost, showing how close the approximation is to the optimal.
- **C*/C**: The ratio of the optimal cost to the approximation cost, also a measure of comparison.
- **max(C/C*, C*/C)**: This takes the maximum of the two ratios to ensure a worst-case measure of performance.
- **P(n)**: A polynomial function that depends on the input size n, representing the allowed approximation ratio or error bound that the algorithm can achieve. **(P(n) >= 1)**

•Thus, the inequality states that the maximum between these ratios should be less than or equal to a certain function P(n), meaning the approximation algorithm performs within acceptable limits when compared to the optimal solution.

Approximation Algorithms

Scenario-2 :

If an algorithm reaches an approximation ratio of P(n), then we call it a P(n)-approximation algorithm.

•For a **maximization problem**, 0< C < C*, and the ratio of C*/C gives the factor by which the cost of an optimal solution is larger than the cost of the approximate algorithm.

•For a **minimization problem**, 0< C* < C, and the ratio of C/C* gives the factor by which the cost of an approximate solution is larger than the cost of an optimal solution.

Approximation Algorithms

•Some examples of the Approximation algorithm :

Here, we will discuss some examples of the Approximation Algorithm as follows.

•The Vertex Cover Problem –

In the vertex cover problem, the optimization problem is to find the vertex cover with the **fewest vertices**, and the approximation problem is to find the vertex cover with **few vertices**.

•Travelling Salesman Problem –

In the traveling salesperson problem, the optimization problem is to find the **shortest cycle**, and the approximation problem is to find a **short cycle**.

•The Set Covering Problem –

This is an optimization problem that models many problems that require resources to be allocated. Here, a logarithmic approximation ratio is used.

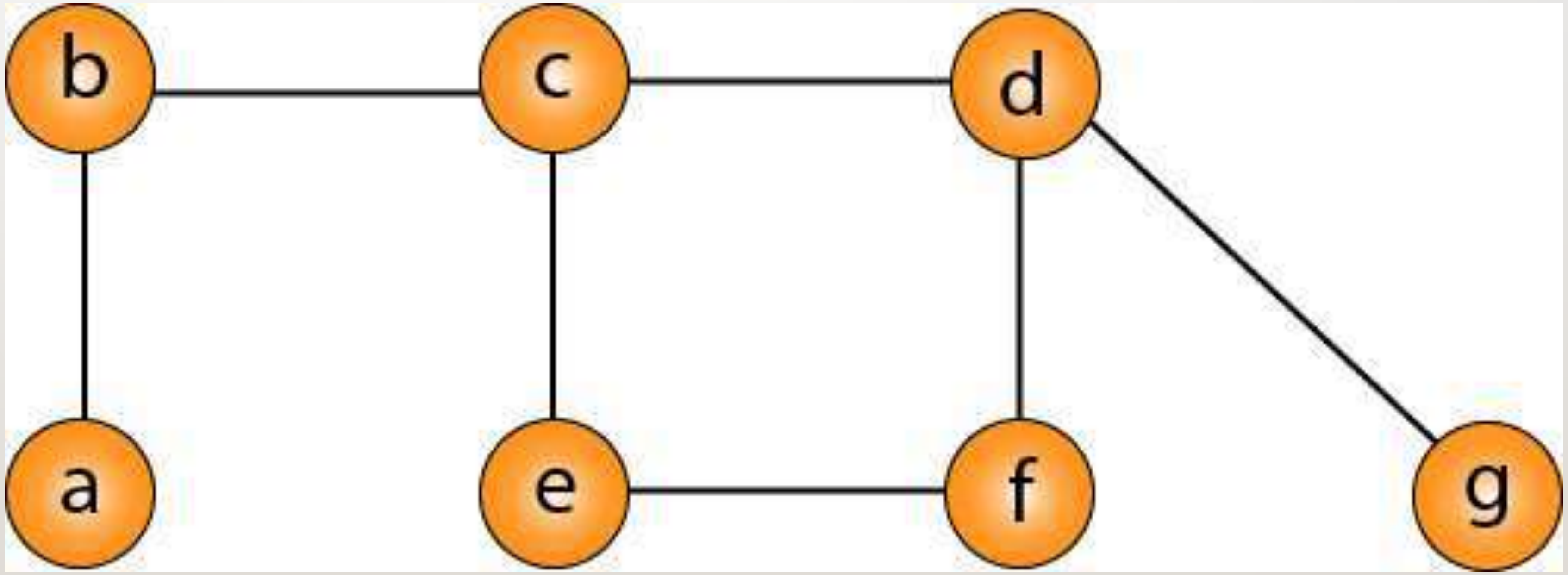
•The Subset Sum Problem–

In the Subset sum problem, the optimization problem is to find a subset of {x1,×2,×3…xn} whose sum is as large as possible but not larger than the target value t.

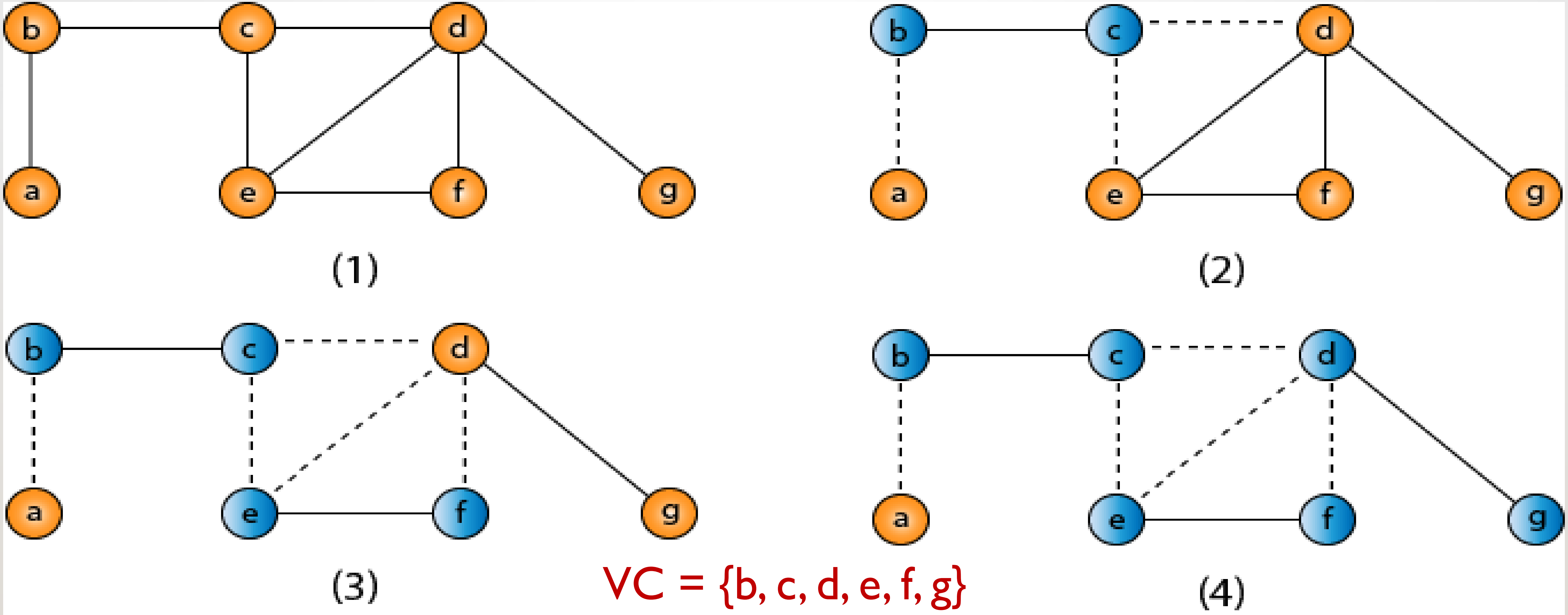
Vertex Cover Problem

A Vertex Cover of a graph G is a set of vertices such that each edge in G is incident to at least one of these vertices.

The decision vertex-cover problem was proven NPC. Now, we want to solve the optimal version of the vertex cover problem, i.e., we want to find a minimum size vertex cover of a given graph. We call such vertex cover an optimal vertex cover C*.



The idea is to take an edge (u, v) one by one, put both vertices to C, and remove all the edges incident to u or v. We carry on until all edges have been removed. C is a VC. But how good is C?



VC = {b, c, d, e, f, g}

```
Approx-Vertex-Cover (G = (V, E))
{
      C = empty-set;
    E'= E;
    While E' is not empty do
      {
    Let (u, v) be any edge in E': (*)
    Add u and v to C;
    Remove from E' all edges incident to
      u or v;
      }
    Return C;
}
```