

Approximate Algorithms

Introduction:

An Approximate Algorithm is a way of approach **NP-COMPLETENESS** for the optimization problem. This technique does not guarantee the best solution. The goal of an approximation algorithm is to come as close as possible to the optimum value in a reasonable amount of time which is at the most polynomial time. Such algorithms are called approximation algorithm or heuristic algorithm.

- For the traveling salesperson problem, the optimization problem is to find the shortest cycle, and the approximation problem is to find a short cycle.
- For the vertex cover problem, the optimization problem is to find the vertex cover with fewest vertices, and the approximation problem is to find the vertex cover with few vertices.

Performance Ratios

Suppose we work on an optimization problem where every solution carries a cost. An Approximate Algorithm returns a legal solution, but the cost of that legal solution may not be optimal.

For Example, suppose we are considering for a **minimum size vertex-cover (VC)**. An approximate algorithm returns a VC for us, but the size (cost) may not be minimized.

Another Example is we are considering for a **maximum size Independent set (IS)**. An approximate Algorithm returns an IS for us, but the size (cost) may not be maximum. Let C be the cost of the solution returned by an approximate algorithm, and C^* is the cost of the optimal solution.

We say the approximate algorithm has an approximate ratio $P(n)$ for an input size n , where

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq P(n)$$

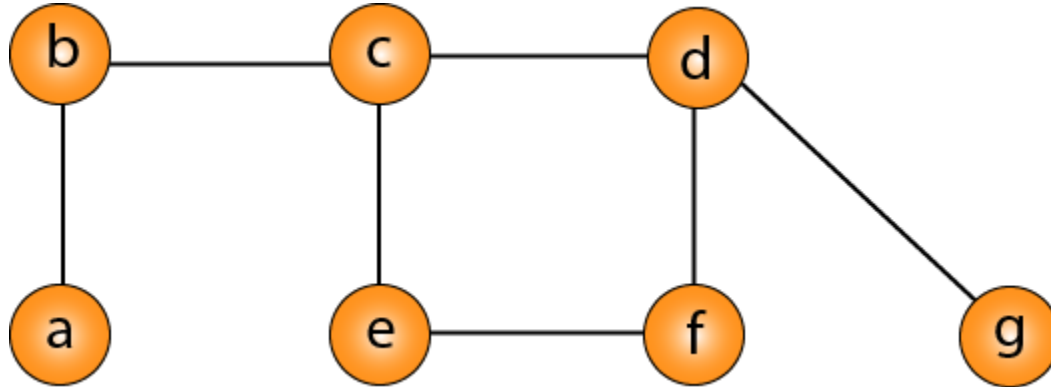
Intuitively, the approximation ratio measures how bad the approximate solution is distinguished with the optimal solution. A large (small) approximation ratio measures the solution is much worse than (more or less the same as) an optimal solution.

Observe that $P(n)$ is always ≥ 1 , if the ratio does not depend on n , we may write P . Therefore, a 1-approximation algorithm gives an optimal solution. Some problems have polynomial-time approximation algorithm with small constant approximate ratios, while others have best-known polynomial time approximation algorithms whose approximate ratios grow with n .

Vertex Cover

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^* .

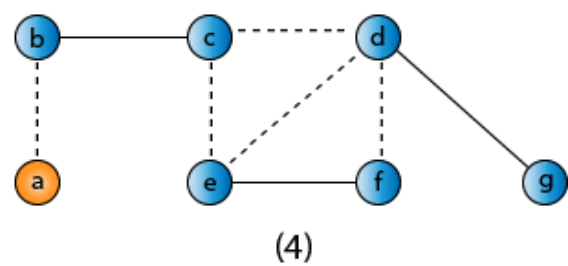
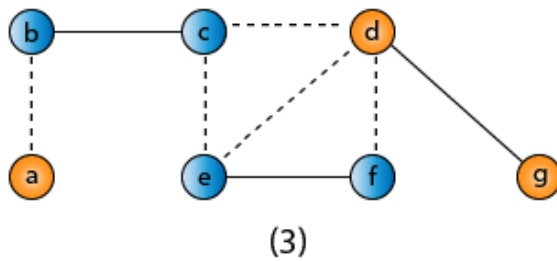
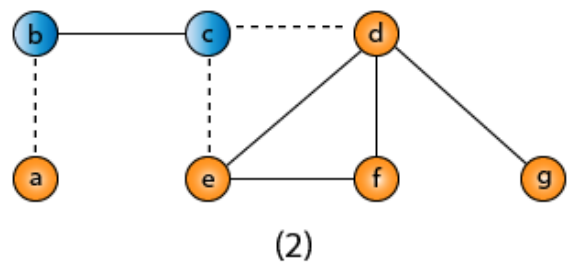
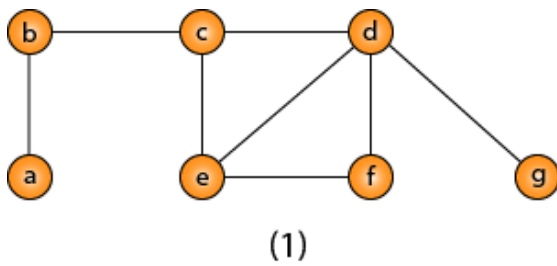


An approximate algorithm for vertex cover:

1. Approx-Vertex-Cover ($G = (V, E)$)
2. {
3. $C = \text{empty-set};$
4. $E' = E;$
5. While E' is not empty **do**
6. {
7. Let (u, v) be any edge in E' : (*)

8. Add u and v to C ;
9. Remove from E' all edges incident to
10. u or v ;
11. }
12. Return C ;
13. }

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 ?



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