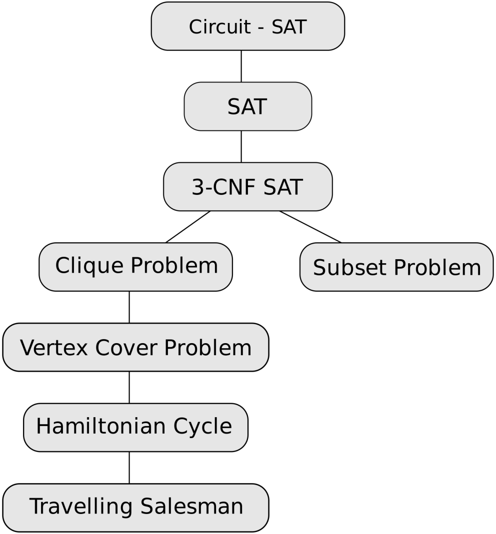
Approximation Algorithms

CPSC 6109 - Algorithms Analysis and Design

Dr. Hyrum D. Carroll

April 17, 2024

NP-Complete problems



Source: wikimedia.org user Actam

NP-Complete Problem Approximations

- Which NP-Complete problem is not important to have an efficient algorithm?

NP-Complete Problem Approximations

- Which NP-Complete problem is not important to have an efficient algorithm?
- 3 options to deal with an exponential run time:
  1. Just run the exponential algorithm (hopefully with small inputs)
  2. Develop a polynomial time algorithm for only certain inputs
  3. If instead of an optimal solution, a near-optimal solution will suffice, then use an approximation algorithm

Approximation Ratio $\rho(n)$

- Quantifies difference between optimal and approximation
- Assumption: Each solution has a positive cost
- C^*: Optimal Solution Cost
- C : Approximation Solution Cost
- Minimization $(0 < C^* \leq C)$

$$\frac{C}{C^*} \leq \rho(n) \tag{1}$$

Approximation Ratio $\rho(n)$

- Quantifies difference between optimal and approximation
- Assumption: Each solution has a positive cost
- C^*: Optimal Solution Cost
- C : Approximation Solution Cost
- Minimization ($0 < C^* \leq C$)

$$\frac{C}{C^*} \leq \rho(n) \tag{1}$$

- Maximization ($0 < C \leq C^*$)

$$\frac{C^*}{C} \leq \rho(n) \tag{2}$$

Approximation Ratio $\rho(n)$

- Quantifies difference between optimal and approximation
- Assumption: Each solution has a positive cost
- C^*: Optimal Solution Cost
- C : Approximation Solution Cost
- Minimization ($0 < C^* \leq C$) & Maximization ($0 < C \leq C^*$)

$$max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n) \tag{1}$$

Approximation Ratio $\rho(n)$

- Quantifies difference between optimal and approximation
- Assumption: Each solution has a positive cost
- C^*: Optimal Solution Cost
- C : Approximation Solution Cost
- Minimization ($0 < C^* \leq C$) & Maximization ($0 < C \leq C^*$)

$$max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n) \tag{1}$$

- $\rho(n) \geq 1$

Approximation Ratio $\rho(n)$

- Quantifies difference between optimal and approximation
- Assumption: Each solution has a positive cost
- C^*: Optimal Solution Cost
- C : Approximation Solution Cost
- Minimization $(0 < C^* \leq C)$ & Maximization $(0 < C \leq C^*)$

$$max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n) \tag{1}$$

- $\rho(n) \geq 1$
- $\rho(n)$-approximation algorithm

Approximation Schemes

- Given inputs of size n and a fixed ϵ
- $(1+\epsilon)$-approximation algorithm
- Generally, the run time increase significantly as ϵ decreases
- A polynomial-time approximation scheme runs in time polynomial to n for any fixed $\epsilon > 0$
- A fully polynomial-time approximation scheme runs in time polynomial to n for any fixed $1/\epsilon > 0$

Example Approximations Presentations

35.1 The vertex-cover problem
35.2 The traveling-salesman problem
35.3 The set-covering problem

Vertex Cover

- The Vertex Cover problem is determining a set of vertices such that every edge is adjacent to one of the vertices in the set

Vertex Cover

- The Vertex Cover problem is determining a set of vertices such that every edge is adjacent to one of the vertices in the set
- Formally, given an undirected graph $G = (V, E)$, the vertex cover is the subset $V' \subseteq V$, such that for every edge (u, v) in G, $u \in V'$ and/or $v \in V'$
- Optimal vertex-cover problem min $|C^*|$
- Approximate vertex cover solution: $|C| \leq 2|C^*|$, $\rho(n) = 2$

Approximate Vertex Cover Solution

```
Set vertexCoverApprox( Graph G ){
    Set vertexCover; // vertices in vertex cover
    Edges edgesCopy = G.Edges(); // copy all of the edges
    Edge edge;
    Vertex u;
    Vertex v;
    while( edgesCopy.isEmpty() == false ){
        edge = getEdge( edgesCopy ); // get any edge
        for( Vertex vertex : edge.getVertices() ){
            vertexCover.add( vertex ); // add to vertex cover
            for( Edge edge : vertex.getAdjacentEdges()){
                // remove edges already covered by vertex
                edgesCopy.remove( edge );
            }
        }
    }
    return vertexCover;
}
```

Approximate Vertex Cover Solution

- Using an adjacency list for E', runs in $O(V + E)$

Approximate Vertex Cover Solution

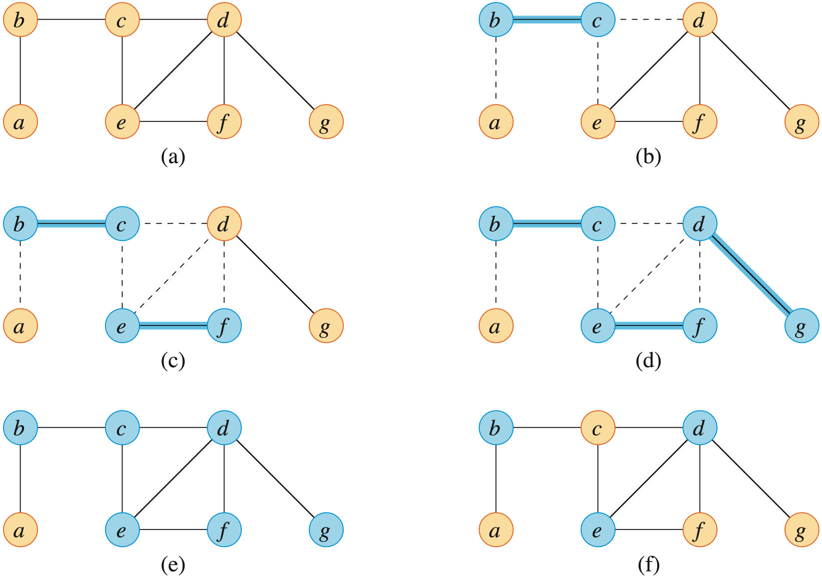


Figure 35.1 from Introduction to Algorithms 4^{th} Edition

Approximation Algorithms Exercise

- Extra credit for presenting about either:
  - Approximate Traveling-salesman Solution
  - Approximate Set-covering Solution
- Presentation needs to include:
  - Describe the problem (informally and formally)
  - Detail the approximate solution
  - Provide the run-time of the approximate solution
  - Provide $\rho(n)$

Approximate Traveling-salesman Solution

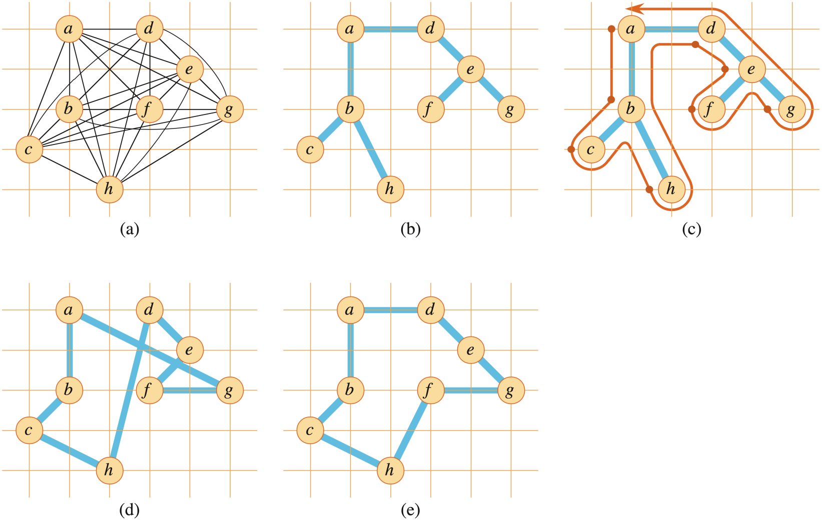


Figure 35.2 from Introduction to Algorithms 4^{th} Edition

Approximate Set-covering Solution

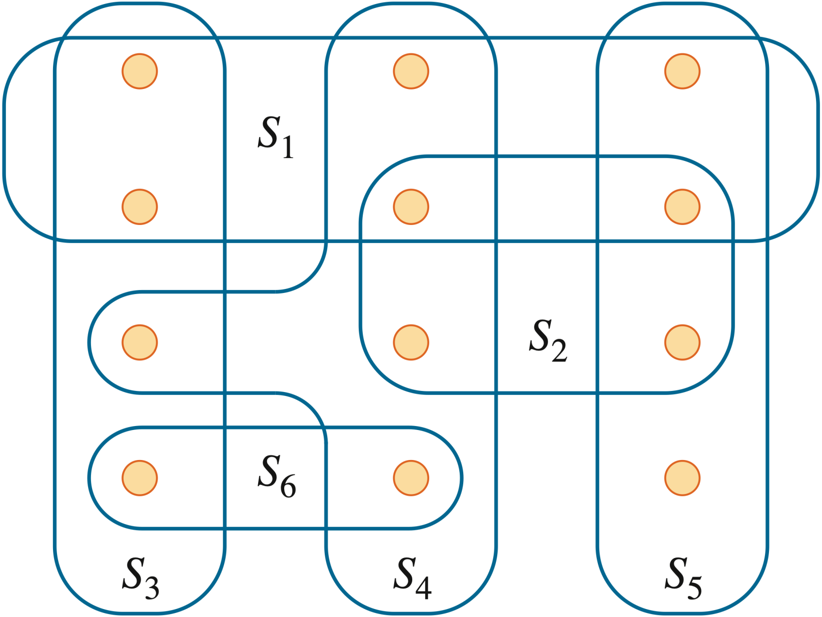


Figure 35.3 from Introduction to Algorithms 4^{th} Edition