NP-Complete Supplemental

CSCI 570

Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu

DISCUSSION 13-SUPPLEMENTAL

Outline

- Problems with Solutions

Problem 1

- Suppose we want to compute a spanning tree T of a given undirected graph G. However, we also have another parameter k; we want to constrain our output to be such that only spanning trees where each vertex v has degree ***at most*** k is considered.
- Prove that the problem of *Degree-Bounded Spanning Tree* is **NP**-Complete.

Solution 1

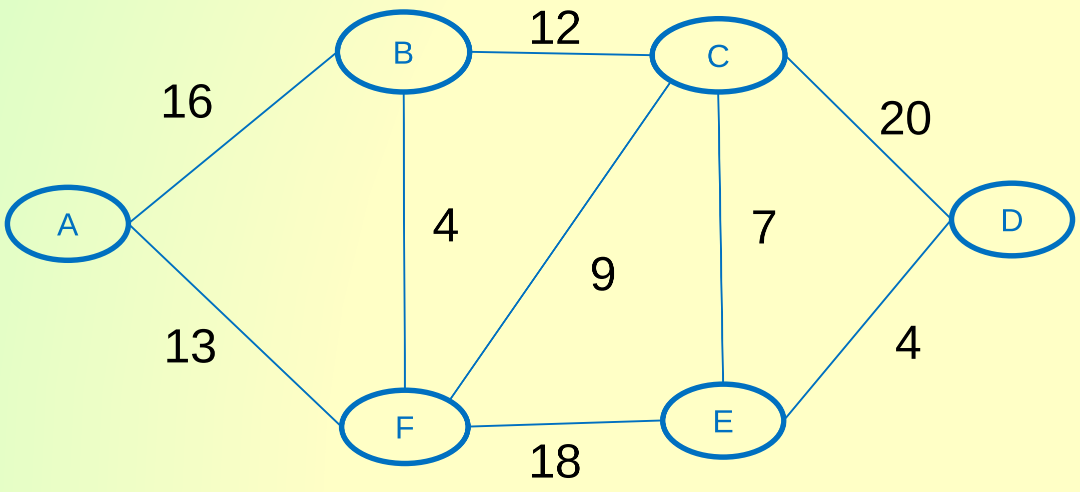
- To prove a problem is NP complete, you must prove two things:
  - The problem is in NP – a certificate can be verified in polynomial time
  - The problem is NP-Hard with respect to NPC problems – given a solution to this problem, convert a known NPC problem to this problem in polynomial time
- The prove the algorithm is in NP, the certificate would consist of the spanning tree
- We could check to see that each node in the spanning tree exists in the original tree – O(n)
- We could check to see that each node in the spanning tree has degree no more than k – O(n)
- We could check to see that the only edges used in the spanning tree exist in the original graph – O(m)

Solution 1

- To prove the problem is NP-hard with respect to another NPC algorithm, we could convert the Hamiltonian Path problem to the Degree-Bounded Spanning Tree
- A Hamiltonian Path is a path that includes every vertex in a graph only once
- This would be a Degree-Bounded Spanning Tree with k=2, which means the Degree-Bounded Spanning Tree is just a specific instance of the Hamiltonian Path problem

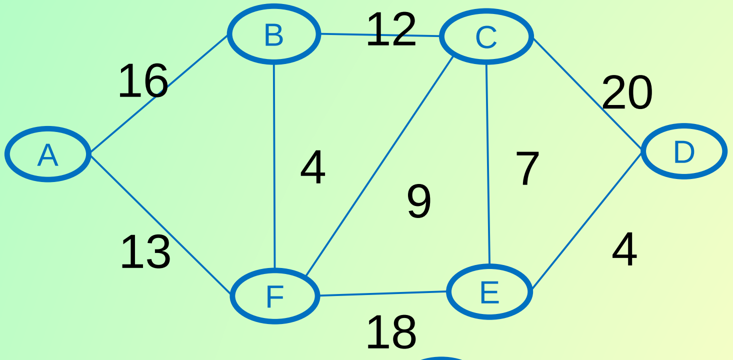
Problem 2

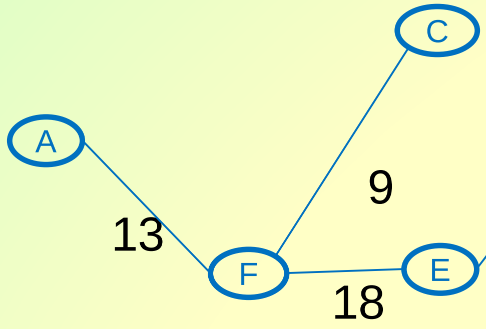
- Recall the *Steiner Tree* problem from last week. Given an undirected graph *G=(V,E)* with nonnegative edge costs and whose vertices are partitioned into two sets, *R* and *S*, find a tree *T* ⊆ *G* with total cost at most C such that for every *v* in *R*, *v* is in *T*. That is, the tree T contains every vertex in *R* (and possibly some in *S*) with a total edge cost of at most *C*.
  - The difficulty in the Steiner tree problem is determining which set S' ⊆ S should be included in T. Give a polynomial time algorithm to find the Steiner tree of minimum cost for the case where the optimal S' is also provided.
  - Give a polynomial time 2-approximation to the problem of finding a minimum-cost Steiner tree from a *complete* graph. You may *not* assume that the optimal S' ⊆ S is given.

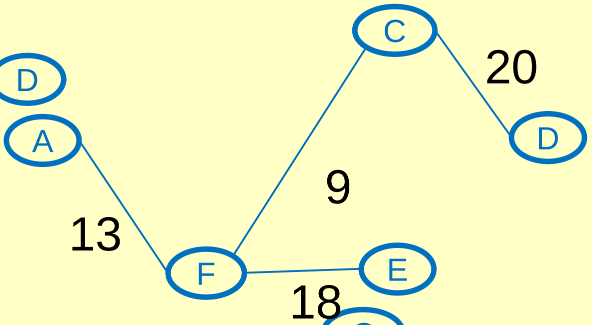


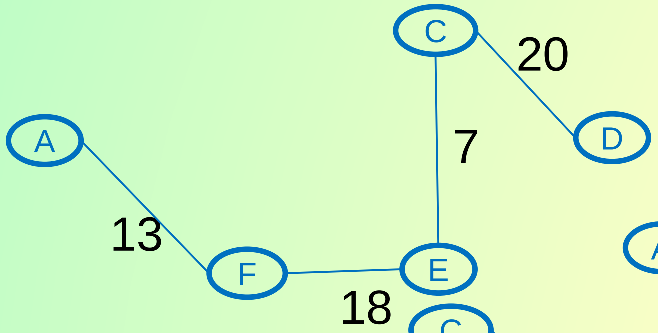
Solution 2

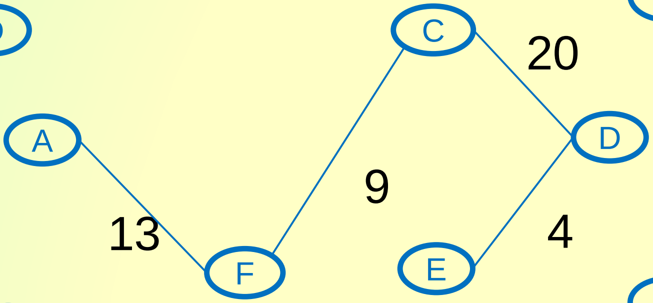
- Suppose you are given C=34, R={A, D, F}, and S’ ⊆ S such that S’={C, E}. We need a polynomial time algorithm for determining the minimum cost Steiner Tree from this data.
- In the following graph, there are a few different trees that contain just those nodes.

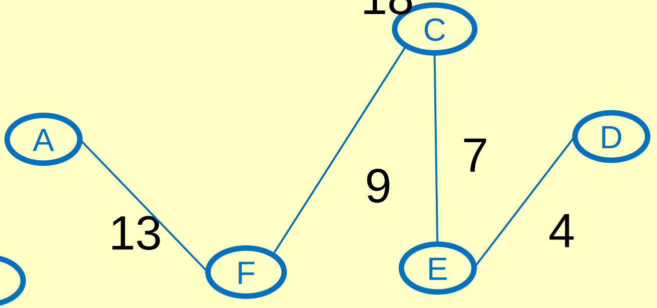


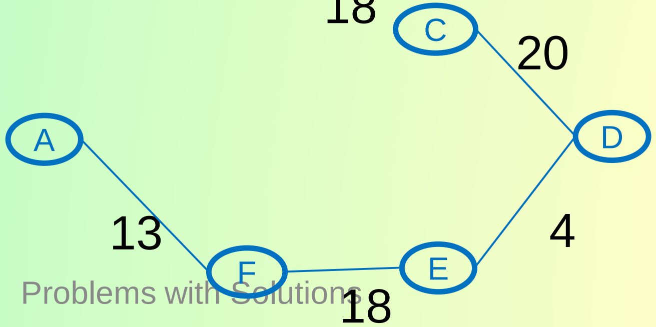


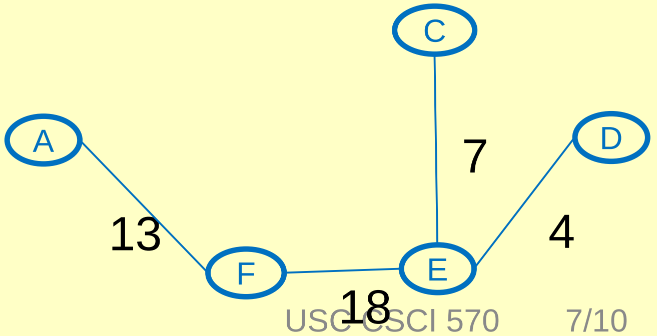






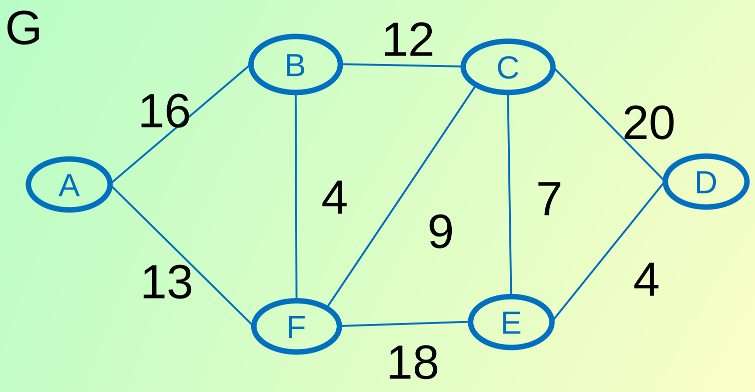


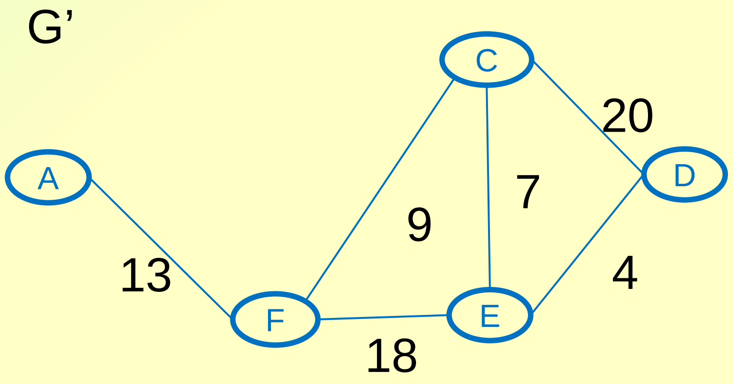


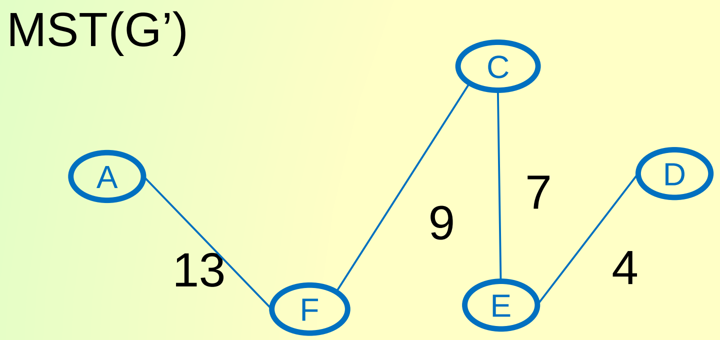


Solution 2

- If we reduce the original graph to a subgraph G'=RUS', then we can find the minimum spanning tree in that graph, which will give us the minimum Steiner Tree in the original graph G. Take V[G']={A,D,F,C,E} with all of the edges incident on those vertices.

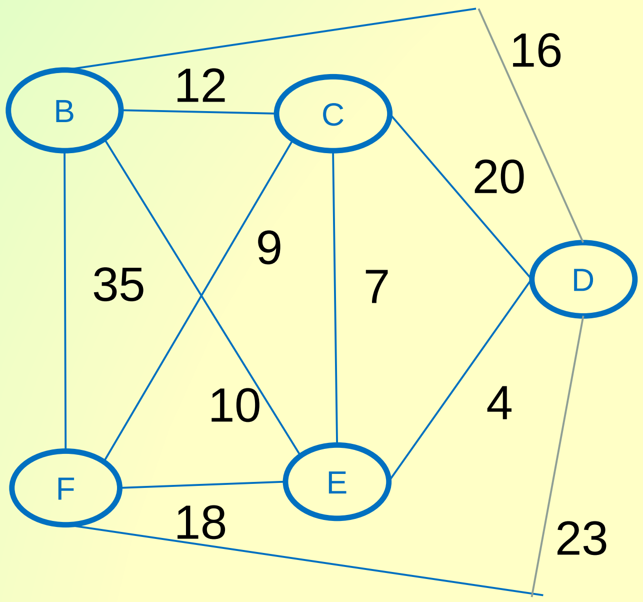






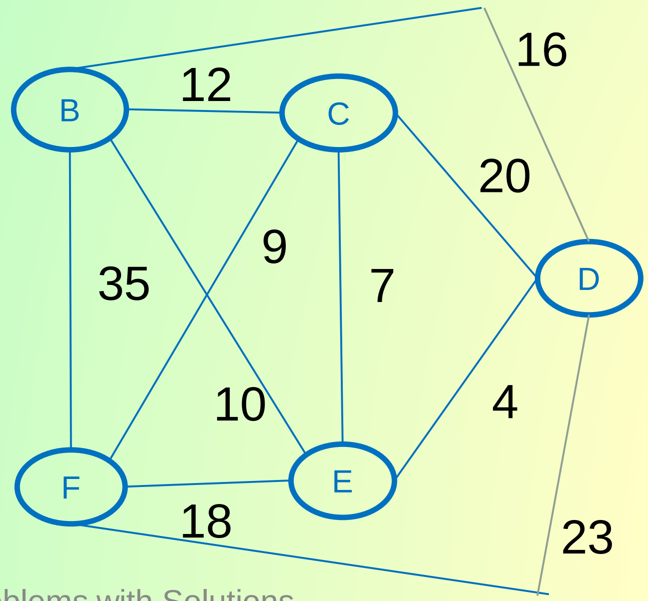
Solution 2

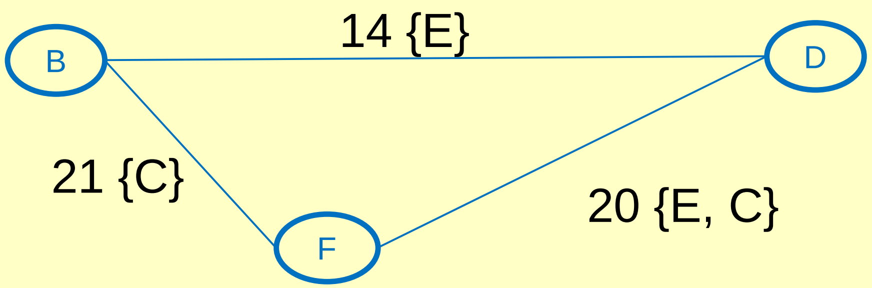
- Give a polynomial time 2-approximation to the problem of finding a minimum-cost Steiner tree from a *complete* graph. You may *not* assume that the optimal S' ⊆ S is given.
- R={B, D, F}



Solution 2

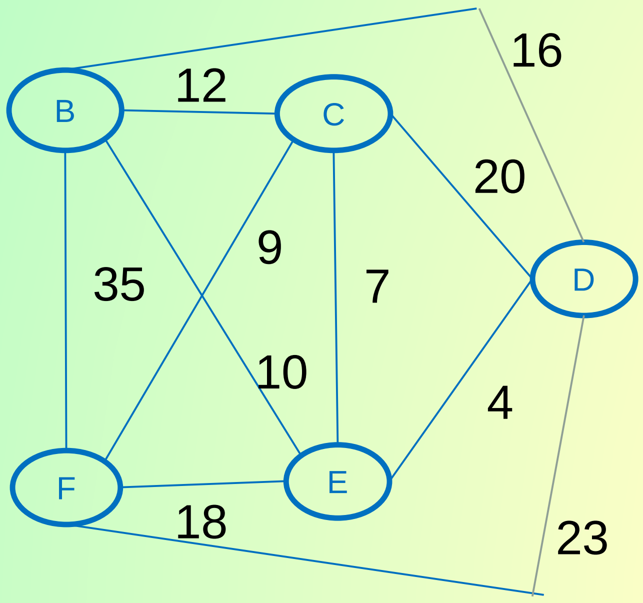
- To approximate the Steiner Tree on a complete graph, we can try to find a minimum spanning tree on a new graph G” where G” consists of all the nodes in R, with the edges being the shortest paths between each pair of nodes.
  - For the graph G below and R={B, D, F}, we would find:
    - Shortest Path from B to D = {B,E,D} with cost=14
    - Shortest Path from B to F = {B,C,F} with cost=21
    - Shortest Path from D to F = {D,E,C,F} with cost=20

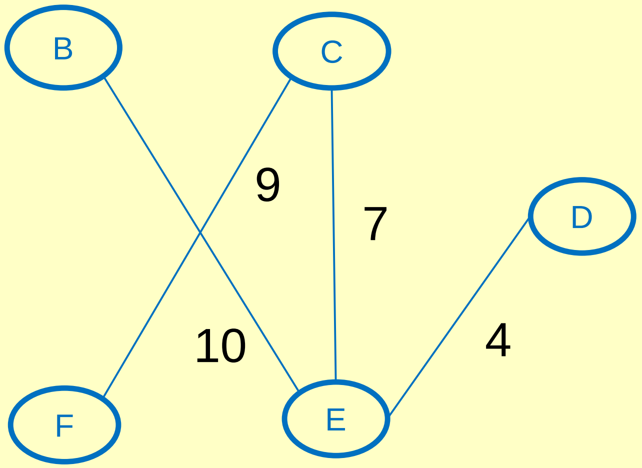


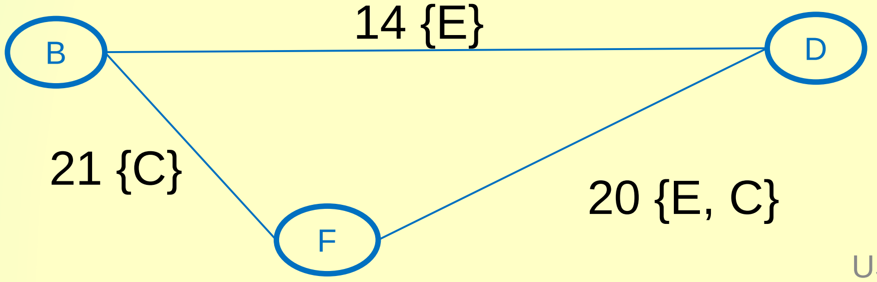


Solution 2

- To approximate the Steiner Tree on a complete graph, we can try to find a minimum spanning tree on a new graph G" where G" consists of all the nodes in R, with the edges being the shortest paths between each pair of nodes.
  - The MST has a cost of 34 (B, D, F)
  - The optimal Steiner Tree has a cost of 30
  - The cost of an MST will always be bounded by 2 times the cost of the optimal Steiner Tree because of the edges used to compute the shortest path being duplicated in shortest paths (i.e. the path in BF contains the same edges as the path in DF)


B
C
D
E
F
12
16
20
35
9
7
10
4
18
23


B
C
D
F
E
9
7
10
4


B
D
F
14 {E}
21 {C}
20 {E, C}