22C:253 Algorithms for Discrete Optimization

Scribe: Rajiv Raman

November 14, 2002

Steiner Forest

We will obtain a 2-approximation algorithm for the steiner forest problem using the primal-dual schema, with the idea of growing duals in a synchronized manner.

Definition Given an undirected graph $G = (V, E)$, a cost function c on the edges, and a collection of disjoint subsets of V, find a minimum cost subgraph in which each pair of vertices belonging to the same subset are connected.

Input
An undirected graph $G = (V, E)$,
a cost function on the edges, $c : E \rightarrow Q^+$, and
a collection of subsets of V, $\{S_1, S_2, \cdots, S_k\}$

Output
A minimum cost subgraph F of G such that $\forall u, v \in S_i, \exists$ a $u - v$ path in F.

Notation

1. Define the connectivity requirement of G be a function r, the set of all unordered pair of vertices to $\{0, 1\}$, such that

$$r(u, v) = \begin{cases} 1 & \text{if } u, v \in S_i \text{ for some } i \\ 0 & \text{otherwise} \end{cases}$$

2. Recall that a cut in G is a partition $(S, \overline{S})$ of V. We will use S to denote such a cut. Define a fuction f such that,

$$f : 2^V \rightarrow \{0, 1\}$$

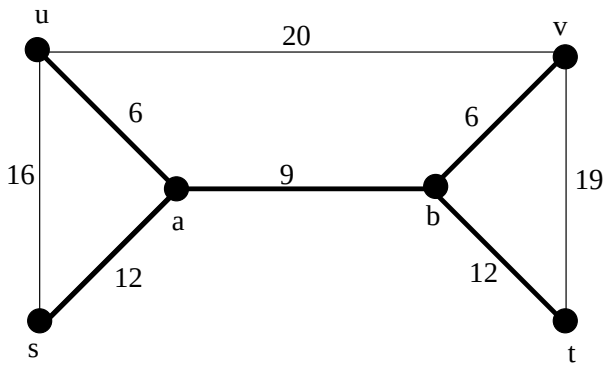
as follows :

$$f(S) = \begin{cases} 1 & \text{if } \exists u \in S, v \in \overline{S} \text{ such that } r(u, v) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Observation For any feasible solution F, $f(S) \leq$ number of edges of F that cross the cut $(S, \overline{S})$. Notice that the converse also holds. i.e., For any subgraph F of G, if for every cut $S \subseteq V$, $f(S) \leq$ number of edges of F crossing $(S, \overline{S})$, then F is a feasible solution to the problem.

Since the converse also holds, we have an alternate charecterization. We can write the optimization problem as :

A min-cost subgraph F of G such that $\forall S \subseteq V$, $f(S) \leq$ number of edges of F crossing $(S, \overline{S})$.



Example 1. Let G be the above graph, with vertices {u, v, a, b, s, t}, and edge weights as displayed on the edges.
Let S1 = {u, v} and S2 = {s, t} be the two subsets of V. Then a feasible soultion is the subgraph defined by the bold edges.

Figure 1: A graph and a feasible subgraph

IP formulation and LP relaxation Let x_e be an indicator variable for each edge $e \in E$. The objective function is :

$$min \sum_{e \in E} x_e c(e)$$

subject to,

$$\sum_{e:e \in \delta(S)} \geq f(S) \qquad \forall S \subseteq V$$

$$x_e \in \{0,1\}; \qquad \forall e \in E$$

where, $\delta(S)$ denotes the set of edges crossing the cut $(S, \overline{S})$.
The LP relaxation is obtained by relaxing $x_e \in \{0,1\} by x_e \geq 0$.

Dual of the LP relaxation Let Y_S denote the dual variable corresponding to cut S. The dual of the primal LP defined above is :

$$max \sum_{S \subseteq V} Y_S f(S)$$

subject to,

$$\sum_{S:e \in \delta(S)} Y_S \leq c(e) \qquad \forall e \in E$$

$$Y_S \geq 0 \qquad \forall S \subseteq V$$

Complementary Slackness Conditions The primal and dual complementary slackness conditions are defined as follows :

Primal:

$\forall$ edge $e \in E$, $x_e \neq 0 \Rightarrow \sum_{S:e\in\delta(S)} Y_S = c(e)$.

We wil use the exact version of the primal complementary slackness condition. i.e., we set $\alpha = 1$

Dual :

$\forall$ cuts $S \subseteq V$, $Y_S \neq 0 \Rightarrow \sum_{e:e\in\delta(S)} x_e = f(S)$.

The approximate version of the dual complementary slackness condition is :

$\forall$ cuts $S \subseteq V$ $Y_S \neq 0 \Rightarrow f(S) \leq \sum_{e:e\in\delta(S)} x_e \leq 2f(S)$.

i.e., $\beta = 2$.

If we can get an integral primal feasible solution, and a dual feasible solution satisfying these constraints, that would imply a factor-2 algorithm for the steiner forest problem. However, we don't know how to satisfy the dual complementary slackness condition.

Algorithm rough sketch

1. Start with $x_e = 0$, $\forall e \in E$ and $Y_S = 0$, $\forall S \subseteq V$
   ($x_e = 0$ represents an infeasible, integral primal solution; and $Y_S = 0$ represents a feasible dual solution).
2. Increase the Y_S's until some edge e becomes tight.
   i.e., $\sum_{S:e\in\delta(S)} Y_S = c(e)$. Throw e into the solution ($x_e = 1$).

Features of the algorithm

1. Y_S values are increased in a synchronous fashion.
2. In each iteration, we will increase Y_S for a small number of cuts S.
   - Define a cut S to be *unsatisfied* if $f(S) = 1$, but no edge in the currently chosen set crosses S.
   - Define an *active cut* as a minimal unsatisfied cut (with respect to inclusion).
     In each iteration, we increase Y_S synchronously $\forall$ active cut S.

Claim : A cut S is active iff it is a connected component in the current subgraph and $f(S) = 1$. This claim will be proved later on.

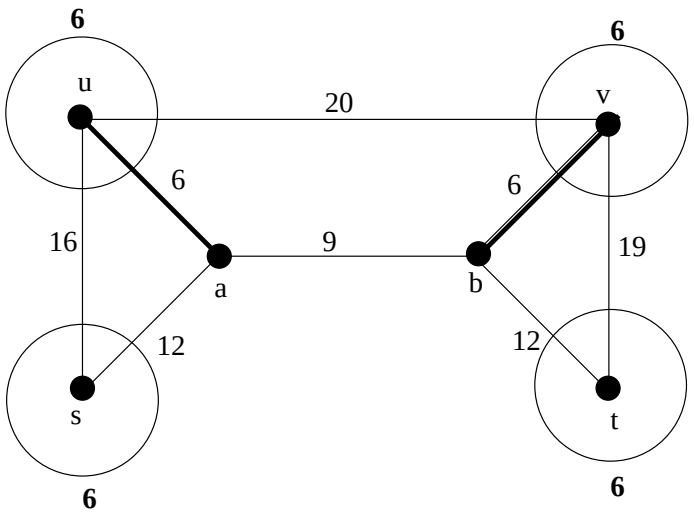
Algorithm

1. Set $F' = \phi$ (the set of chosen edges; corresponds to saying $x_e = 0$ $\forall e \in E$)
2. while($\exists$ an unsatisfied cut) do
   Increase Y_S value synchronously for every active cut S, until some edge e becomes tight.
   $F' \leftarrow F' \bigcup \{e\}$.
3. endwhile

4. Prune the redundant edges from F'. i.e., delete every edge e from F' such that $F' - \{e\}$ is feasible.

Claim : F' is a primal feasible solution, and y is a dual feasible solution.
Proof : Before the pruning step, F' satisfies all connectivity requirements (because there are no unsatisfied cuts). F' is a forest because of the property of active cuts. edges that become tight, and are added in any iteration connect one connected component to another.

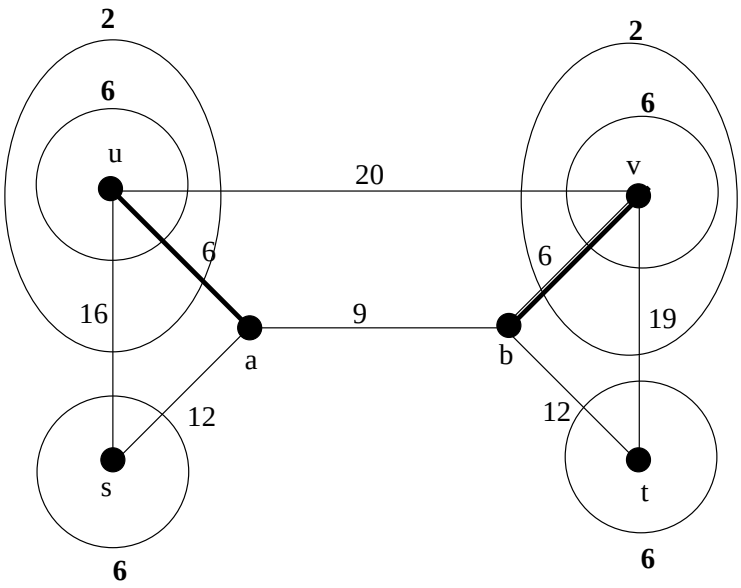
Therefore, for any u, v such that the connectivity requirement is 1, ther is a unique uv path. Hence, all edges in the path are non-redundant and hence not deleted. Hence, after the pruning step also, connectivity is maintained. □



In the begining of the iteration, the active sets are {s}, {t}, {u}, {v}. When their dual variables are raised by 6 each, the edges {u,a} and {b,v} go tight. One of them, say {u,a} is picked, and in the next iteration, {u,a} replaces the active set {u}.In the next iteration, without having to raise any of the varaibles, the edge {b,v} becomes tight, and {b,v} replaces the active set {v}. The edges that are picked are marked in bold.

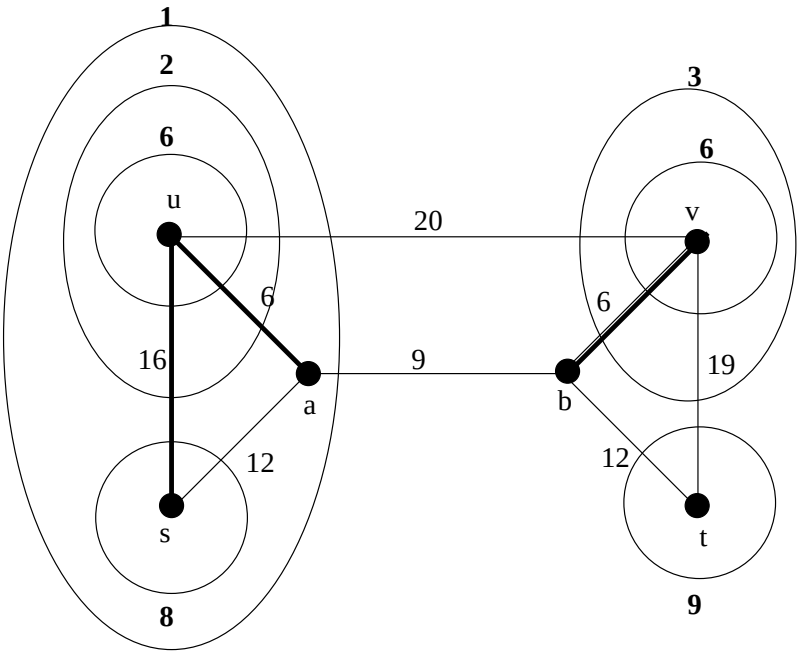
Figure 2: First instance when the Y_S values become tight

Example execution of the algorithm Consider the graph and the subsets of the vertices as in example 1. The following figures show the execution of the algorithm on the above graph.



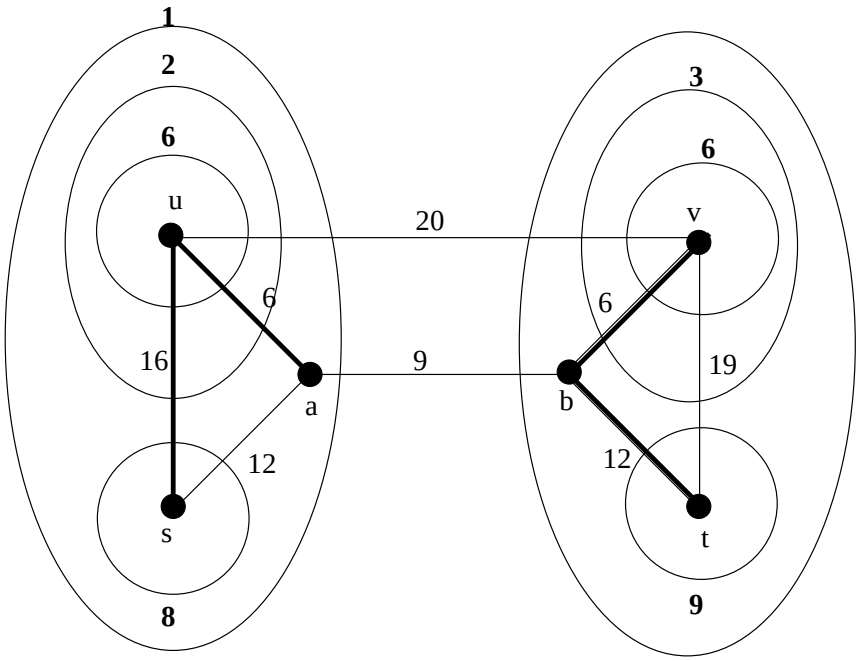
When the Y_S values are increased by 2, the edge {u,a} becomes tight, and the active sets are now : {u,s,a}, {v,b}, {t}.

Figure 3: Instance when the Y_S values become tight



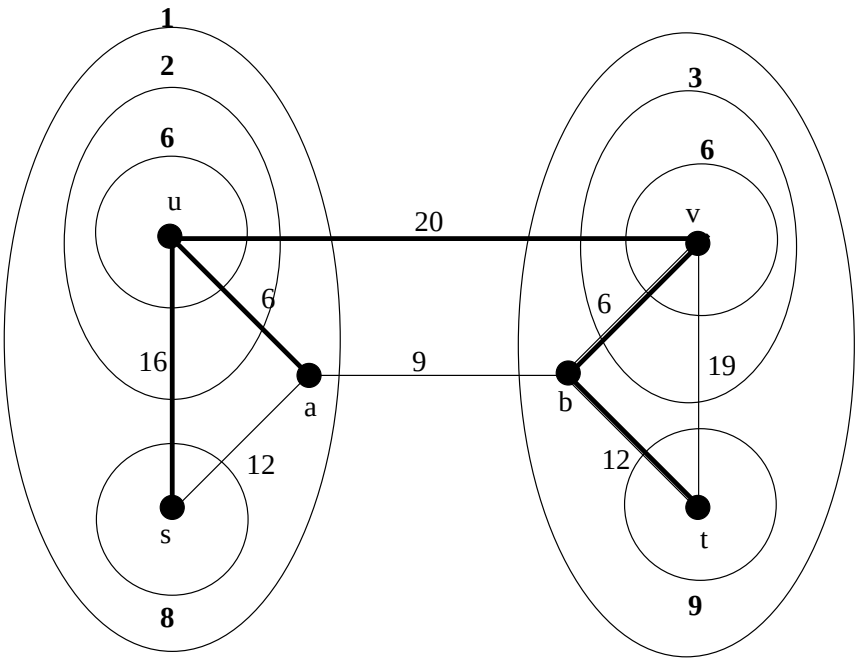
At the next iteration, the edge {b,t} becomes tight. The active sets are {u,s,a}, {v,b,t}

Figure 4: Instance when the Y_S values become tight



At the next iteration, the edge {u,v} becomes tight

Figure 5: Instance when the Y_S values become tight



The subgraph produced by the algorithm consits of the edges {u,a}, {v,b}, {u,a}, {b,t}, {u,v}

Figure 6: Final output of the algorithm