CMPUT 675: Approximation Algorithms **Fall 2015**

Lecture 9 (Sept 29, 2015): Primal/Dual for Steiner Forest

Lecturer: Mohammad R. Salavatipour *Scribe: Dean Koch*

9.1 Primal-Dual Method

Recall our formulation of the primal and dual LP problems: Given vectors $\mathbf{c} = (c_1, c_2, ..., c_n)$, $\mathbf{b} = (b_1, b_2, ..., b_m)$, and an $m \times n$ matrix A, we seek solutions $\mathbf{x} = (x_1, x_2, ..., x_n)$, $\mathbf{y} = (y_1, y_2, ..., y_m)$ satistfying:

Primal

$$\min_{\mathbf{x}} \{\mathbf{c} \cdot \mathbf{x}\}$$
$$\text{such that } A\mathbf{x} \geq \mathbf{b}$$

Dual

$$\max_{\mathbf{y}} \{\mathbf{b} \cdot \mathbf{y}\}$$
$$\text{such that } \bar{A}\mathbf{y} \leq \mathbf{c}$$

By the LP duality theorem, a finite optimum, $\mathbf{x}^*$, for the primal implies the existence of a finite optimum, $\mathbf{y}^*$, for the dual (and vice versa), and at these solutions the objective functions are equal: $\mathbf{c} \cdot \mathbf{x}^* = \mathbf{b} \cdot \mathbf{y}^*$. For such solutions, we also have the **complementary slackness property**:

Primal slackness condition

for $i = 1, ...n$

$$\text{either } x_i^* = 0 \text{ or } \sum_{j=1}^{m} a_{ij} y_j^* = c_i \tag{9.1}$$

Dual slackness condition

for $j = 1, ...m$

$$\text{either } y_j^* = 0 \text{ or } \sum_{i=1}^{n} a_{ij} x_i^* = b_j \tag{9.2}$$

If a relaxed version of these slackness conditions can be shown to hold for feasible solutions $\mathbf{x}$ and $\mathbf{y}$ (not necessarily optimal), a bound on the value of the objective for the optimal solution will result

Lemma 1 *Fix any $\alpha > 0$ and $\beta > 0$ for which $\alpha\beta > 1$. If, for feasible solutions $\mathbf{x}$ and $\mathbf{y}$, the following* ***relaxed slackness conditions*** *hold:*

Relaxed primal slackness condition

for $i = 1, ...n$

$$\frac{c_i}{\alpha} \leq \sum_{j=1}^{m} a_{ij} y_j \leq c_i \tag{9.3}$$

Relaxed dual slackness condition

for $j = 1, ...m$

$$b_j \leq \sum_{i=1}^{n} a_{ij} x_i \leq \beta b_j \tag{9.4}$$

then it follows that $\sum_{i=1}^{n} c_j x_j \leq \alpha\beta \sum_{j=1}^{m} b_j y_j$

Proof. For each $i = 1, ...n$ multiply the relaxed primal slackness condition by αx_i, yielding n inequalities. Sum together all of these to obtain:

$$\sum_{i=1}^{n} c_i x_i \leq \sum_{i=1}^{n} \left(\left(\sum_{j=1}^{m} a_{ij} y_j \right) x_i \right) \leq \alpha \sum_{i=1}^{n} c_i x_i$$

Similarly, for each $j = 1, ...m$ multiply the relaxed dual slackness condition by αy_j, to get m inequalities, the sum of which is:

$$\alpha \sum_{j=1}^{m} b_j y_j \leq \sum_{j=1}^{m} \left(\left(\sum_{i=1}^{n} a_{ij} x_i \right) y_j \right) \leq \alpha\beta \sum_{j=1}^{m} b_j y_j$$

In both cases, the order of summation in the double sum can be exchanged to show that the middle term of each chain of inequalities is identical.

■

Notice that if a finite optimal solution $\mathbf{x}^*$ exists for the primal, then we can use Lemma 1 and the LP duality theorem to bound the value of its objective function:

$$\sum_{i=1}^{n} c_j x_j \leq \mathbf{c} \cdot \mathbf{x}^* = \mathbf{b} \cdot \mathbf{y}^* \leq \sum_{j=1}^{m} b_j y_j \leq \alpha\beta \sum_{j=1}^{m} b_j y_j \tag{9.5}$$

9.1.1 Approximation using primal-dual

We can sometimes use the relaxed complementary slackness conditions to build an approximation scheme. The **general idea** is:

1. Start with an infeasible primal and a feasible (but non-optimal) dual (*e.g.* $\mathbf{x} = \mathbf{y} = 0$)
2. Iteratively improve the feasibility of the primal, and optimality of the dual (always maintaining its feasibility)
3. Once a dual variable goes tight (*i.e.* satisfying (9.2)), stop increasing it. Increase the variables in the primal corresponding to tight dual constraints. Increase these only by integral amounts (to ensure an integral solution)
4. Once the primal is feasible, use equation (9.5) to bound its error (no more than factor $\alpha\beta$)

It is not always clear when this strategy will work, but when it does, it often gives a nice and clean algorithm that is usually efficient as well. A common example in textbooks is set cover. We will consider the more interesting Steiner forest problem.

9.1.2 An example: Steiner forest

The **generalized Steiner tree**, or **Steiner forest** problem is the following:

Input: A graph $G(V, E)$ with a cost function $c : E \to \mathbb{Q}^+$ defined on the edges, and a disjoint collection of subsets of V, say $S_1, S_2, ...S_k \subseteq V$

Goal: Find a minimum cost subgraph in which each S_i $(i = 1...k)$ belongs to a connected component

In Figure 9.1, for example, we are given subsets $S_1 = \{s, t\}$ and $S_2 = \{u, v\}$ and must identify the minimum cost Steiner tree(s) needed to (individually) interconnect both S_1 and S_2. A feasible solution is highlighted.

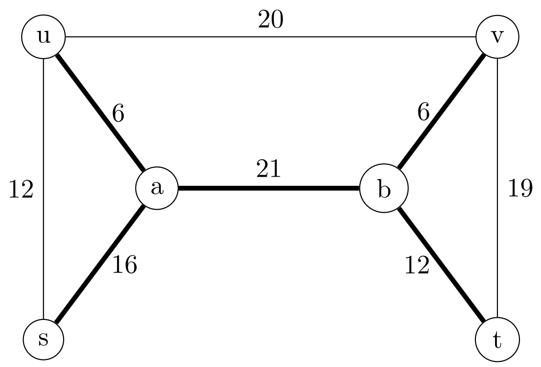

Figure 9.1: Example Steiner forest problem. A feasible solution is indicated by the bolded edges

9.1.3 LP formulation of Steiner forest

We can rephrase the Steiner forest problem with the requirement of pairwise connectivity of all pairs within each of the sets S_i $(i = 1...k)$. First, some definitions:

connecting requirement, r, is defined on all vertex pairs u, v of V:

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

cut δ, for a given set $S \subseteq V$, is the set of edges having one end in S and one end outside S:

$$\delta(S) := \{e = (u, v) | u \in S, v \notin S\}$$

function $f(S)$ indicates whether, in the cut of S, we separate a vertex pair that must be connected:

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

indicator variable, $\mathbf{x} = (x_1, x_2, ...x_{|E|})$, to indicate edges picked for our solution:

$$x_e := \begin{cases} 1 & \text{if edge e is picked for our solution} \\ 0 & \text{otherwise} \end{cases}$$

Now we can define the LP problem:

Primal for Steiner forest

$$\min_{\mathbf{x}} \left\{ \sum_{e \in E} c_e x_e \right\}$$

$$\text{s.t. } \forall S \subseteq V \sum_{e \in \delta(S)} x_e \geq f(S) \tag{9.6}$$

Dual for Steiner forest

$$\max_{\mathbf{y}} \left\{ \sum_{S \subseteq V} f(S) y_s \right\}$$

$$\text{s.t. } \forall e \in E : \sum_{S | e \in \delta(S)} y_S \leq c_e \tag{9.7}$$

If the condition (9.7) holds with equality for a given edge, we say that the edge has *gone tight*. We now consider an algorithm following the guidelines in 9.1.1. Note that we may start with $\mathbf{x} = \mathbf{y} = \mathbf{0}$, since this is a feasible solution for the dual. We then integrally raise the variables until the primal is feasbile.

The idea is to raise the y_S values simultaneously until an edge e goes tight, then set $x_e = 1$ (picking this edge to include in our solution). We then freeze the corresponding y_S values, and repeat (with a reduced number of free y_S variables). The challenge is to narrow down the number of sets S that must be considered (since this could be exponential at the beginning).

As a first step, consider that, in the interest of satisfying constraint (9.6), we need only increase the y_S values for which $f(S) = 1$ (since our solution $\mathbf{x}$ has no negative components, if $f(S) = 0$, then (9.6) is always trivially satisfied). Define the degree of set S to be the number of picked edges crossing cut $(S, \overline{S})$

Primal: For each $e \in E, x_e \neq 0 \Rightarrow \sum_{S: e \in \delta(S)} y_s = c_e$. Equivalently, every picked edge must be tight

Relaxed dual conditions: The following relaxation of the dual conditions would have led to a factor 2 algorithm: for each $S \subseteq V, y_s \neq 0 \Rightarrow \sum_{e: e \in \delta(s)} x_e \leq 2f(S)$, i.e., every raised cut has degree at most 2. However, we do not know how to ensure this condition. But we can still obtain a 2-approximation algorithm by relaxing this conidition further.

We will see that in fact only a subset of these y_S must be incremented at each step. To make this concrete, two more definitions are necessary. If $S \subseteq V$, we say:

Definition 1
- *S is **unsatisfied** if $f(S) = 1$ and no edges from $\delta(S)$ have been picked yet*
- *S is **active** if it is a minimal (inclusion-wise) unsatisfied set*

9.1.4 Formal definition and analysis of the algorithm

Lemma 2 *Set S is active if and only if it is a connected component in the currently picked forest, and $f(S) = 1$.*

Proof. Let S be an active set and suppose that S comprises a part (but not all) of a connected component of F. Then there exists an edge from S to $F - S$. Since this edge belongs to $\delta(S)$, and has been picked, then S is satisfied by definition and therefore cannot be active (contradiction).

Now suppose S is active and contains multiple connected components. Since $f(S) = 1$, one of these components must contain a vertex in one of the S_i $(i = 1...k)$. But then this component, on its own, is unsatisfied, contradicting minimality.

```
Steiner Forest Primal-Dual Algorithm

    Input: Graph $G = (V, E)$, subsets $S_i$, $i = 1...k$ and a cost $c_e \in \mathbb{Q}^+$ for each edge
    Output: A minimum cost subgraph $G' \subseteq G$ such that $\forall u, v \in S_i$ $(i = 1...k)$, $u$ is connected to $v$
    1. $F \leftarrow \emptyset$
    2. $y_S \leftarrow 0, \quad \forall S \subseteq V$
    3. while $\exists S \subseteq V$ which is unsatisfied, do
    4.    For each active $S$, simulataneously raise $y_S$ until an edge $e$ goes tight
    5.    $F \leftarrow F \cup \{e\}$
    6. $F' \leftarrow \{e \in F | F - e$ is infeasible$\}$
    7. return $F'$
```

Figure 9.2: Algorithm Primal-Dual

For the reverse implication, we simply assume that S is a connected component of F with $f(S) = 1$. Since it is connected in F, no chosen edge can leave S (i.e. no edge in $\delta(S)$ is picked). For the same reason, it is minimal, for any subset would exclude an edge in F and thus contribute a chosen edge to $\delta(S)$. Therefore, S must be active. ■

9.1.5 An example to visualize the algorithm

Figure 9.1 shows a sample run of the algorithm. Suppose we have two disjoint subsets $S_1 = \{u, v\}$ and $S_2 = \{s, t\}$. At the beginning of the algorithm, $\{u\}, \{v\}, \{s\}, \{t\}$ are four active sets, each of which contains one vertex only. The algorithm raises their y_S values simultaneously, and stops at the value of 6 when edge ua and bv become tight. One of them, say ua is picked, and the iteration ends. In the second iteration, $\{u, a\}$ replaces $\{u\}$ as an active set and the algorithm finds already tight edge bv. Then algorithm updates active sets and raise value of their variables, and continues. The next edge to go tight is us when set $\{u, a\}$ has value 2 and $\{s\}$ has value 8; the two sets merge into one active set. next bt goes tight when $\{b, v\}$ has value e and $\{t\}$ has value 9. At this point set $\{u, s, a\}$ has value 1. Finally edge uv goes tight. Figure below shows the final result of running the algorithm. In this figure, active sets are shown with a closed area containing them and the final value of their variables are depicted beside the boundary of these closed areas. The bold edges are the edges added to F in the loop. At the end, all edges in F except the redundant edge ua are added to F' and returned.

9.1.6 analysis continued: runtime, feasibility, and approximation ratio

Lemma 3 *Algorithm 9.2 completes in polynomial time.*

Proof. Since we add an edge in every iteration of the while loop, the total number of iterations is at most $|V| - 1$. In each iteration we simply identify all the connected components of the subgraph, F (which can be done in polynomial time), and check whether $f(S) = 1$ for each of these. Then by Lemma 2, we have identified all active sets. Noting that since the existence of an active set implies the existence of an unsatisfied set, we have evaluated the condition of the loop in this process. Thus the loop completes in polynomial time.

The last step of the algorithm checks if edges can be deleted to yield a feasible solution. Since F is a tree, this amounts to asking if the edge lies on the path between any two vertices in the same S_i subset. This can also be implemeneted in polynomial time.

Therefore the algorithm completes in polynomial time.

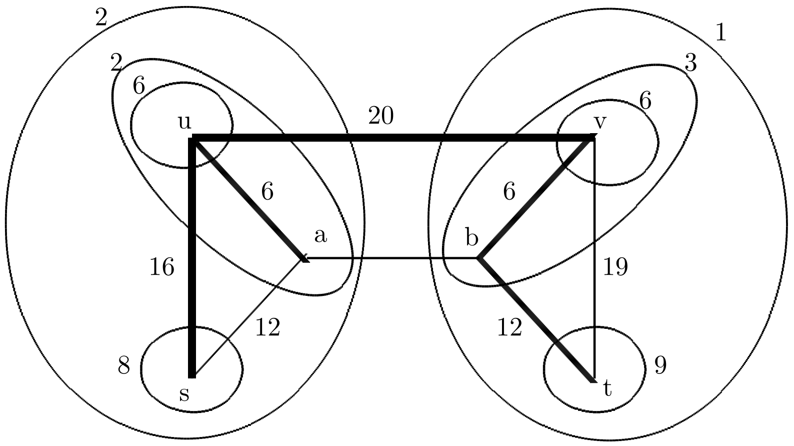

Figure 9.1: A sample run of algorithm STEINER-FOREST$(G, \mathcal{S})$.

■

Lemma 4 *Algorithm 9.2 yields feasible primal and dual solutions.*

Proof. The dual condition (9.7) can never be violated, since we begin with $\mathbf{y} = \mathbf{0}$ (feasible) and we freeze the values of the y_S variables as soon as an edge becomes tight (*i.e.*, once (9.7) holds with equality, the sum on the left hand side can no longer increase). As for the primal, consider that as long as an active set exists we keep iterating. This means that once the algorithm terminates, there are no active sets. *i.e.* whenever $f(S) = 1$, we have at least once edge in $\delta(S)$ selected, and thus the sum in (9.6) is no smaller than 1. Therefore the primal solution is feasible.

■

For the next lemma, we require a definition:

Define the **degree in** F' of any subset $S \subseteq V$ by:

$$\deg_{F'}(S) = \text{ the number of edges of } \delta(S) \text{ in } F'$$

Lemma 5 *Let C be any connected component with respect to the selected edges F (in the current iteration of algorithm 9.2). If $f(C) = 0$ then* $\deg_{F'}(C) \neq 1$

Proof. Suppose $f(C) = 0$ and $\deg_{F'}(C) = 1$, and denote by e the edge that leaves C to connect with a vertex in $F' \backslash C$. Since e was not removed in the pruning step of the algorithm, there must be some vertex pair (u, v) for which their connecting path includes e. Furthermore, since e is the only edge connecting C to $F' \backslash C$, it must be that case that one of u, v lies in C and the other does not. However, since we must also have $r(u, v) = 1$, then by definition $f(C) = 1$, contradicting our assumption. ■

Lemma 6 *For any primal and dual solution computed by algorithm 9.2,*

$$\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S \tag{9.8}$$

Proof. First note that since an edge is added to F' only when it is tight, we have:

$$\begin{aligned}\sum_{e\in F'} c_e &= \sum_{e\in F'}\sum_{S|e\in\delta(S)} y_S\\ &= \sum_{S\subseteq V}\sum_{S|e\in\delta(S)\cap F'} y_S\\ &= \sum_{S\subseteq V}\deg_{F'}(S)y_S \end{aligned} \tag{9.9}$$

We can break this sum into pieces by observing that the y_S values were increased from their starting value of zero in stages; consider only the increment Δ_i made to the y_S values in the i^{th} iteration of the algorithm. This increment applied only to the active sets, and contributed the following quantity to the sum in (9.9):

$$\Delta_i \sum_{S \text{ active at step i}} deg_{F'}(S)$$

Now consider the connected components of F in this iteration. Suppose we coalesce each component into a single node. We may then connect these coalesced nodes together, by adding edges that appear in subsequent iterations (and are not pruned at the end). This generates a new graph H' (see Figure 9.3) in which the node corresponding to active set S has degree $deg_{F'}(S)$.

If we delete all isolated nodes from H', then what remains is a tree, which has average degree no more than 2. Thus:

$$\sum_{S \text{ active at step i}} deg_{F'}(S) \leq 2\times\{\#\text{ of nodes in } H'\} = 2\times\{\#\text{ of active sets in step i}\}$$

Applying this result to (9.9), we have:

$$\begin{aligned}\sum_{e\in F'} c_e &= \sum_{S\subseteq V}\deg_{F'}(S)y_S\\ &= \sum_i \Delta_i\left(\sum_{S \text{ active at step i}} deg_{F'}(S)\right)\\ &\leq \sum_i \Delta_i\,(2\times\{\#\text{ of active sets in step i}\})\\ &= 2\sum_i \Delta_i\,\{\#\text{ of active sets in step i}\}\\ &= 2\sum_{S\subseteq V} y_S\end{aligned}$$

■

At last, we conclude that the value of the objective for the primal is no more than twice that of the dual. Since the optimal lies in between these values, we conclude that algorithm 9.2 is a 2-approximation.

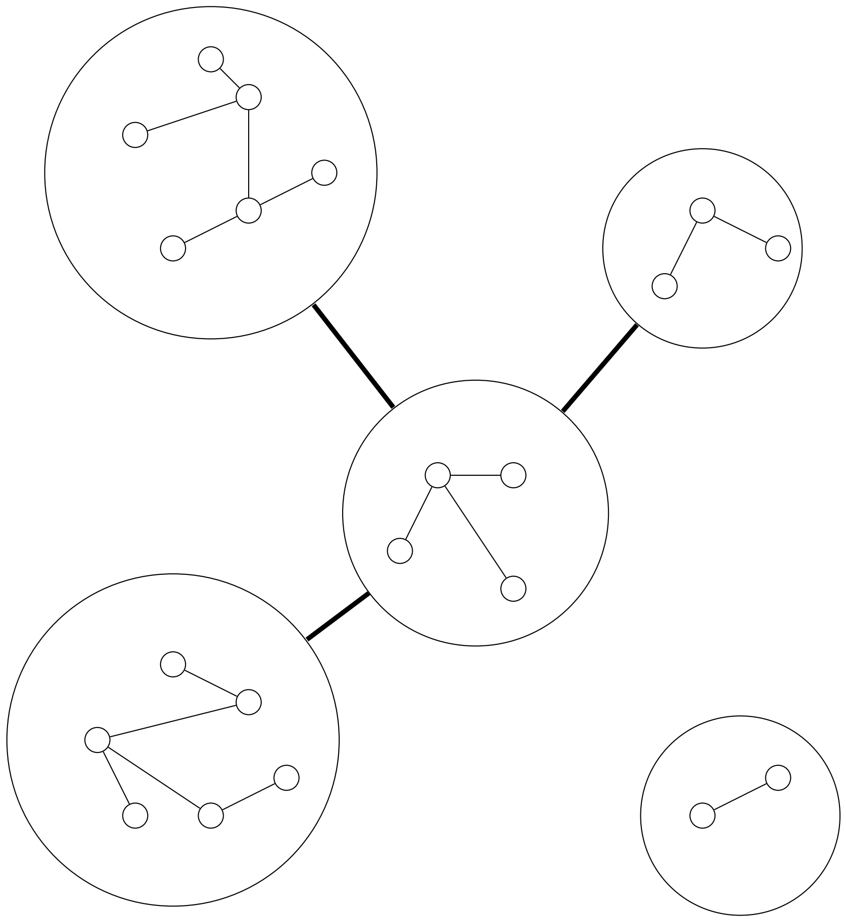

Figure 9.3: Illustration of how connected components of F are grouped together into vertices (large circles) of the new graph H', which inherits edges (bolded edges, added in subsequent iterations) in F', forming a forest. We delete isolated nodes, and obtain a tree with average degree at most 2.