CS787: Advanced Algorithms	**Scribe:** Ankit Pensia
Lecture 11: Primal Dual wrap-up; Online algorithms	**Date:** 2/28/2019

11.1 Recap

Steiner Tree problem: Given $G = (V, E)$ with edge costs c_e and terminal set $T \subset V$, find the cheapest subgraph spanning T.

We wrote the following LP relaxation in primal and dual

$$\begin{aligned} &\text{(P)} && &&\text{(D)} \\ &\min \quad \sum_e c_e x_e && &&\max \quad \sum_{s\in\mathcal{S}} y_S && (11.1.1)\\ &\text{subject to} \sum_{e\in\delta(S)} x_e \geq 1, \quad \forall S \in \mathcal{S} && &&\text{subject to} \quad y_S \geq 0, \quad \forall S \in \mathcal{S} && (11.1.2)\\ &\qquad x_e \geq 0, \quad \forall e && &&\sum_{S: e\in\delta(S)} y_S \leq c_e, \forall e && (11.1.3) \end{aligned}$$

where $\mathcal{S} = \{S \subset V : |S \cap T| \geq 1 \text{ and } |S^c \cap T| \geq 1\}$ and $\delta(S)$ denotes the set of edges crossing S. The constraints in Eq. (11.1.2) corresponds to the dual complementary slackness while the constraints in Eq. (11.1.3) corresponds to the primal complementary slackness.

11.2 Approximation algorithm

We'll give a 2-approximation algorithm for this problem. Note that tighter approximations for this problem are known in literature. Following remarks regarding the hardness of problem are in order:

1. For $T = V$, the optimal steiner tree is the minimum spanning tree (MST), which has efficient algorithm.
2. The problem in general is hard because we don't know which vertices would be Steiner vertices (non-terminal vertices).
3. The LP is difficult to solve exactly because there are exponentially large number of constraints in the primal problem, and thus exponentially large number of variables in dual.

The solution that we would give would only have polynomially number of non-zero y.

11.2.1 Generic template

In the last lecture, we covered the following template for solving primal dual LPs.

1. Start with $x = 0$, $y = 0$.

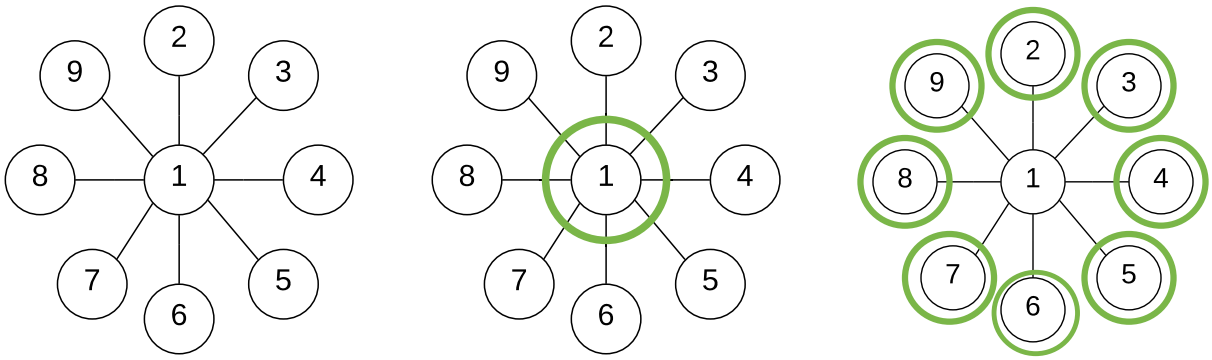

Figure 11.2.1: Example where the order of dual variables matter.

2. Pick some dual variables.
3. While x is infeasible:
   (a) Pick some unfrozen dual variables to raise.
   (b) Raise them until some dual constraint goes tight
   (c) Set $x_e = 1$ for any such constraint and freeze corresponding y's.

For this template, we have the following result

1. Primal complementary slackness is satisfied at each step.
2. y is feasible at each step.
3. x is feasible at the end.

11.2.2 Examples

We'll start with this basic template and see some cases where it is important to (1) consider the order of dual variables and (2) do some post-processing. We'll make the necessary changes in the final algorithm accordingly.

Example: Consider the graph given in Figure 11.2.1(a) with edge weights to be 1 for every edge e. Let the terminal set be V; Thus, the optimal steiner tree is the whole graph. We can choose one of two possible dual variables shown in Figure 11.2.1 (b) and (c). The choice of dual variable S is shown in green. It can be seen that (b) has cost 1 and (c) has cost $n-1$. As we want to maximize the dual objective, (c) is a preferable choice.

Example: Consider the clique K_5 shown in Figure 11.2.2. The terminal set is again V. If we choose the sets shown in green, we can increase each such y_S till 0.5. In the primal problem, each edge would be selected and thus the output would be the whole graph. Since it contains cycles, it is sub-optimal. We'd thus remove cycles at the end.

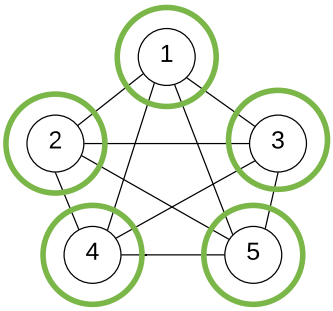

Figure 11.2.2: Example to show that post-processing is needed.

11.2.3 Refined algorithm for 2-Approximation (Reverse Delete)

At any iteration, let $F = \{e : x_e = 1\}$, the set of edges selected so far. Let $\mathcal{M}$ = minimal unsatisfied sets in S' under the solution F = connected components in (V, F) that contains terminals.

We refine the template for primal-dual in the following algorithm:

1. Initialize $F = \emptyset, x = 0, y = 0$
2. While $\mathcal{M} \neq \emptyset$
   (a) simultaneously raise y_s for all $S \in \mathcal{M}$ at a uniform rate until some constraint in (D) becomes tight
   (b) $\forall e$ with $\sum_{S:e\in\delta(S)} y_S = c_e$, set $X_e = 1$. Add e to F.
   (c) Update $\mathcal{M}$, freeze all y_S variables with $e \in \delta(S)$.
3. Consider edges $e \in F$ in reverse order of addition. Remove e from F if that doesn't violate feasibility.

Claim 11.2.1 *The above algorithm achieves approximation ratio of* 2.

Proof: Similar to the template above, we see that when the algorithm terminates:

1. x and y are feasible.
2. Primal CS holds.

By primal complimentary slackness, the cost of solution is

$$\begin{aligned}\sum_{e\in F} c_e &= \sum_{e\in F} \sum_{S:e\in\delta(S)} y_S \\ &= \sum_S y_S \sum_{e\in\delta(S)\cap F} 1 \\ &= \sum_S y_S \times \deg_F(S)\end{aligned}$$

where $\deg_F(S) = |\{e \in F \cap \delta(S)\}|$. We'll show that $\sum_S y_S \times \deg_F(S) \leq 2\sum_S y_S$, which will prove the result.

Claim 11.2.2

$$\frac{d}{dtime}\sum_S y_s deg_F(S) \leq 2\frac{d}{dtime}\sum_S y_s$$

Setting $\frac{d}{dtime}\sum_S y_s = 1$ *for* $S \in \mathcal{M}$ *and* 0 *otherwise, we can restate it as follows*

$$\sum_{S\in\mathcal{M}} deg_F(S) \leq 2|\mathcal{M}|.$$

Proof: Any leaf node that is not in $\mathcal{M}$ will be deleted in "reverse delete". Then by definition of tree, $\deg_F(S) \leq 2$ for any S, thus the statement holds. ■

■

11.3 Online Algorithms

Online algorithms pertain to the problems where the input is not revealed at once in the beginning, but is given to the user one at a time. We introduce online algorithms with the following toy example: You want to go skiing but you don't how many times you'd go skiing in advance. Every time you go skiing, you face a decision: either pay \$1 to rent the ski or pay $\$M$ to buy it forever. Concretely,

1. For every day $i \in \{1, \ldots, n\}$:
   (a) If you haven't bought the ski already, you either
      i. pay \$1 to rent the ski.
      ii. or pay $\$M$ to buy the ski.
   (b) Otherwise, you pay nothing.

The objective is to reduce the overall cost. It is easy to see that the optimal cost of this problem is $\texttt{OPT} = \min(M, n)$. However, n is not known to the algorithm. Consider the algorithm : `ALG` - you rent until you've spent M dollars, then you buy it.

The cost of this algorithm is

$$\begin{aligned}\text{cost}(\texttt{ALG}) &= \begin{cases} n, & \text{if } n \leq M \\ 2M, & \text{otherwise} \end{cases} \\ &\leq 2\min(n, M)\end{aligned}$$

For a minimzation problem, we define the competitive ratio of an algorithm `ALG` to be

$$\text{Competitive ratio} = \max_{\text{input x of any length}} \frac{\texttt{ALG}(x)}{\texttt{OPT}(x)}$$

It is different from approximation ratio in principle because in competitive ratio, we are about lack of knowledge and not about computational considerations. Instead of hardness of approximation, we have info-theoretic lower bounds based on adversarial arguments. We classify adversaries into two categories:

1. Adversary can see algorithm's coin flips.
2. Adversary cannot see this, i.e., adversary is oblivious (to algorithm's randomness).