**CS 6550: Randomized Algorithms** **Spring 2019**

# Lecture 14

# Primal Dual Method: Approximate Algorithm for Steiner Forest

February 26, 2019

*Lecturer: Vivek Madan* *Scribes: Ivan Dario Jimenez*



## 14.1 Generic Primal-Dual Algorithm

**Algorithm 1:** Generic Primal-Dual Algorithm

1. Formulate problem as an integer program. Relax it to get a linear program (LP) and its dual(DLP).
2. Generate an infeasible solution $x_0$ to the LP and feasible solution $y_0$ to the DLP.
3. **while** $x_i$ *is not feasible* **do**
4. $\quad y_{i+1} \leftarrow$ update $y_i$ s.t. at least one more dual constraint is tight.
5. $\quad x_{i+1} \leftarrow$ set corresponding primal variable in $x_i$ to be 1
6. **return** $x_{end}$

**Remark:** Primal-dual algorithm *does not* solve primal or dual programs.

## 14.2 $2\left(1-\frac{1}{k}\right)$ - approximation algorithm for steiner tree problem

### 14.2.1 Steiner Tree Problem

Given:

1. An undirected graph $G = (V, E)$
2. A cost function $\omega : E \to \mathbb{Q}^+$ [1]
3. A set of terminal vertices $T \subseteq V$

Solve[2]:

$$H^* = (V_H^*, E_H^*) = \arg \min_{H=(V_H,E_H)} \sum_{e \in E_H} \omega(e)$$
$$\text{s.t.} \quad V_H \subseteq V$$
$$E_H \subseteq E$$
$$\text{H connects all } t \in T$$

[1] This is equivalent to assigning a non-negative weight to each edge.
[2] The minimum cost subgraph that connects all terminal nodes in the graph.

**Theorem 14.1** *There exists a primal-dual algorithm for the Steiner tree problem with approximation ratio* $2\left(1-\frac{1}{k}\right)$

**Remark:** There exists an algorithm with improved approximation ratio for steiner tree; however, it is not discussed here.

In order to prove theorem 14.1, we will need two useful concepts: separation and $\delta$ of a separation.

**Definition 14.2** $S$ *separates* $T$ *means* $S \cap T \neq \emptyset \wedge S \cap T \neq T \wedge S \subseteq V$

Definition 14.2 shows when a set $S$ separates $T$ which is useful for thinking about a set of vertices in between some of the terminal nodes. It may be confusing at first glance why a $S$ contains elements of $T$ if it is meant to separate it. Note that if $S$ contains some but not all elements of $T$ then at least some of the elements of $S$ must be added to a steiner tree for $T$.

**Definition 14.3** $\delta(S) = \{e = (v_1, v_2) | e \in E \wedge v_1 \in S \wedge v_2 \notin S\}$ [3]

Meanwhile, $\delta(S)$ can be understood as the set of edges at the boundary of a set of vertices $S$.

### 14.2.2 Primal

$$\begin{aligned} \min & \sum_{e \in E} \omega(e) x_e \\ \text{s.t.} & \sum_{e \in \delta(S)} x_e \geq 1 \quad \forall_{S \subseteq V} S \text{ separate } T \\ & x_e \geq 0 \end{aligned} \tag{14.1}$$

The primal shown in equation 14.1 is the linear relaxation of the straight forward integer programming formulation of the Steiner Tree problem. The integer programming formulation restricts $x_e \in \{0,1\}$. The primal variables $x_i$ represent which edges are selected in the tree. When multiplied with $\omega(e)$ the cost computes the sum of the costs of the edges corresponding to the steiner tree. The constraint can be thought of as ensuring that there exists a path connecting all terminals in terms of sets $S$ that separate $T$. This makes sense when considering that a $S = t$ s.t $t \in T$ separates $T$. Notice that the minimal subgraph connecting all $t \in T$ must be a tree since you could drop an edge in a cycle and keep all components connected.

### 14.2.3 Dual

$$\begin{aligned} \max & \sum_{S : S \text{separates} T} y_s \\ \text{s.t} & \sum_{e \in \delta(S)} y_s \leq \omega(e) \quad \forall e \in E \\ & y_s \geq 0 \end{aligned} \tag{14.2}$$

The dual variable $y_s$ corresponds to the value of a cut $S$ that separates $T$. With this we can see that the constraint could be interpreted as ensuring that every edge must pay for the cuts that it traverses by having a greater or equal cost. Notice that is $c_e = 1 \quad \forall e \in E$ then we can understand the dual as finding the largest collection of edge-disjoint cuts.

[3] The set of all edges with one end point in $S$ and one end point not in $S$.

## 14.3 Algorithm

We will use the following in the algorithm:

- The set of components: $\Psi = \{\{x\} : x \in T\}$
- For each component $c \in \Psi$, $x_c$ is the tree on $C$ found in the algorithm:

  $x_{\{x\}} = \{(\{x\}, \phi) : x \in T\}$.
- Steiner forest initialized to a forest on vertex set $T$ and no edges. At the end $F$ will be a steiner tree.

  $F = (T, \phi)$
- During the algorithm, we would grow components by adding edges and vertices. However, not all the edges added to these components are part of the steiner tree F we return at the end. We only add edges to $F$ when two components merge:

  $(x_c, c \in \Psi) \neq F$

---

**Algorithm 2:** Algorithm to solve Steiner Tree Problem:

---

**input :** $G, T, \omega$
**output:** $F$
$\Psi = \{\{x\} : x \in T\}$
$x_{\{x\}} = \{(\{x\}, \emptyset) : x \in T\}$
$F = (T, \emptyset)$
$y_s = 0 \quad \forall S \subseteq V : S \quad \text{separates} \quad T$
$t = 0$
$M_{\{x\}} = \{\{x\}\}$
**while** $|\Psi| > 1$ **do**
  **while** $\sum_{s:sseparatesT, e \in \delta(S)} y_s \leq x_e$ *is not tight for some new* $e = (u, v) \in E$ **do**
    $\forall c \in \Psi$ increase $y_c \leftarrow y_c + \Delta t$
  **if** $\exists_{C_i, C_j \in \Psi} \quad u \in C_i \wedge v \in C_j$ **then**
    Add $C_i \cup C_j$ to $\Psi$
    Delete $C_i, C_j$ from $\Psi$
    $x_{C_i \cup C_j} = x_{C_i} \cup x_{C_j} + e$
    Let $p$ be a path connecting $F \cap C_i$ and $F \cap C_j$ in $X_{C_i \cup C_j}$
    $F = F \cup p$ ( add edges and vertices of $p$ to $F$
    $M_{c_i \cup c_j} = M_{c_i} \cup M_{c_i} \cup \{C_i \cup C_j\}$
  **if** $u \in C_i$ *for some* $C_i \in \Psi, v \notin C_j$ *for any* $C_j \in \Psi$ **then**
    add $c_i + v$ to $\Psi$
    delete $c_i$ from $\Psi$
    $x_{c_i + v} = x_{c_i} + e$ (add vertex v and edge e to $c_i$)
    $M_{c_i} = M_{c_i} \cup \{c_i + v\}$
  **if** *Neither* **then**
    $t \leftarrow t + \Delta t$
**return** F

---

Algorithm 2 is divided into three parts after initialization. Notice that in the algorithm we initialize a moat variable $M$ and a time variable $t$ that are useful for analysis but not necessary for the computation of a Steiner Tree. After initializing we start a while loop that will end when $|\Psi|$, the number of active separating sets, is reduced to one. The while loop in lines 8 to 10 increases the dual variables by $\Delta t$ until a new constraint is tight. When the constraint becomes tight, it will be tight for some edge $e$ which we will keep track of for the remaining two parts. Next we handle two cases for $e$: either $e$ is between two existing active cuts in $\Psi$ or it is a new edge. If the edge is between two cuts in $\Psi$, we merge them with the necessary bookkeeping. Otherwise we simply add the edge to the component $C_i$ it connects to and do the necessary bookkeeping. Notice that when we merge two sets, part of the bookkeeping ensure that $F$ contains an updated steiner forest.

## 14.4 Analysis

**Lemma 14.4** *At any time $t$, for any $C \in \Psi$, $x_c$ is a tree.*

**Lemma 14.5** *At the end of the algorithm, $F$ is a steiner tree.*

**Lemma 14.6** *At any time $t$, $\{y_s : S$ separates $T\}$ is a feasible dual solution.*

Lemmas 14.4, 14.5 and 14.6 can be proved by induction.

**Lemma 14.7** *At time $t \geq 0$, for $C \in \Psi$, let $F_c$ be the edges of $F$ (at time $t$) with both end points in $C$.*

$$Z(C) = \sum_{s \in M_c} y_s$$
$$Cost(C) = \sum_{e \in F_c} w_e$$

*Then, $Cost(c) \leq 2\,(Z(C) - t)$*

**Proof:** At time $t = 0$, $F = (T, \phi)$, $\Psi = \{\{x\} : x \in T\}$ , $y_s = 0$, $\forall S : S$ separates $T$ Hence, $Cost(C) = 0$, $Z(C) = 0\ \forall C \in \Psi \rightarrow \text{Cost}(C) \leq 2(Z(C) - t)$at $t = 0$

- For ease of exposition, we divide the events into three cases:
  1. $t$ increases by $\Delta t$ and no change in the set of components.
  2. A vertex $v$ is added to some component $C_i$ at time t.
  3. Two components $C_i, C_j$ merge at time t.

Case 1: Set of components $\Psi$ does not change. By induction, $\text{Cost}(C) \geq 2(Z(C) - t)$ at time $t$. $y_c$ increases $\Delta t$ for each $C \in \Psi$. $Z(C) = \sum_{S \in M_C} y_s$ increases by $\Delta t$ since $C \in M_C$. $F$ does not change. Hence, $\text{Cost}(C)$ does not change. Therefore $\text{Cost}(C) \leq 2(Z(C) - t)$ at time $t + \Delta t$.

Case 2: A vertex $v$ is added to $C_i$ ($t$ does not change) $C_i + v$ is added to $\Psi$ and $C_i$ is deleted. By induction, $\text{Cost}(C_i) \leq 2(Z(C_i) - t)$.

$\text{Cost}(C_i + v) = \text{Cost}(C_i)$ since, no edges are added to $F$ and $F_{C_i+v} = F_{C_i}$

$Z(C_i + v) = \sum_{S \in M_{C_i+v}} y_S = \sum_{s \in M_{C_i}} y_S + y_{C_i+v}$

when $v$ is added to $C_i$, $y_{C_i+v} = 0$. Hence, $Z_{C_i+v} = Z_{C_i}$, $\text{Cost}(C_i + v) = \text{Cost}(C_i)$.

$\rightarrow \quad \text{Cost}(C_i + v) \leq 2(Z(C_i + v) - t)$

Case 3: Two components $C_i, C_j \in \Psi$ merge ($t$ does not change).

$C_i \cup C_j$ is added to $\Psi$, $C_i$, $C_j$ are deleted from $\Psi$. By induction,

$\text{Cost\_old}(C_i) \leq 2(Z(C_i) - t)$

$\text{Cost\_old}(C_j) \leq 2(Z(C_j) - t)$

Where _old denotes before merging.

$\text{Cost\_new}(C_i \cup C_j) \leq \text{Cost\_old}(C_i) + \text{Cost\_old}(c_j) + 2t \leq 2(Z(C_i) + Z(C_j) - t)$

$Z(C_i \cup C_j) = \sum_{s \in M_{C_i \cup C_j}} y_s = \sum_{s \in M_{c_i}} y_s + \sum_{s \in M_{c_j}} y_s + y_{c_i \cup c_j}$

Since, $C_i \cup C_j$ is just added to $M_{C_i \cup C_j}$, $y_{C_i \cup C_j} = 0$.

Hence,

$Z_{C_i \cup C_j} = Z(C_i) + Z(C_j)$

$\rightarrow \quad \text{Cost\_new}(C_i \cup C_j) \leq 2(Z(C_i \cup C_j) - t)$ end of Lemma 4s proof.

■

**Theorem 14.8** *Optimal Steiner-tree Cost:*

$$\sum_{e \in E(F)} w_e \leq 2(1 - \frac{1}{|T|})$$

**Proof:** At the end of the algorithm ($t = \text{end}$) let the component in $\Psi$ be $C^*$.
Then, at $t = t_{\text{end}}$,

$$\text{Cost}(C^*) = \sum_{e \in F_{C^*}} w_e = \sum_{e \in E(F)} w_e$$
$$Z(c^*) = \sum_{s \in M_{C^*}} y_s = \sum_{s:s \text{ separate } T} y_s$$

Hence, by lemma 4,

$$\sum_{e \in E(F)} w_e \leq 2 \left( \sum_{s:s \text{ separates } T} y_s - t_{\text{end}} \right)$$

At any given time $t$, the number of components in $\Psi$ is at most $|T|$. Hence, when $t$ increases by $\Delta t$, $\sum_{s: \text{ separates } T}$ increases by at most $|T|\Delta T$

$\rightarrow \quad \sum_{s:s \text{ separates } T} y_s \leq |T| t_{\text{end}}$ or $t_{end} \geq \frac{\sum_{s:s \text{ separates } T} y - s}{|T|}$ substituting in 1.

$\rightarrow \sum_{e \in E(F)} w_e \leq 2(1 - \frac{1}{|T|} \sum_{S:S \text{ separates } T} y_s)$

Since $\{y_S : S \text{separates} T\}$ is a feasible dual solution, $\sum_{s:s\text{separates}T} y_s \leq$ optimal-dual-value. By strong duality, optimal-dual-value = optimal-primal-value. Since, primal is a relaxation of the steiner tree probllem, optimal-primal-value $\leq$ optimal-steiner-tree-cost. Combining, all four inequalities, we get $\sum_{e \in E(F)} w_e \leq 2(1 - \frac{1}{|T|})$optimal-steiner-tree-cost.

Therefore, we get a $2(1 - \frac{1}{|T|})$ approximation algorithm for the steiner tree problem. ■

# References

[1] Ravi, R. "A primal-dual approximation algorithm for the Steiner forest problem." Information processing letters 50.4 (1994): 185-189.