COMPSCI 638: Graph Algorithms October 9, 2019

Lecture 13

Lecturer: Debmalya Panigrahi *Scribe: Kevin Sun*

1 Overview

In the Lecture 12, we gave a geometric description of a primal-dual algorithm for the Steiner forest problem. In this lecture, we formally state this algorithm and prove that it is a 2-approximation. We then apply this algorithm to the Generalized Steiner Forest problem.

2 A Primal-Dual Algorithm for Steiner Forest

Recall the Steiner forest problem: we are given an undirected graph $G = (V, E)$, where each edge e has cost $c(e) \geq 0$, and a set of k vertex pairs (s_i, t_i) for $i = 1, \ldots, k$ known as *terminal pairs*. Our goal is to find a minimum-cost subset of edges that connects every (s_i, t_i) pair. In this section, we present and analyze an algorithm for this problem given by Agrawal, Klein, and Ravi [AKR95] and simplified and generalized by Goemans and Williamson [GW95].

Let $\mathcal{S}$ denote the subsets of V that separate at least one (s_i, t_i) pair. For any $S \subset V$, we let $\delta(S)$ denote the set of edges with exactly one endpoint in S. Then the primal linear programming relaxation of the Steiner forest problem is the following:

$$\text{(P): } \min \sum_{e \in E} c(e) x_e$$
$$\sum_{e \in \delta(S)} x_e \geq 1 \quad \forall S \in \mathcal{S}$$
$$x_e \geq 0 \quad \forall e \in E,$$

Our algorithm heavily relies on the dual, given below:

$$\text{(D): } \max \sum_{S \in \mathcal{S}} y_S$$
$$\sum_{S : e \in \delta(S)} y_S \leq c(e) \quad \forall e \in E$$
$$y_S \geq 0 \quad \forall S \in \mathcal{S}$$

Our algorithm builds a solution by increasing the dual variables y_S for every $S \in \mathcal{A}$, where $\mathcal{A}$ is initially the set of all terminal vertices (as singleton sets). Geometrically, this step corresponds to expanding balls/moats surrounding the terminals. We say a dual constraint is *tight* when it is satisfied with equality; when this happens, the corresponding edge is completely contained in a moat and we add it to our solution F.

Note that when dual constraint becomes tight, we never increase the y_S values in that constraint again, so our dual solution is always feasible. Furthermore, whenever F is modified, we update $\mathcal{A}$ so that it contains the set of connected components of (V, F) in $\mathcal{S}$.

Finally, once all components are inactive (i.e., $\mathcal{A}$ is empty) and F contains the set of edges corresponding to tight dual inequalities, we perform reverse-delete on F. In reverse order of addition to F, we delete an edge if the resulting set of edges remains a feasible Steiner forest.

Algorithm 1 A primal-dual algorithm for Steiner Forest

```
 1: Initialize: $F = \emptyset, y_S = 0 \quad \forall S \subseteq V$.
 2: Let $\mathcal{S}$ denote the subsets of $V$ that separate at least one $(s_i, t_i)$ pair.
 3: while $F$ is not a Steiner forest do
 4:     Set $\mathcal{A}$ as the set of connected components of $(V, F)$ in $\mathcal{S}$.
 5:     Increase $y_S$ for every $S \in \mathcal{A}$ uniformly until $\sum_{S:e\in\delta(S)} y_S = c(e)$ for some $e \in E$.
 6:     Add $e$ to $F$. (Note: the $y_S$ corresponding to this constraint will never increase again.)
 7: $F' = F$
 8: for $e \in F'$ in reverse order of addition to $F$ do
 9:     if $F' \setminus \{e\}$ is a Steiner forest then
10:         Remove $e$ from $F'$.
11: return $F'$
```

Theorem 1. *Algorithm 1 is a 2-approximation algorithm for the Steiner forest problem.*

Before proving this theorem, we state a useful lemma whose proof we defer for now.

Lemma 2. *Let F' denote the final output of Algorithm 1 and let $\mathcal{A}_t$ denote the set $\mathcal{A}$ at the beginning of some iteration t. Then the following inequality holds:*

$$\sum_{S\in\mathcal{A}_t} \left|F' \cap \delta(S)\right| \leq 2|\mathcal{A}_t|.$$

Proof of Theorem 1. Let F^* denote an optimal Steiner forest; we want to show $c(F') \leq 2c(F^*)$. Notice that edges in F' correspond to tight dual constraints, so we have

$$c(F') = \sum_{e\in F'} \sum_{S:e\in\delta(S)} y_S = \sum_S y_S \left|F' \cap \delta(S)\right|.$$

By weak duality, we also know $\sum_S y_S \leq c(F^*)$, so it would be sufficient for us to show $|F \cap \delta(S)| \leq 2$ for every $S \in \mathcal{S}$, but this is not necessarily true. (For example, consider a star centered at $s_1 = \cdots = s_k$ whose other endpoints are the t_i.) So instead, we will prove

$$\sum_S y_S \cdot \left|F' \cap \delta(S)\right| \leq 2\sum_S y_S \tag{1}$$

by showing that in each iteration of the algorithm, the increase in the left-hand side (LHS) of (1) is at most twice the increase of the right-hand side (RHS). Note that in our analysis, F' refers to the final output of Algorithm 1, but we consider the changes in y_S.

Consider some iteration t of the algorithm, and let $\mathcal{A}_t$ denote the set $\mathcal{A}$ at the beginning of the iteration. Let ϵ denote the amount by which y_S increased in this iteration for every $S \in \mathcal{A}_t$. Then the LHS of (1) increases by $\epsilon \sum_{S\in\mathcal{A}_t} |F' \cap \delta(S)|$ and the RHS of (1) increases by $2\epsilon|\mathcal{A}_t|$. By Lemma 2, this implies that (1) holds throughout the entire algorithm. ☐

Proof of Lemma 2. Let F_t denote the set F at the beginning of iteration t, so $\mathcal{A}_t$ denotes the connected components of (V, F_t) that cut at least one (s_i, t_i) pair. Notice that (V, F_t) is a forest because F is initially a forest, and each edge we add joins two existing connected components. In fact, this implies that (V, F) is acyclic throughout the entire algorithm.

Thus, we can now consider the following graph H: each vertex u_T corresponds to a tree T in (V, F_t), and $\{u_{T_1}, u_{T_2}\}$ is an edge if some edge in $F' \setminus F_t$ joins T_1 and T_2. Observe that H is also a forest because, as we showed above, (V, F) never contains a cycle.

The *active* vertices of H are the ones that correspond to trees in $\mathcal{A}_t$; the remaining vertices of H are *non-active*. We want to show that the average degree of active vertices in H is at most 2.

We claim that every leaf of H is active. For contradiction, suppose u_{T_1} is a non-active leaf of H incident to some vertex u_{T_2}. Then there exists an edge e in $F' \setminus F_t$ that joins the trees T_1, T_2 in (V, F_t). Furthermore, e was not deleted by reverse-delete, which means $F' \setminus \{e\}$ is not feasible. However, T_1 is non-active, which means it does not separate any (s_i, t_i) pair. This means $F' \setminus \{e\}$ is actually feasible, contradicting the previous statement.

Thus, every vertex with degree 1 in H is active, so the average degree of non-active vertices is at least 2. The handshaking lemma tells us that the average degree of all vertices in any tree is at most 2, so the average degree of active vertices is at most 2. □

3 Generalized Steiner Forest

In this section, we apply Algorithm 1 to solve the Generalized Steiner Forest (GSF) problem: we are still given an undirected graph $G = (V, E)$, each edge e has cost $c(e) \geq 0$, and a set of k terminal pairs (s_i, t_i) for $i = 1, \ldots, k$. In addition, each (s_i, t_i) pair has an integer *demand* quantity $r_i \geq 1$, and in our solution $F \subseteq E$, the minimum s_i-t_i cut value must be at least r_i for every $i \in \{1, \ldots, k\}$. In the version we consider, F is allowed to be a multiset, that is, F can include multiple copies of any edge.

The primal LP relaxation of the GSF problem is the following:

$$\text{(P-G):}\ \min \sum_{e \in E} c(e) x_e$$
$$\sum_{e \in \delta(S)} x_e \geq f(S) \quad \forall S \subseteq V$$
$$x_e \geq 0 \quad \forall e \in E,$$

where $f : 2^V \to \mathbb{Z}_{\geq 0}$ is defined as the maximum demand crossing S. In other words, if $\mathcal{S}_i$ denotes the subsets of V that separate s_i and t_i, then $f(S) = \max_{i: S \in \mathcal{S}_i} r_i$. We let (IP-G) denote the integer version of (P-G), which has the additional constraint $x_e \in \{0, 1\}$ for every $e \in E$.

3.1 Weakly Supermodular Functions

Notice that the LP relaxation of the Steiner forest problem (given in Section 2) is (P-G) with $f(S) = 1$ if S cuts at least one (s_i, t_i) pair and 0 otherwise. Thus, Algorithm 1 is a special case of a more general algorithm given by Goemans and Williamson [GW95], who considered an entire class of functions $f : 2^V \to \{0, 1\}$ known as *proper functions*.

Definition 1. *A function $f : 2^V \to \{0, 1\}$ is* proper *if $f(V) = 0$ and it satisfies the following:*

1. *Symmetry: $f(A) = f(V \setminus A) \quad \forall A \subseteq V$.*

2. *Maximality: if $A \cap B = \emptyset$, then $f(A \cup B) \leq \max\{f(A), f(B)\}$.*

More specifically, Algorithm 1 gives a 2-approximation for the Steiner forest problem, which is (IP-G) on a particular proper function f. The algorithm of Goemans and Williamson [GW95] (which we now call the GW algorithm) gives a 2-approximation for (IP-G) on *any* proper function f. Now we consider an even larger (proof omitted) class of functions.

Definition 2. *Let $f : 2^V \to \mathbb{Z}$ be a function satisfying $f(\emptyset) = f(V) = 0$. Then f is* weakly supermodular *if for any $A, B \subseteq V$,*

$$f(A) + f(B) \leq \max\{f(A \setminus B) + f(B \setminus A), f(A \cup B) + f(A \cap B)\}.$$

We note that the GSF demand function $f(S) = \max_{i:S \in \mathcal{S}_i} r_i$ is weakly supermodular; the proof is fairly straightforward. Weakly supermodular functions are closely related to a well-known class of functions known as submodular functions, which formally capture the notion of diminishing returns. A notable example of a submodular function is the cut function $\delta : 2^V \to \mathbb{Z}_{\geq 0}$, where $\delta(S)$ is defined as the number of edges with exactly one endpoint in S.

Fact 3. *The cut function δ satisfies the following inequalities for all $A, B \subseteq V$:*

$$f(A \setminus B) + f(B \setminus A) \leq f(A) + f(B)$$
$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$$

We are now ready to present and analyze an algorithm for the GST problem.

3.2 Primal-Dual Algorithms for GSF

Our algorithm is based on the following intuition: the GSF demand function f_r defined as $f_r(S) = \max_{i:S \in \mathcal{S}_i} r_i$ is weakly supermodular. The algorithm will define and solve a sequence of Steiner forest instances by applying the GW algorithm on (IP-G), and in each iteration, we add the resulting edges to our overall solution. We then update the demand function to account for the new solution, and from Fact 3, this "residual" demand function is still supermodular, so we can repeat this process until all demands are satisfied.

Algorithm 2 An primal-dual approach for GSF with demand function f_r

```
1: Initialize: $F = \emptyset, f_{res} = f_r$.
2: while $\exists S$ such that $f_{res}(S) \neq 0$ do
3:     Define $g : 2^V \to \{0, 1\}$ as follows: $g(S) = 1$ if $f_{res}(S) \geq 1$ and 0 otherwise.
4:     Run the GW algorithm on (IP-G) with demand function $g$ to obtain $F_{res} \subseteq E$.
5:     Add $F_{res}$ to $F$.
6:     For each $S \subseteq V$, set $f_{res}(S) = \max\{0, f_{res}(S) - |F_{res} \cap \delta(S)|\}$.
7: return $F$
```

As we will see, a simple modification to Algorithm 2 yields a much better approximation guarantee, but before seeing this modification, we first analyze this simpler version. Let r_{max} denote the maximum demand quantity of any (s_i, t_i) pair.

Theorem 4. *Algorithm 2 is a $2r_{max}$-approximation algorithm for the GSF problem.*

Proof. We first show that Algorithm 2 terminates in at most r_{max} iterations. Consider any iteration, and let S be a subset of V that satisfies $g(S) = 1$. Since F_{res} is a is a feasible Steiner forest with demand function g, it must include at least one edge of $\delta(S)$, which means $f_{res}(S)$ decreases by at least one at the end of the iteration. Thus, the total number of iterations is at most r_{max}.

Let OPT denote the cost of an optimal solution, and for any iteration i, let OPT_i denote the optimal solution cost of the (IP-G) Steiner forest instance we solve in iteration i. Since the demand g in any iteration is at most f_{res}, we have $OPT_i \leq OPT$.

Finally, let ALG_i denote the cost of edges added in iteration i. Since the GW algorithm is a 2-approximation of (IP-G), we have $ALG_i \leq 2 \cdot OPT_i \leq 2 \cdot OPT$. Summing across the (at most) r_{max} iterations, we see that the total cost is at most $2r_{max} \cdot OPT$. □

We now give the modification of Algorithm 2. The intuition is the following: instead of simultaneously satisfying demand for *all* S such that $f_{res(S)} \geq 1$, we start with the subsets that satisfy $f_{res(S)} = r_{max}$ and work our way down. In the first iteration, we can reduce the maximum demand from r_{max} to $r_{max} - 1$, so in the next iteration, we can focus on the subsets that satisfy $f_{res(S)} = r_{max} - 1$. We repeat this process until all demands are satisfied.

Algorithm 3 An improved version of Algorithm 2

```
1: Initialize: $F = \emptyset, f_{res} = f_r$.
2: for $i = r_{max}, r_{max} - 1, \ldots, 1$ do
3:     Define $g_i : 2^V \to \{0,1\}$ as follows: $g_i(S) = 1$ if $f_{res}(S) = i$ and 0 otherwise.
4:     Run the GW algorithm on (IP-G) with demand function $g_i$ to obtain $F_i \subseteq E$.
5:     Add $F_i$ to $F$.
6:     For each $S \subseteq V$, set $f_{res}(S) = \max\{0, f_{res}(S) - |F_i \cap \delta(S)|\}$.
7: return $F$
```

Theorem 5. *Algorithm 3 is a $2H_{r_{max}}$-approximation for the GSF problem, where $H_n \approx \log n$ denotes the n-th Harmonic number.*

Proof. As discussed above, since F_i is a feasible solution for the Steiner forest problem with demand function g_i, each iteration decreases the demand of any S initially satisfying $f_{res(S)} = i$ to $i - 1$. Thus, running the algorithm for iterations $i = r_{max}, \ldots, 1$ results in a feasible solution.

Let x^* denote an optimal solution to the Steiner forest integer program (IP-G), and let OPT denote its cost. We claim that in any iteration i, x^*/i is a feasible solution for the Steiner forest instance with demand g_i. Let S be any subset with demand in g_i (i.e., $g_i(S) = 1$), so in this iteration, $f_{res}(S) = i$. Since x^* is feasible, we have

$$\sum_{e \in \delta(S)} x_e^* \geq f_r(S) \geq f_{res}(S) = i,$$

which means x^*/i satisfies the demand $g_i(S) = 1$.

Thus, letting OPT_i denote the cost of the optimal solution of (IP-G) in iteration i, we have $OPT_i \leq OPT/i$. Since the GW algorithm is a 2-approximation, the cost of F_i is $ALG_i \leq 2 \cdot OPT_i$. Putting this together, our total cost ALG satisfies

$$ALG = \sum_{i=1}^{r_{max}} ALG_i \leq 2 \sum_{i=1}^{r_{max}} OPT_i \leq 2 \cdot OPT \sum_{i=1}^{r_{max}} \frac{1}{i} = 2H_{r_{max}} \cdot OPT. \qquad \square$$

4 Summary

In this lecture, we presented a primal-dual algorithm for the Steiner forest problem. We also introduced the notion of proper and weakly supermodular functions, which we will further study in the next lecture. Finally, we saw how to apply the Steiner forest algorithm to solve the generalized Steiner forest problem.

References

[AKR95] Ajit Agrawal, Philip Klein, and R Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

[GW95] Michel X Goemans and David P Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.