Lecture 18

Lecturer: Michel X. Goemans *Scribe: Ahmed Ismail and Johnny Chen*

In the previous lecture, we discussed the Bar-Yehuda and Even 2-approximation algorithm for the vertex cover problem, and introduced the Generalized Steiner tree problem, and outlined a 2-approximation algorithm of Goemans and Williamson which formulates the problem using duality which constructs both an integer solution and a dual solution of a relaxed version of the problem, and began the construction of the proof of correctness of the algorithm.

1 The Generalized Steiner tree problem

Problem 1 (Generalized Steiner tree problem) *Given a graph $G = (V, E)$ with costs $c(e) \geq 0, \forall e \in E$, and a set of vertex pairs $T \subseteq V \times V$, find a subgraph F of minimum cost such that $\forall (s,t) \in T$, s and t are connected in F.*

As a linear program, this problem can be formulated as trying to determine the optimal solution OPT for

$$\min \sum_{e \in E} c(e)\, x(e), \tag{1}$$

such that

$$\sum_{e \in \delta(S)} x(e) \geq 1, S \in \mathcal{F}, \tag{2}$$

$$x(e) \in \{0,1\} \tag{3}$$

where

$$\mathcal{F} = \{S \subseteq V : \exists (s,t) \in T; |S \cap \{s,t\}| = 1\}. \tag{4}$$

Rather than deal with this complicated integer linear program, Goemans and Williamson relax the problem by replacing the integer constraint (3) with a nonnegativity constraint, and therefore look for the solution LB which satisfies (1) subject to the constraints (2) as well as

$$x(e) \geq 0, \forall e.$$

Goemans and Williamson also consider the corresponding dual problem,

$$\max \sum_{S \in \mathcal{F}} y_S \tag{5}$$

such that

$$\sum_{S \in \mathcal{F}: e \in \delta(S)} y_S \leq c(e), \forall e, \tag{6}$$

$$y_S \geq 0. \tag{7}$$

Considering (1) and (5) together, we can obtain both a feasible set of edges F as well as a set of y_S's satisfying (6), such that

$$\sum_{e \in F} c(e) \leq 2 \sum_{S \in \mathcal{F}} y_S \leq 2OPT. \tag{8}$$

The details of the algorithm to compute F and the y_S's were provided in the previous lecture.

2 Eliminating the y_S's

Our aim is to establish that (8) is valid for the Goemans-Williamson algorithm. However, before we resume our proof of its correctness, we note that although we keep track of the y_S's for the purpose of establishing correctness, we do not actually use the y_S's as variables when implementing this algorithm. Instead, we choose to keep track of $d(i)$, which is defined as

$$d(i) = \sum_{S:S\ni i} y_S,$$

for every vertex i. Then, for an edge $(i,j) \in \delta(S)$, where i and j belong to different connected components of F,

$$\sum_{S:(i,j)\in\delta(S)} y_S = d(i) + d(j).$$

The goal of the dual program is to maximize each y_S while still obeying the constraints implied by (6); if we choose to ignore the individual y_S's, we can try to minimize $c_e - d(i) - d(j)$ instead.

Thus, our algorithm proceeds as follows.

Initialize $[F = \emptyset;\ \mathcal{C} = \{\{i\} : i \in V\};\ d(i) \leftarrow 0\ \forall i \in V;\ k \leftarrow 1;\ (y_S \leftarrow 0\ \forall S \in \mathcal{F})]$;

While $\mathcal{F} \cap \mathcal{C} \neq \emptyset$ do:

(see observation in notes below)

$$\epsilon = \min\left(\min_{i\in C_p, j\in C_q, p\neq q, C_p, C_q\in\mathcal{F}} \frac{c_{ij} - d(i) - d(j)}{2},\ \min_{i\in C_p, j\in C_q, p\neq q, C_p\in\mathcal{F}, C_q\notin\mathcal{F}} c_{ij} - d(i) - d(j)\right);$$

Let $e_k \leftarrow (i,j)$ attain this minimum.

For $C \in \mathcal{C} \cap \mathcal{F}$:

For $i \in C$: $d(i) \leftarrow d(i) + \epsilon$;

$F \leftarrow F \cup \{e_k\}$;

$k \leftarrow k + 1$;

Update the connected components $\mathcal{C}$;

$F' \leftarrow F$;

For $\ell \leftarrow k-1$ down to 1:

if $F' - \{e_\ell\}$ is feasible then $F' \leftarrow F' - \{e_\ell\}$; (delete step)

Output F'.

Note that the delete step in the algorithm, as written above, is easier for the purposes of the proof. However, we do not need to perform the delete in this manner: although we can cycle through the edges in the order in which we added them, we can also attempt to remove them *in any order.* Either route will yield the unique solution which has paths between our pairs of vertices s's and t's, which will be the same result as using the condition

$$F' = \{e \in F : F \setminus \{e\} \text{ is not feasible}\}.$$

3 Correctness of the algorithm

Our goal is to establish the following result.

Theorem 1 *The algorithm of Goemans and Williamson returns a feasible set of edges F and a set of y_S's such that*

$$\sum_{e\in F} c(e) \leq 2 \sum_{S\in\mathcal{F}} y_S \leq 2OPT. \tag{9}$$

In our previous lecture, we established the following lemma.

Lemma 2 *If $\mathcal{C} \cap \mathcal{F} = \emptyset$, then F is feasible.*

Thus, since by construction the algorithm only terminates if $\mathcal{C} \cap \mathcal{F} = \emptyset$, we know that any resulting F obtained is feasible. We thus focus on establishing the inequality (9). Let us first define two sets O and E as follows.

Definition 1 *Let $O = \mathcal{C} \cap \mathcal{F}$ and $E = \mathcal{C} \cap \bar{\mathcal{F}}$, so that $\mathcal{C} = E \cup O$.*

We now construct a graph $H = (\mathcal{C}, E(H))$ by shrinking all the connected components into vertices. In this construction, the set $E(H)$ is

$$E(H) = \{(C_p, C_q) : \text{if}\, \exists\, (i,j) \in F' : i \in C_p, j \in C_q\}.$$

Since H has no cycles, it is a forest, and therefore

$$\sum_{v\in O} d_v + \sum_{v\in E} d_v \leq 2|O| + 2|E| - 2,$$

where d_v is the degree of vertex v in the forest H. If H is a single tree, then equality holds. We can ignore all vertices v such that $d_v = 0$. We now want to prove the following result.

Lemma 3 *There does not exist $v \in E$ such that $d_v = 1$.*

Proof: Suppose that $\exists v \in E : d_v = 1$,where v corresponds to the connected component C, and let e_k be the unique edge corresponding to this vertex. Since we could not remove e_k during the delete step, there exists an edge $(s,t) \in T$ such that $s \in C$, $t \notin C$. However, this implies that

$$C \in F \Rightarrow v \in O \Rightarrow v \notin E,$$

which is clearly a contradiction. Thus, there does not exist $v \in E$ such that $d_v = 1$. □

This lemma lets us say that

$$\sum_{v\in O} d_v + 2|E| \leq \sum_{v\in O} d_v + \sum_{v\in E} d_v \leq 2|O| + 2|E| - 2. \tag{10}$$

Cancelling like terms from the outside of (10) gives us

$$\sum_{v\in O} d_v \leq 2|O| - 2. \tag{11}$$

Now, as we know that

$$\sum_{e\in F'} c_e = \sum_{e\in F'} \sum_{S: e\in\delta(S)} y_S,$$

we can exchange the order of summations to say that

$$\sum_{e\in F'} c_e = \sum_{S\in\mathcal{F}} y_S \cdot |F' \cap \delta(S)|.$$

Now it remains to prove the following lemma.

Lemma 4 *During the execution of the algorithm,*

$$\sum_{S \in \mathcal{F}} y_S \cdot |\delta(S) \cap F'| \leq 2 \sum_{S \in \mathcal{F}} y_S. \tag{12}$$

Proof: The proof of this is carried out by induction on k. The base step is trivial, since we initialize $y_S = 0$ at the start of the algorithm.

Then, during an iteration, y_S increases by an amount ϵ for each $S \in \mathcal{F} \cap \mathcal{C}$, so the right-hand side of the inequality increases by $2\epsilon |\mathcal{F} \cap \mathcal{C}|$. By the same reasoning, the left-hand side increases by $\sum_{S \in \mathcal{F} \cap \mathcal{C}} \epsilon |\delta(S) \cap F'|$. Thus, we want to show that

$$\sum_{S \in \mathcal{F} \cap \mathcal{C}} |\delta(S) \cap F'| \leq 2 |\mathcal{F} \cap \mathcal{C}|.$$

However, we know by construction that $\mathcal{F} \cap \mathcal{C} = O$, and when $v \in O$, $|\delta(S) \cap F'|$ is just the degree d_v of node v in H. As a result, we have

$$\sum_{v \in O} d_v \leq 2 |O|,$$

which is a weaker form of the inequality (11). This completes the proof of (12), and with it, of the correctness of the algorithm. □

4 Implementation and generalizations

This approach also works and gives a bound for problems such as

$$\min \sum c_e x_e : \sum_{e \in \delta(S)} x_e \geq 1, x_e \in \{0, 1\}, \forall S \in \mathcal{F}$$

for other collections of sets $\mathcal{F}$, $\mathcal{F} = \{S : |S| \text{ odd}\}$ (for perfect matchings) or $\mathcal{F} = \{S : |S \cap A| \neq |S \cap B|\}$.

With the following implementation considerations, the algorithm runs in $O(n^2 \; log \; n)$ time.

- The Union-Find data structure can be used to update connected components (in roughly linear time).
- Instead of using a double loop to find ϵ (which would take m^2 time), we can create a "time-axis," keep track of the time when each edge becomes tight, presuming nothing else in the graph changes.
- Each iteration (there are n of them) can be done with fewer than n priority queue operations, since modifying a connected component necessitates changing the time of up to n edges. Thus there are at most $O(n^2)$ priority queue operations, and the total running time will be $O(n^2 \log n)$.