IOE 691: Approximation Algorithms Date: 01/23/2017

# Lecture Notes: Min Degree Spanning Tree (Local Search)

Instructor: Viswanath Nagarajan Scribe: Qingya Liu

## 1 Minimum degree spanning tree problem

**Definition 1.1.** *The **degree of a spanning tree** $T$ is the maximum degree of all vertices $v \in V$ of the spanning tree, i.e.* $\deg(T) = \max_{v \in V} d_T(v)$.

For example, consider the tree in figure 1 which has degree 5.

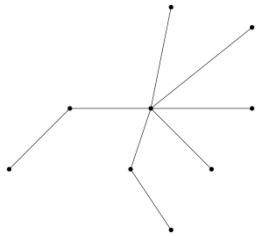

Figure 1: Diagram of Spanning Tree with $\deg(T) = 5$.

The minimum degree spanning tree problem involves computing a spanning tree in a given graph $G = (V, E)$ with minimum degree. The problem has many applications. For example, in telecommunication network design, the bottle-neck is often the congestion control at the single router (node) with many connecting edges. Reducing the maximum degree of all vertices in the network can be described as a minimum degree spanning tree problem. This problem is NP-hard by a reduction from the Hamilton path problem.

We introduce a local search algorithm to find the minimum degree spanning tree. Given a graph $G = (V, E)$ and a spanning tree $T$ (Figure 2), suppose $u \in V(G)$ is a vertex with high degree. To reduce the degree $d_T(u)$, we choose two other vertices $v, w \in V(G)$ such that adding the edge $(v, w)$ to $T$ creates a cycle containing $u$. Then we can remove an edge $(u, u')$ from $T$ to obtain another spanning tree with smaller degree for $u$ The algorithmic description is below.

**Algorithm 1:** Local Search Alg for Min Deg Spanning Tree

**Initialization:** let $T$ be any spanning tree.
Repeat the following as long as possible:
**Local Move:** pick $u, v, w \in V$ s.t.
$d_T(u) \geq \deg(T) - \log_2 n$;
$u$ lies on the $v - w$ path in $T$;
$(v, w) \in E$;
$d_T(v) \leq d_T(u) - 2$ and $d_T(w) \leq d_T(u) - 2$;
**Update:** $T \leftarrow S = (T \cup \{(v, w)\}) \setminus \{(u, u')\}$ where $u'$ is a neighbor of $u$ in the cycle of $T \cup \{(v, w)\}$

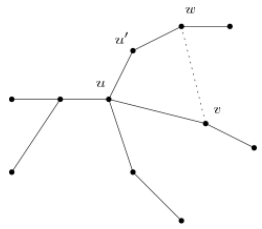


Figure 2: Example of a local move for Min Degree Spanning Tree

The new degrees of the spanning tree $S$ satisfy

$$\begin{aligned} d_S(u) &= d_T(u) - 1 \\ d_S(v) &= d_T(v) + 1 \\ d_S(w) &= d_T(w) + 1 \\ d_S(x) &\leq d_T(x),\ \forall x \in V \setminus \{u, v, w\} \end{aligned}$$

Note that the algorithm is only performed when

$$d_T(u) \geq 2 + d_T(v),\ d_T(u) \geq 2 + d_T(w),\ d_T(u) \geq \deg(T) - L$$

where $L = \log_2 n$. So $\deg(S) \leq \deg(T)$ after any local move.

### 1.1 Bounding the number of iterations

We first show that the running time of this algorithm is polynomial. Each iteration (local move) clearly takes polynomial time. So it is sufficient to bound the number of iterations. We do not know if the objective value of the problem will improve after each iteration, but we can define the **potential function** as a proxy for objective value $\deg(T)$

$$\phi(T) = \sum_{u \in V} 3^{d_T(u)}$$

The potential function comes from the idea that when dealing with non-differentiable functions like max we can introduce the soft-max function where

$$\max(d_1, \cdots, d_n) \approx \ln(\sum e^{d_i})$$

To bound the number of iteration, we first prove the following lemma.

**Lemma 1.1.** *$\phi(S) - \phi(T) \leq -\rho\phi(T)$, where $\rho = \frac{2}{27n^3}$*

*Proof.* Let $d(\cdot)$ denote the degrees in tree $T$. Consider vertices $u, v, w$

$$\begin{aligned} &\text{Degree of } u \Rightarrow 3^{d_S(u)} - 3^{d(u)} = 3^{d(u)-1} - 3^{d(u)} = -\frac{2}{3} \cdot 3^{d(u)} \\ &\text{Degree of } v \Rightarrow 3^{d_S(v)} - 3^{d(v)} \leq 3^{d(v)+1} - 3^{d(v)} = 2 \cdot 3^{d(v)} \leq 2 \cdot 3^{d(u)-2} \\ &\text{Degree of } w \Rightarrow 3^{d_S(w)} - 3^{d(w)} \leq 3^{d(w)+1} - 3^{d(w)} = 2 \cdot 3^{d(w)} \leq 2 \cdot 3^{d(u)-2} \end{aligned}$$

The total change in $\phi$ due to these three vertices would be $-\frac{2}{9} \cdot 3^{d(u)}$. By plugging in $L = \log_2 n$, we have

$$\phi(T) \leq n \cdot 3^{\deg(T)} \leq n \cdot 3^{d(u)+L} \leq n \cdot 3^{d(u)} \cdot n^{1.5}$$

Since the degrees of $V \setminus \{u, v, w\}$ can only decrease, we can bound $\phi(S) - \phi(T)$ by the total change due to their degrees.

$$\phi(S) - \phi(T) \leq -\frac{2}{9} \cdot 3^{d(u)} \leq -\frac{2}{9} \cdot 3^{\deg(T)-L} \leq -\frac{2}{27n^3}\phi(T)$$

□

Suppose the algorithm runs for $k$ iterations, with spanning trees $T_1 \to T_2 \cdots \to T_k$. First observe that $3n \leq \phi(T) \leq n3^n$ for any spanning tree $T$. Then by Lemma 1.1,

$$\phi_k - \phi_{k-1} \leq -\rho\phi_{k-1} \Rightarrow \phi_k \leq (1-\rho)\phi_{k-1} \leq (1-\rho)^{k-1}\phi_1$$

$$\implies \quad 3n \leq \phi_k \leq (1-\rho)^{k-1} \cdot n3^n$$

$$\implies \quad k \leq 1 + \frac{O(n)}{\log(1-\rho)}$$

$$\implies \quad \text{Number of iterations} \leq \frac{n}{\log(1-\rho)} = O(n/\rho)$$

The last equality uses the fact that $\log_2(1+x) \geq x$ for $x \in [0, 1]$.

## 1.2 Approximation ratio

Let $ALG$ denote the degree of the algorithm's spanning tree and $OPT$ the optimal value. Here we will show the following.

**Theorem 1.1.** $ALG \leq 2OPT + \log_2 n$

The main idea is the following lower bound on $OPT$.

> Consider any partition of $V$ into $k$ parts, and $S \subseteq V$ such that every edge crossing the partition has at least one end-point in $S$. Then OPT $\geq \frac{k-1}{|S|}$.

To see this, note that any spanning tree must have at least $k-1$ edges crossing the partition since the spanning tree needs to connect all $k$ parts.

Let $S_i$ denote the vertices with $d_T(v) \geq i$, then $V = S_1 \supseteq S_2 \supseteq S_3 \supseteq \cdots \supseteq S_{\deg(T)} \neq \emptyset$. Observe that there is some $i \leq \deg(T) - L$ with $|S_{i-1}| \leq 2|S_i|$. Suppose not: then we have $|S_{\deg(T)-L}| > 2^L |S_{\deg(T)}| \geq n$ which is a contradiction, since there are only $n$ nodes.

We will now apply the above lower bound. Remove all edges $F$ in $T$ adjacent to vertices of $S_i$ to obtain a partition of $V$ into some number $k$ parts (each connected component in $T \setminus F$ is a part). By definition of $S_i$ we know that $F$ contains at least $i$ edges incident to each vertex of $S_i$. Moreover, as $F \subseteq T$ the number of edges in $F$ with both end-points in $S_i$ is at most $|S_i| - 1$ (otherwise $F$ and hence $T$ would contain a cycle!). So the number of edges $|F| \geq i|S_i| - |S_i| + 1$. This means that the number $k$ of connected components in $T \setminus F$ is $|F| + 1 \geq i|S_i| - |S_i| + 2$.

Each edge in $G$ connecting distinct parts in the above partition has to be

1. an edge of $T$ incident on $S_i$, or

2. an edge that creates a cycle in $T$ containing some node in $S_i$. Since tree $T$ is locally optimal, at least one endpoint of each such edge has degree at least $i-1$, i.e. it is in $S_{i-1}$.

In either case, at least one end-point of each edge crossing the partition lies in $S_{i-1}$. So by applying the lower bound with $S = S_{i-1}$,

$$\text{OPT} \geq \frac{k-1}{|S|} \geq \frac{(i-1)|S_i|+1}{|S_{i-1}|} \geq \frac{i-1}{2} \geq \frac{\deg(T)-L-1}{2}$$

Thus $\deg(T) \leq 2 \cdot \text{OPT} + L + 1$

**Remark:** A better approximation algorithm with $OPT + 1$ guarantee is known using a more sophisticated local search. An LP-based algorithm provides such a result even for the problem with edge costs.