6.1 Local Search: Min Degree Spanning Tree (MDST)

6.1.1 MDST

Problem description

Input: Graph $G = (V, E)$.

Feasible solution: Spanning tree T.

Objective: $\min \Delta(T)$, where $\Delta(T) = \max_{u \in V} d_T(u)$.

Here $d_T(u)$ is defined as the degree of u in graph T. Recall that the fundamental cycle of a non-tree edge e (with respect to some spanning tree) is the unique cycle created by adding e to the spanning tree. We made the following definition last class.

Definition 6.1.1 *$(u, \{v, w\})$ is a u-improvement if there is an x adjacent to u in T such that $\{u, x\}$ is on the fundamental cycle of $\{v, w\}$, and if we add $\{v, w\}$ to T while removing $\{u, x\}$ we get a spanning tree T' such that* $\max\{d_{T'}(v), d_{T'}(w)\} \leq d_{T'}(u) = d_T(u) - 1$.

6.1.2 Local Search (LS) Algorithm for MDST:

Algorithm 1 MDST Local Search Algorithm

Input: Undirected graph $G = (V, E)$.
Output: A minimum spanning tree of G.

```
T ← Arbitrary spanning tree
while there is a u-improvement with d_T(u) ≥ Δ(T) − log(n) do
    T ← T with u-improvement
end while
return T
```

Last class we proved that this algorithm terminates in polynomial time (at most $O(n^4)$).

Theorem 6.1.2 *Let T be the tree returned by LS. Then*

$$\Delta(T) \leq 2 \cdot \Delta^* + \log n$$

where Δ^* is the maximum degree of the optimal tree.

Before getting to the proof, note this lemma.

Lemma 6.1.3 *Consider partitioning G into piece $V_1, V_2, \ldots, V_k$. Let $V' \subseteq V$ be the set of nodes that have at least one neighbor in a different piece. Then any spanning tree must have at least $k-1$ edges between the pieces. Hence $\Delta^* \geq \frac{k-1}{|V'|}$.*

Proof of Theorem 6.1.2: First some definitions. Throughout, we will only care about $i \geq \Delta^* - \log n$.

Definition 6.1.4 *Let S_i be the nodes with degree at least i in T.*

Definition 6.1.5 *Let E_i^T be edges of T incident on nodes in S_i.*

Claim 6.1.6 $|E_i^T| \geq (i-1)|S_i| + 1$

Proof: By the definition of S_i, we have that

$$\sum_{u \in S_i} d_T(u) \geq i|S_i|.$$

Now the number of edges $\{u, v\} \in T$ with $u, v \in S_i$ is at most $|S_i| - 1$ because T is a tree. This implies that

$$|E_i^T| \geq i|S_i| - (|S_i| - 1) = (i-1)|S_i| + 1$$

as claimed. ■

Now let E_i^G be edges in G between components of $T - E_i^T$.

Claim 6.1.7 *Let $e = \{x, y\} \in E_i^G$. Then either x or y in S_{i-1}.*

Proof: There are two cases to consider:

Case 1: $e \in E_i^T$. Then x or y is in $S_i \subseteq S_{i-1}$.

Case 2: $e \notin E_i^T$. Suppose that neither x nor y is in S_{i-1}. Since e goes between components of $T - E_i^T$, if we add e to T it closes a cycle which contains some edge $\{u, v\} \in E_i^T$. So either u or v in S_i – without loss of generality we will assume that $u \in S_i$. But then $(u, \{x, y\})$ is a u-improvement, since $d_T(u) \geq i$ and by assumption both x and y have degree in T at most $i-2$. This is a contradiction, since T is is returned by the algorithm so has no u-improvements for $u \in S_i$. ■

Claim 6.1.8 *There exists an $i \geq \Delta(T) - \log n$ such that $|S_{i-1}| \leq 2|S_i|$.*

Proof: Suppose this is false. Then $|S_{i-1}| > 2|S_i|$ for all $i \geq \Delta(T) - \log n$. Since $|S_{\Delta(T)}| \geq 1$, this implies that $|S_{\Delta(T)-\log n}| > n$, a contradiction. ■

Let i^* be the value from Claim 6.1.8 applied to T. Consider OPT. It needs at least $|E_{i^*}^T| - 1$ edges from $E_{i^*}^G$ to get between components of $T - E_{i^*}^T$. By Claim 6.1.7, any edge in $E_{i^*}^G$ has at least 1

endpoint in S_{i^*-1} so

$$\begin{aligned}\Delta^* &\geq \frac{|E^T_{i^*}| - 1}{|S_{i^*-1}|} \\ &\geq \frac{(i^*-1)|S_{i^*}|}{2|S_{i^*}|} \\ &\geq \frac{i^*-1}{2} \\ &\geq \frac{\Delta(T) - \log n}{2}.\end{aligned}$$

This implies that $\Delta(T) \leq 2\Delta^* + \log n$. ■

6.1.3 Weighted Max-Cut

Weighted Max Cut is the same as Max-Cut, but there is a weight w_e for each edge e and the objective is the max weight cut. Formally

Input: Graph $G = (V, E)$ with a weight w_e for each edge e.

Feasible solution: $S \subseteq V$.

Objective: $\max \sum_{e \in E(S,\bar{S})} w_e$.

Let $W = \sum_{e \in E} w_e$. Notice that vanilla local search might take $\Omega(W)$ time.

For each node v, let $\delta(v) = \{\{u, v\} \in E\}$ denote the edges incident on it. For a set of edges E', we let $w(E') = \sum_{e \in E'} w_e$. Finally, for $S \subseteq V$ we let $w(S) = w(E(S, \bar{S}))$ denote the weight of the edges that have one endpoint in S and one endpoint in $\bar{S}$.

6.1.4 Local Search (LS2) Algorithm for Weighted Max-Cut:

Algorithm 2 Weighted Max-Cut Local Search Algorithm

Input: Undirected graph $G = (V, E)$.
Output: A minimum spanning tree of G.

$S \leftarrow \{v\}$, where $w(\delta(v)) = \max_{u \in V} w(\delta(u))$.
while $\exists v \in \bar{S}$ (or S) such that $w(S \cup \{v\}) \geq (1 + \frac{\epsilon}{n})w(S)$ (or $w(S \setminus \{v\}) \geq (1 + \frac{\epsilon}{n})w(S)$) **do**
  Add (remove) v if first (second) condition.
end while
return S

Theorem 6.1.9 *There are at most $O(\frac{1}{\epsilon} \cdot n \log n)$ iterations which implies that the running time is $O(\frac{1}{\epsilon} \cdot n^2 \log n)$.*

Definition 6.1.10 *Let $S_0 = \{v\}, S_1, S_2, \ldots, S_r$ be sets created by LS2.*

Proof of Theorem 6.1.9: Notice

$$\begin{aligned} w(S_0) &\geq \frac{W}{n}, \\ w(S_i) &\geq \left(1+\frac{\epsilon}{n}\right) w(S_{i-1}) \\ &\geq \left(1+\frac{\epsilon}{n}\right)^i \frac{W}{n}. \end{aligned}$$

Then,

$$\begin{aligned} & w(S_r) \geq \left(1+\frac{\epsilon}{n}\right)^r \frac{W}{n} && \text{(Note that } w(S_r) \leq W\text{)} \\ \implies & \left(1+\frac{\epsilon}{n}\right)^r \leq n \\ \implies & r \leq \frac{\log n}{\log\left(1+\frac{\epsilon}{n}\right)} \leq \frac{\log n}{\frac{\epsilon}{2n}} = \frac{2}{\epsilon} n \log n. \end{aligned}$$

■

Theorem 6.1.11 *LS2 is a $(2+\epsilon)$-approximation.*

Definition 6.1.12 $w(\{v\}, S) = \sum_{e=\{v,u\}, u\in S} w_e$.

Proof of Theorem 6.1.11: Let S be the set returned by LS2. Then $\forall v \in S$

$$\begin{aligned} w(S \setminus \{v\}) &\leq \left(1+\frac{\epsilon}{n}\right) w(S) \\ w(S \setminus \{v\}) - w(S) &\leq \frac{\epsilon}{n} w(S) \\ w(\{v\}, S) - w(\{v\}, \bar{S}) &\leq \frac{\epsilon}{n} w(S) \\ w(\{v\}, \bar{S}) &\geq w(\{v\}, S) - \frac{\epsilon}{n} w(S) \\ 2w(\{v\}, \bar{S}) &\geq w(\delta(v)) - \frac{\epsilon}{n} w(S) \\ w(\{v\}, \bar{S}) &\geq \frac{1}{2} w(\delta(v)) - \frac{\epsilon}{2n} w(S). \end{aligned}$$

Similarly for $v \in \bar{S}$

$$w(\{v\}, S) \geq \frac{1}{2} w(\delta(v)) - \frac{\epsilon}{2n} w(S).$$

Therefore

$$\begin{aligned} 2w(S) = \sum_{v\in S} w(\{v\}, \bar{S}) + \sum_{v\in \bar{S}} w(\{v\}, S) &\geq \sum_{v\in V} \left(\frac{1}{2} w(\delta(v)) - \frac{\epsilon}{2n} w(S)\right) \\ &= W - \frac{\epsilon}{2} w(S) \end{aligned}$$

which implies

$$w(S) \geq \frac{W}{2+\frac{\epsilon}{2}}.$$

Since $OPT \leq W$, this completes the proof. ■