CS880: Approximations Algorithms

Scribe: Siddharth Barman **Lecturer:** Shuchi Chawla
Topic: Min Degree Spanning Tree **Date:** 02/15/07

In this lecture we give a local search based algorithm for the Min Degree Spanning Tree problem.

8.1 Min Degree Spanning Tree

Problem Statement: Given an unweighted graph G, find a spanning tree with least possible max degree.

This problem cab be shown to be $\mathcal{NP}$-Hard by reducing Hamiltonian path to it. Essentially existence of a spanning tree of max degree two is equivalent to a having a Hamiltonian path in the graph. This also shows that the best approximation we can hope for (unless $\mathcal{P} = \mathcal{NP}$) is $\Delta^* + 1$ where Δ^* is the max degree of the optimal tree. Note that we usually express approximations in a multiplicative form, but also occasionally in an additive form. In general, for additive approximations to be meaningful the optimal value should be bounded from below. As in this case Δ^* is an integer no less than two.

In this lecture we provide a *local search* approximation algorithm which achieves an approximation of $2\Delta^* + \log n$. The algorithm and its analysis is by Fürer and Raghavachari [1]. They also provide an improved approximation with max degree $\Delta^* + 1$ in the same paper.

The main idea behind local search is to consider a graph over the solution space; nodes of this meta graph represent possible solutions. We then define a set of low cost operations (edges in the meta graph) which morph one solution into another. The strategy is to start from some solution/node and traverse the meta graph, improving the objective function at each step. We stop when we hit a local optimum. The length of the path traversed dictates the time complexity of the algorithm. Also for a good approximation factor we must show that (compared with the global optima) any local minimum reached by the algorithm is not too bad.

For the min degree spanning tree problem the solution space consists of all spanning trees. The connecting operation is to swap edges till the degree goes down. Formally:

Definition 8.1.1 (T-swap:) *Add edge $e_1 \notin T$ and remove edge $e_2 \in T$ s.t. e_2 is on the unique cycle that is formed by adding e_1 to the tree.*

In order to improve the objective function value, we would like our local search algorithm to perform T-swaps as long as one of the following happens:

- the max degree of the current tree decreases or,
- the number of max degree nodes decrease

Unfortunately the *T-swap* operation with the above defined improvement criteria might get stuck with a high max degree tree (as shown in Figure 1). In particular in the figure, adding edge $(2, 3)$

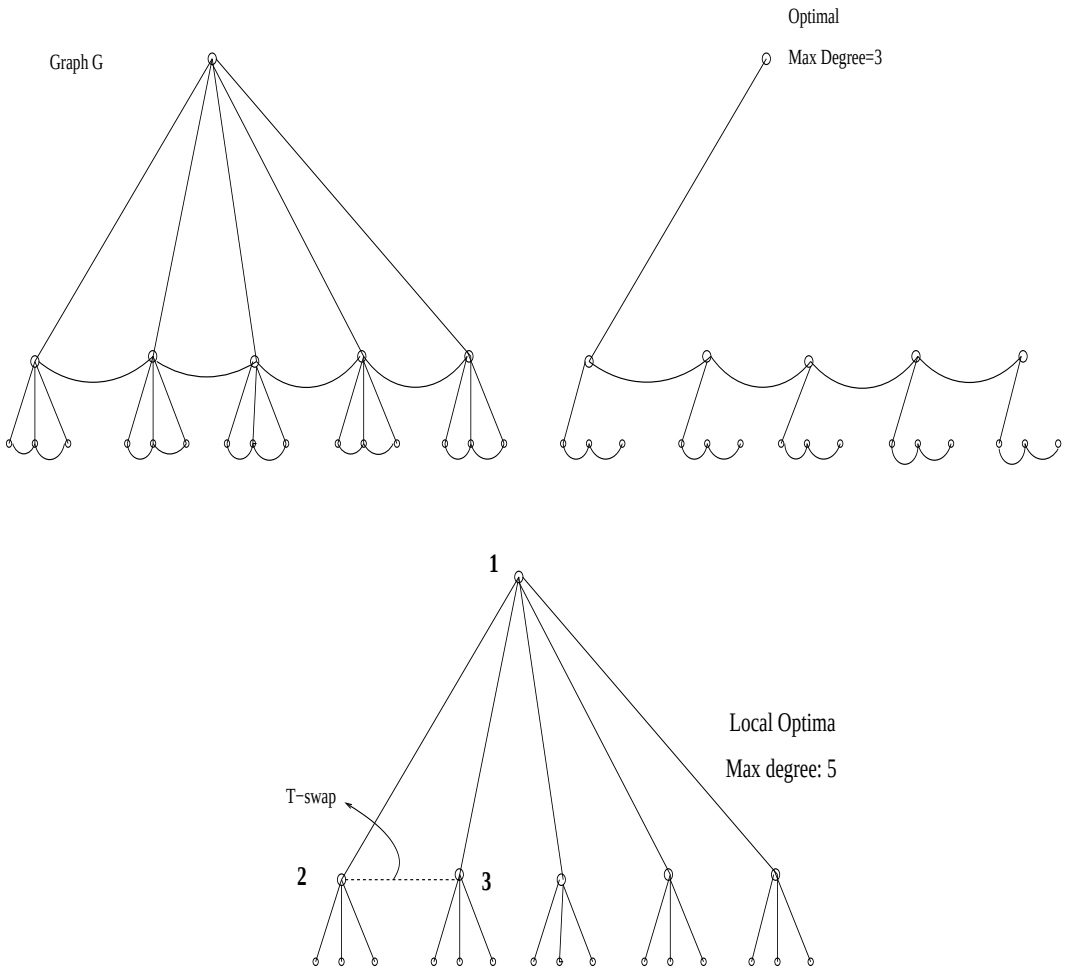

Figure 8.1.1: Allowed T-swaps

introduces cycle $1 \to 2 \to 3 \to 1$, but removing any edge edge from this cycle keeps the max degree at 5. So although the optimal tree has max degree only 3; the local optimal has max degree 5. Extending this example we can show that a local optimum can have max degree $\Omega(n)$ even when the optimal solution has max degree 3.

To overcome this problem, we allow T-swaps even when the above two conditions do not hold. In particular, consider edges $e_1 = (u_1, u_2)$ and $e_2 = (u_3, u_4)$ with $e_1 \notin T$, $e_2 \in T$ and e_2 lying on the unique cycle in $T \cup \{e_1\}$, say the T-swap of e_1 and e_2 transforms T into T'. Finally let δ indicate the max degree of the four end points of edges e_1 and e_2 in T, i.e. $\delta = \max_i\{deg_T(u_i)\}$ and similarly in T', $\delta' = \max_i\{deg_{T'}(u_i)\}$. We perform a T-swap if $\delta' < \delta$, i.e. we perform a T-swap if the max degree over the four end points of the involved edges goes down.

This operation does not necessarily decrease the objective function value, however we note that the degree sequence of the tree decreases lexicographically. In particular, say the string χ_T represents

the multiset of degrees of the vertices in tree T arranged in decreasing order. For example, with a max degree d, χ_T would look something like $d\ d\ d-1\ d-1$ .....221111. By performing a T-swap we replace the degree entry of δ by a lower value, δ', thus decrementing χ_T. The relevant observation here is that each step of the local search decreases χ_T lexicographically.

Let T be a locally optimal tree obtained by the above mentioned algorithm and T^* be the globally optimal tree. We denote by Δ the degree of T and by Δ^* the degree of T^*. Then we have the following,

Theorem 8.1.2 $\Delta \leq 2\Delta^* + \log n + 1$, *i.e. the degree of the local optimal found by the local search algorithm is bounded above by twice the degree of the global optimal with an additive factor of* $\log n + 1$.

We first give a generic lower bound on Δ^* in the following lemma. This along with the fact that T is a local optima will be used to establish a *witness* for Δ^*. We define $c(X)$ as the number of connected components in $G[V \setminus X]$.

Lemma 8.1.3 *For any $X \subseteq V$ we have*

$$\frac{|X| + c(X) - 1}{|X|} \leq \Delta^*$$

Proof:

We consider the average degree of X in any spanning tree of the given graph G. Any spanning tree must contain at least $|X| + c(X) - 1$ edges connecting the $c(X)$ components and the nodes in X to each other. Fewer edges would leave at least one of these disconnected from the other. Also components are not connected amongst themselves so each of these $|X| + c(X) - 1$ edges is incident to a vertex belonging to set X hence the average degree of X is no less than $\frac{|X|+c(X)-1}{|X|}$. This is true for any spanning tree hence $\Delta^* \geq \frac{|X|+c(X)-1}{|X|}$ ■

We now proceed to prove Theorem 8.1.2.

Proof of Theorem 8.1.2: Say the local optimal reached is T, we consider X_t the set of all nodes with degree no less than t in T. We denote the number of connected components in $T[V \setminus X_t]$ by $c_T(X_t)$. Note that the total degree of nodes in X_t is no less than $t|X_t|$.

We can bound this degree sum $t|X_t|$ by counting edges in T: Let us consider each of the $c_T(X_t)$ components as a single vertex and preserve connections of T between connected components and vertices in X_t i.e. edges between a connected component and a node of X_t are preserved, along with edges with both end points in X_t. The number of edges in this tree is $|X_t| + c_T(X_t) - 1$. The degree contribution of edges with both end points in X_t needs to be counted once more. But such edges are at most $|X_t| - 1$. Hence degree sum $t|X_t| \leq |X_t| + c_T(X_t) - 1 + |X_t| - 1$, which gives us

$$c_T(X_t) \quad \geq \quad t|X_t| - 2(|X_t| - 1) \tag{8.1.1}$$

We essentially use $c_T()$ to determine $c()$, T being a local optimum. The observation is that if A and B are two different connected components in $T[V \setminus X_t]$ and there is a graph edge between A and B then the end points of this edge is in X_{t-1} (see Figure 2). Otherwise this edge would have

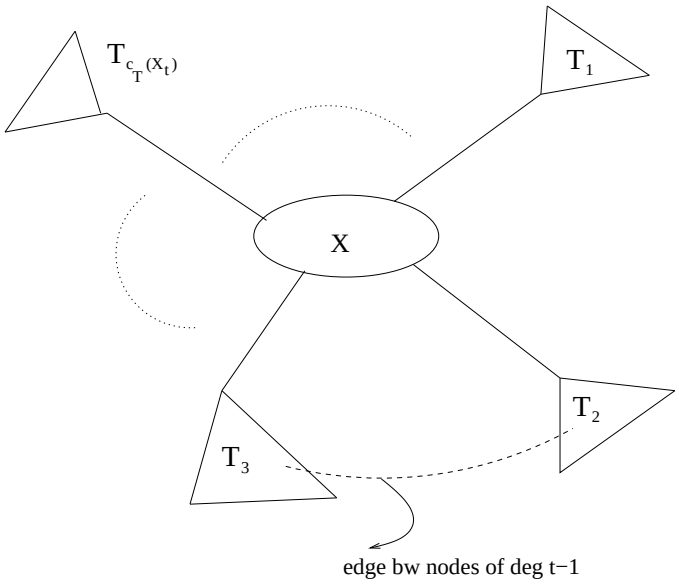

Figure 8.1.2: Connected components in tree

been T-swapped in to reduce the degree of one of the vertices currently in X_t. So if A and B are connected in the graph, it must be via an edge with end points in X_{t-1}. Hence, if we remove the X_{t-1} vertices all the $c_T(X_t)$ components definitely get disconnected in the original graph. This implies that $c(X_{t-1}) \geq c_T(X_t)$ for all t.

By lemma 8.1.3 we have

$$\Delta^* \geq \frac{c(X_{t-1}) + |X_{t-1}| - 1}{|X_{t-1}|}$$

Since $|X_{t-1}| \geq |X_t|$ and $c(X_{t-1}) \geq c_T(X_t)$ the above along with eqn. 8.1.1 reduces to

$$\begin{aligned} \Delta^* &\geq \frac{t|X_t| - 2(|X_t| - 1) + |X_t| - 1}{|X_{t-1}|} \\ &= \frac{(t-1)|X_t| + 1}{|X_{t-1}|} \\ &\geq \frac{(t-1)|X_t|}{|X_{t-1}|} \end{aligned}$$

We now pick a t that is large, and simultaneously for which $|X_t|$ is a large fraction of $|X_{t-1}|$. Note that there is always a t in $[\Delta - \log n, \Delta]$ s.t. $|X_t| \geq \frac{1}{2}|X_{t-1}|$. This holds because $\log n$ jumps of relative size more than $1/2$ will use up all the n vertices.

For this t, $\Delta^* \geq (\Delta - \log n - 1) \times \frac{1}{2}$ hence we have $\Delta \leq 2\Delta^* + \log n + 1$. ■

We now proceed to determine the running time of algorithm. We prove that the number of steps required to reach a local optimal is polynomial n.

Consider the degree sequence χ_T of nodes in the tree $\Delta\ \Delta\ \Delta - 1 \ldots 2\ 2\ 1\ 1$. As stated earlier at every step χ_T moves down lexicographically. But the number of different sequences and therefore, the number of possible steps is exponentially large. Instead we use a potential function argument to show fast convergence. We define the following potential function $\phi(T) = \sum_v 3^{deg(v)}$. The intuition behind this definition is that we assign exponential higher weight to high degree vertices. Also note that $\phi(T_0) \leq n3^{n-1}$.

Define $\Delta\phi$ as the change in potential function value obtained after a T-swap. Say currently we are at tree T_1 and T-swap edge (u_1, u_2) by introducing (v_1, v_2) to obtain new tree T_2, i.e. $T_2 = T_1 \setminus \{(u_1, u_2)\} \cup \{(v_1, v_2)\}$. Say

- $deg_{T_1}(u_1) = x_1$ and $deg_{T_1}(u_2) = x_2$
- $deg_{T_1}(v_1) = y_1$ and $deg_{T_1}(v_2) = y_2$

Assume w.l.o.g. that $x_1 \geq x_2$, since the T-swap was legal we have $y_1 + 1 \leq x_1 - 1$ and $y_2 + 1 \leq x_2$. This implies

$$\Delta\phi = \phi(T_1) - \phi(T_2) = \frac{2}{3}\ 3^{x_1} + \frac{2}{3}\ 3^{x_2} - \frac{4}{3}\ 3^{x_1 - 1}$$

We ignore the second term in the summand to obtain

$$\Delta\phi \geq \frac{2}{3}\ 3^{x_1} - \frac{4}{3}\ 3^{x_1 - 1} \geq \frac{2}{3}\ 3^{x_1 - 1}$$

We modify the algorithm to allow a T-swaps only if the max degree of the four end points of e_1 and e_2 is greater than $\Delta - \log n$ (this does not impact the analysis of Theorem 8.1.2). Hence $\Delta\phi \geq$ const $\times \frac{3^\Delta}{n}$.

As $\phi(T_1) \leq n3^\Delta$, we have $\Delta\phi \geq$ const $\times \frac{\phi(T_1)}{n^2}$. In other words in n^2 steps we shave off a constant factor. Therefore in $n^2 \times \log(Initial\ value) = O(n^2 \log(n3^{n-1})) = O(n^3)$ steps the algorithm would reach a local optimum. This implies that the running time of the algorithm is polynomial in n.

To summarize the local search algorithm is given as follows:

1. Start with an arbitrary spanning tree.
2. Say current max degree of tree is Δ
3. Consider edges $e_1 = (u_1, u_2)$ and $e_2 = (u_3, u_4)$ with $e_1 \notin T$, $e_2 \in T$ and e_2 lying on the unique cycle in $T \cup \{e_1\}$;
   Let $x_i = deg_T(u_i)$ for $i \in [4]$ and $\delta = \max_{i \in [4]}\{x_i\}$;
   Let $\delta' = \max\{x_1 + 1, x_2 + 1, x_3 - 1, x_4 - 1\}$;
4. If ($\delta > \delta'$ and $\delta \geq \Delta - \log n$) perform T-swap of edges e_1 and e_2.
5. Repeat steps 2 through 4 until no edges satisfy the conditions in step 4.

References

[1] M. Fürer, B. Raghvachari. Approximating the minimum degree spanning tree to within one from the optimal degree. *In: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms(SODA)* (1992), pp: 317-324