15-854: Approximations Algorithms **Lecturer:** R. Ravi
Topic: Local Search: Max Leaf Spanning Tree, Min Degree Spanning Tree **Date:** 9/26/2005
Scribe: Florin Oprea

5.1 Introduction

In this lecture we discuss the local search technique for designing approximation algorithms. The technique works as follows: we start with an arbitrary solution for a given problem, and perform small changes to it that improve the objective function. This is repeated as long as any improvement can be made. When this is no longer possible we have reached a local optimum. Using the local optimality property we can then bound the cost of the obtained solution. The performance of the algorithm (both the running time and the quality of the output) depends crucially on the type of small changes applied to the current solution: the more involved the local search for the next solution, the better the quality of the output, but at the cost of a higher running time.

We illustrate this technique with algorithms for the max leaf spanning tree and the min degree spanning tree problems.

5.2 Max Leaf Spanning Tree

Problem definition. Given an undirected graph $G = (V, E), |V| = n, |E| = m$, find a spanning tree T of G with a maximum number of leaves.

The maximum-leaf spanning tree problem is NP-complete by a reduction from the dominating set problem ([3]). We now present a local search algorithm based on the following simple observation: if we add a new edge f to a tree T we obtain a cycle. Removing an edge $e \neq f$ from the cycle gives us a new tree $T \setminus e \cup f$. If the new tree has more leaves than the former we have made an improvement.

Formally, we define the following two operations: T-swap and T-improvment. For a pair of edges (e, f), with $e \in T,\ f \in G \setminus T$ such that e belongs to the unique cycle of $T \cup f$, we define T-swap$(e, f) = T \setminus e \cup f$. Denote by $l(T)$ the number of leaves of the tree T. We define T-improvment as an arbitrary T-swap(e, f) such that $l(T \setminus e \cup f) > l(T)$. The local search algorithm is then the following:

Local Search - Max Leaf Spanning Tree
Let T be a spanning tree of G.
While there exists a T-improvment
  Let $T :=$ T-improvment.
Output T

Note first that this algorithm terminates in polynomial time: at each step we increase the number of leaves of the tree Twhich is bounded by n. Also each step can be implemented in polynomial

time by examining the edges in $G \setminus T$. Denote by LOT the output of the algorithm and let T be an arbitrary spanning tree of G. We will prove the following result:

Theorem 5.2.1 *For any tree T,*

$$l(LOT) \geq \frac{1}{10}\, l(T) \tag{5.2.1}$$

Since T can be any tree, it can also be an optimal one, which gives us an approximation ratio of 10 for local search.

Corollary 5.2.2 *The previous local search algorithm is a 10-approximation for the max leaf spanning tree problem.*

Before going to the proof of theorem 5.2.1, let us first note that 10 is not the best approximation factor for local search. In fact, a more carefull analysis can reduce it to 5 (for details, see [4]).

For the proof of theorem 5.2.1 let us start with some notation. Let $n_i(T)$ be the number of nodes of degree i for tree T (or simply n_i if there is no confusion about the tree T). Let $n_{\geq 3}(T)$ be the number of nodes of degree ≥ 3. Note that $l(T) = n_1(T)$. The following two claims will be used in the proof.

Claim 5.2.3 *For any tree T, $n_{\geq 3}(T) < n_1(T)$.*

Proof: Let T be an arbitrary tree. Then:

$$\sum_{v \in V} \deg_T(v) = 2(n-1) \tag{5.2.2}$$

Using the notation defined above, we have:

$$\sum_{v \in V} \deg_T(v) = \sum_{\deg i} i\, n_i \tag{5.2.3}$$

$$= 1 \cdot n_1 + 2 \cdot n_2 + \sum_{i \geq 3} i\, n_i \tag{5.2.4}$$

and also,

$$n_1 + 3 \sum_{i \geq 3} n_i \leq n_1 + \sum_{i \geq 3} i\, n_i = 2(n - n_2 - 1) \tag{5.2.5}$$

From equations (5.2.3) and (5.2.5) we conclude that $\sum_{i \geq 3} n_i < n_1$. ∎

An alternate proof for this claim is the following: starting from a tree T, collapse all the nodes of degree 2. The remaining tree has only internal nodes of degree at least 3. A simple induction argument now shows that, in such a tree, the number of leaves is strictly greater than the number of internal nodes.

The following claim bounds the number of 2-*paths* of a tree. A 2-*path* in a graph is a maximal path with all the internal nodes of degree 2. Let $n_{\text{2-paths}}(T)$ be the number of such paths in a tree T.

Claim 5.2.4 $n_{2\text{-}paths}(T) \leq 2\, l(T)$

Proof: Note that each 2-path is bounded by nodes of degree 1 or at least 3. This implies that $n_{2\text{-paths}} \leq n_1 + n_{\geq 3} - 1$. By the previous claim, this is at most $2\, l(T)$. ■

Proof of Theorem 5.2.1: Let T be an arbitrary spanning tree of G and LOT a tree output by our algorithm. Recall that we are trying to upper bound the number of leaves of T by the number of leaves of LOT. Consider now the following partition of the nodes of LOT into 3 bins: the first bin contains all the nodes of degree 1, the second bin contains all the nodes of degree 2 and the third bin contains all the nodes of degree ≥ 3. Let us count how many of the leaves of T can be in each bin. The first one can have at most $n_1(\text{LOT}) = l(\text{LOT})$ leaves since this is the size of the bin. The same thing is true about the third bin, by claim 5.2.3. Finally, note that it is sufficient to show that the second bin has at most $8\, l(\text{LOT})$ leaves of T to obtain the 10-approximation.

To show this we will further partition the nodes of the second bin according to the 2-paths that they belong to (in LOT). Since the number of these paths is at most $2\, l(\text{LOT})$ (by claim 5.2.4), it is sufficient to prove that any such path has at most 4 leaves of T.

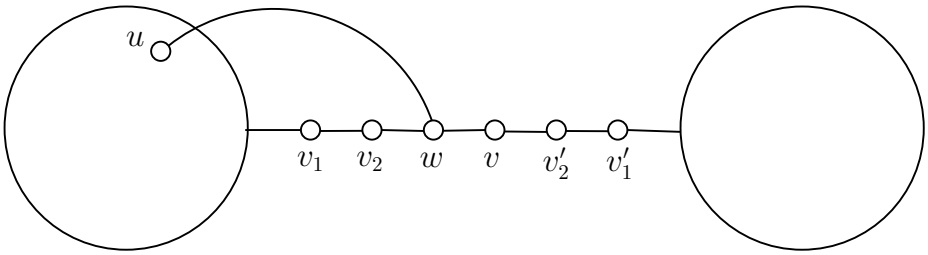

Figure 5.2.1: LOT with 5 leaves of T on a 2-path

Assume by contradiction that some 2-path P in LOT has (at least) 5 nodes from T. Denote these nodes by v_1, v_2, v, v'_2, v'_1 (see figure 5.2.1). Let v' be a node not in P. We denote by $path_T(v, v')$ the path form v to v' in T. Let u be the node closest to v in $path_T(v, v')$ such that u is not in P. Let w be the last node from P on $path_T(v, v')$ before u. Note that $v_2 \notin path_T(v, w)$ and also $v'_2 \notin path_T(v, w)$ (since otherwise they wouldn't be leaves of T). But $path_T(v, w) = path_{\text{LOT}}(v, w)$ by the choice of w and hence $w \in path_{\text{LOT}}(v_2, v'_2)$ (in other words $path_T(v, u)$ has to break off from P before v_2 or v'_2). Now one of the edges v_1v_2 or $v'_1v'_2$ has to be on the cycle formed in LOT by the addition of edge $uw \in T \setminus \text{LOT}$. This contradicts the local optimality of LOT. ■

5.3 Min Degree Spanning Tree

Problem definition. Given an undirected graph $G = (V, E)$, find a spanning tree T of G such that its maximum degree is the smallest among all spanning trees of G.

Denote by $\Delta_T = \max_{v \in V} \deg_T(v)$ the maximum degree of T. Finding a min degree spanning tree is NP-hard (see for instance [3]). Here we give an approximation algorithm for this problem due to Furer and Raghavachari ([2]). The algorithm is similar to the local search solution presented

earlier for the max leaf spanning tree problem.

Starting from an arbitrary spanning tree T we can add an edge $e \in G \setminus T$ that creates exactly one cycle in T. Removing a different edge from the cycle gives us a new spanning tree T'. If the maximum degree of T' is less than that of T we have made an improvement. We can then repeat this step as long as such an improvement exists.

We can define the same operations T-swap and T-improvment as before. Note first that a T-swap is a local operation: we only need to look at the degrees in T of the endpoints of the edges involved in the swap. Also note that if u_1v_1 and u_2v_2 are two edges involved in a T-improvment, then the maximum of $\deg_T(u_1)$ and $\deg_T(v_1)$ and the maximum of $\deg_T(u_2)$ and $\deg_T(v_2)$ have to be at least at a distance of 2 apart (otherwise the maximum degree does not decrease).

In fact, we need to be more carefull when defining a T-improvment, since otherwise we might get stuck in a place far from optimal. Consider the example from figure 5.3.2. Here the dotted lines indicate edges in $G \setminus T$. The tree T has one node of maximal degree d while all its children have degree $d-1$. Suppose we consider a T-improvment operation to be successful only if it brings down the maximum degree of the tree. In that case no direct T-improvment operation can be performed for T. The reason is that the only edges able to form a cycle which contains the node of degree d are those with both endpoints of degree $d-1$. Thus our condition regarding the distance of at least 2 between the maximum of degrees does not hold for any pair of edges (one of which is incident to the node of degree d). Yet improvement can be made if we first bring down the degree of some nodes of degree $d-1$ and then the degree of the node of degree d.

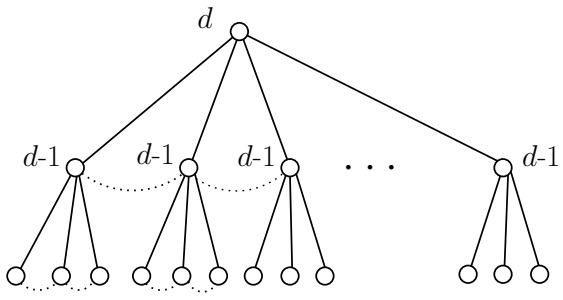

Figure 5.3.2: Tree with no direct improvment for the max degree node

This suggests the following definition for a T-improvment: a T-improvment is a T-swap at the end of which the maximum degree of the four vertices involved in the swap is strictly less than at the beginning of the swap. Note that now there is no guarantee that the maximum degree of T will decrease after each T-improvment. But if we look at the degree sequence of T sorted decreasingly after each T-improvment, we can see that between improvements it decreases in lexicographic order.

With the above definition of a T-improvment the local search algorithm is identical to the one from section 5.2. Furer and Raghavachari show that its output is not far from optimal using the following notion of a *witness*. Assume we remove a set $X \subseteq G$ of nodes from the graph G. Denote

by $cc(G \setminus X)$ the number of connected components resulted in G after this removal. We claim that any spanning tree T of G contains at least $cc(G \setminus X) + |X| - 1$ edges incident to vertices in X. To see why this is true consider the bipartite graph formed by the nodes of X on one side, the connected components on the other side and with edges the edges in T that connect either a pair of nodes of X or a node of X with a connected component (see figure 5.3.3). Note that there are no edges between the connected components in $G \setminus X$. Then any spanning tree for this bipartite graph has exactly $cc(G \setminus X) + |X| - 1$ edges and each of them is incident to some node in X. On the other hand all these edges correspond to distinct edges in T. In conclusion at least $cc(G \setminus X) + |X| - 1$ edges of T are incident to nodes of X.

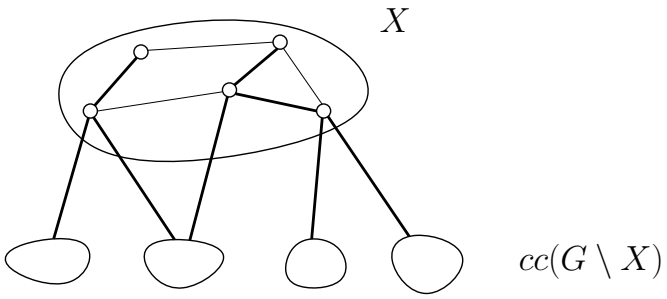

Figure 5.3.3: A witness for the max degree of a spanning tree

A simple averaging argument implies that there is some node in X with degree at least:

$$W(X) = \left\lceil \frac{cc(G \setminus X) + |X| - 1}{|X|} \right\rceil \tag{5.3.6}$$

Therefore the maximum degree of T satisfies $\Delta_T \geq W(X)$. We call $W(X)$ a *witness* for the maximum degree of T provided by the set X.

Let $W(G) = \max_{X \subseteq V} W(X)$ be the best such witness of a graph G. This is also called the *wimpiness* of the graph G. Let Δ^* be the maximum degree of an optimal tree T for the min degree spanning tree problem. The previous argument also implies that $\Delta^* \geq W(G)$.

It is worth mentioning here another property of a graph, which is in some sense the reciprocal of the wimpiness. We define the *toughness* of a graph G as:

$$T(G) = \min_{X \subseteq V} \left\lceil \frac{|X|}{cc(G \setminus X)} \right\rceil \tag{5.3.7}$$

This property is important because it provides a lower bound for the min degree of a spanning tree of a graph. In fact Agrawal, Klein and Ravi show in [1] that the maximum degree of the min degree spanning tree of a graph G is within a logarithmic factor from the toughness of G.

References

[1] A. Agrawal, P.Klein, and R. Ravi. How tough is the minimum-degree steiner tree? a new approximate min-max equality (complete with algorithms). Technical Report CS-91-49, Brown University, 1991.

[2] Martin Furer and Balaji Raghavachari. Approximating the minimum degree spanning tree to within one from the optimal degree. In *SODA '92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 317–324, 1992.

[3] M.R. Garey and D.S. Johnson. *Computers and intractability.* W.H. Freeman, 1979.

[4] H. Lu and R. Ravi. The power of local optimization: approximation algorithms for maximum-leaf spanning tree. Technical Report CS-96-05, Brown University, 1996.