Lecture 9 : Feb 8, 2002

Lecturer: Kamesh Munagala [1] *Scribe: Easwaran R*[2]

Local Search Techniques

Introduction

In the local search technique, the first step is to obtain any feasible solution to the problem. Given some feasible solution, all the 'nearby' feasible solutions are considered, and the better among them is chosen. This process is repeated till a locally optimal solution (a solution which is better than all its nearby solutions) is obtained.

For this technique to produce an approximation algorithm, the following needs to be ensured:

1. The local optimum is computable in polynomial time.
2. The local optimum is 'comparable' to the global optimum.

Local search for MAX-CUT

Algorithm 1 MAX-CUT

Input : Graph $G(V, E)$
Output : A partition of V into V_1 and V_2 , such that the CUT-SIZE is maximal

```
Arbitrarily partition the vertices into two sets V1 and V2
repeat
  if Moving a vertex from V1 to V2 or vice-versa improves the CUT size then
    Move the vertex
    Update V1 and V2
  end if
until No further improvement can be made
```

[1] Dept of Comp. Sci., Stanford Univ.
[2] Dept of Comp. Sci. and Automation, Indian Institute of Science, Bangalore.

The running time of the above algorithm is $O(|V| * |E|)$. The proof of the following theorem is left as an exercise.

Theorem 1 *The LOCAL-SEARCH for* MAX-CUT*produces a* 0.5 *approximation*

Minimum degree spanning tree

Definition 1 *The Minimum degree spanning tree (MDST) for a graph $G(V, E)$ is a spanning tree such that the maximum degree of any vertex in the tree is minimized.*

Let the degree of the optimal solution $= \Delta^*$

Deciding whether $\Delta^* = 2$ is $\mathcal{NP} - hard$ as this problem is reducible to finding a Hamiltonian Path in the graph. The following theorem states that a polynomial time approximation exists for the problem.

Theorem 2 *There exists a polynomial time algorithm that finds a tree of degree* Δ^*+1

In this lecture, we will be showing a $\log n$ approximation for this problem.

An algorithm to get $2\Delta^* + \log n$ degree spanning tree

Theorem 3 *If removing W vertices disconnects a graph into T connected components, then*

$$\Delta^* \geq \frac{W + T - 1}{W}$$

Proof: Removing the W vertices results in T components (Figure 1). For the W vertices to be connected, a minimum of $W - 1$ edges are needed, and since the T components have to be connected, another T edges are needed. Hence a total of $W + T - 1$ edges are needed. These edges are to be incident on the W vertices. Hence, there is atleast a vertex among the W vertices with degree $\geq \frac{W+T-1}{W}$ in any spanning tree. ■

The idea behind the local search technique for Minimum Degree Spanning Tree is as follows: Given a spanning tree, try to add an edge between two tree vertices and then remove an edge from the resulting cycle to see if the degree reduces.

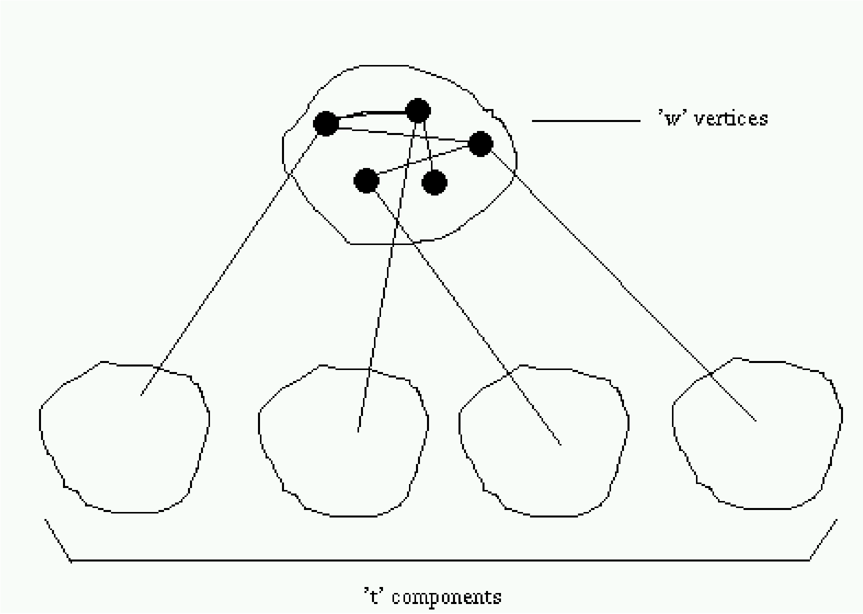

Figure 1: A graph split into t components by removing w vertices

Algorithm 2

Input :
Output :

1: **while** An edge (u, v) exists AND for some w in the path $u \to v$ in the tree $deg(w) > \max(deg(u), deg(v)) + 1$ **do**
2: Add edge (u, v)
3: Remove an edge in the cycle incident on w
4: **end while**

Let T_{LOPT} be the locally optimal tree produced by the algorithm, and let δ be its degree.

Theorem 4 *For any T_{LOPT},*

$$\delta \leq 2\Delta^* + \log n$$

Let S_i = Set of vertices in T_{LOPT} of degree $\geq i$

Lemma 1 *There exists $i \in [\delta - \log_2 n, \delta]$ s.t. $|S_{i-1}| \leq 2|S_i|$.*

Proof: If not, $|S_{i-1}| > 2|S_i|$ forall $i \in [\delta - \log n, \delta]$. Since $|S_\delta| \geq 1$, this implies $|S_{\delta - \log n}| > n$, which is a contradiction ■

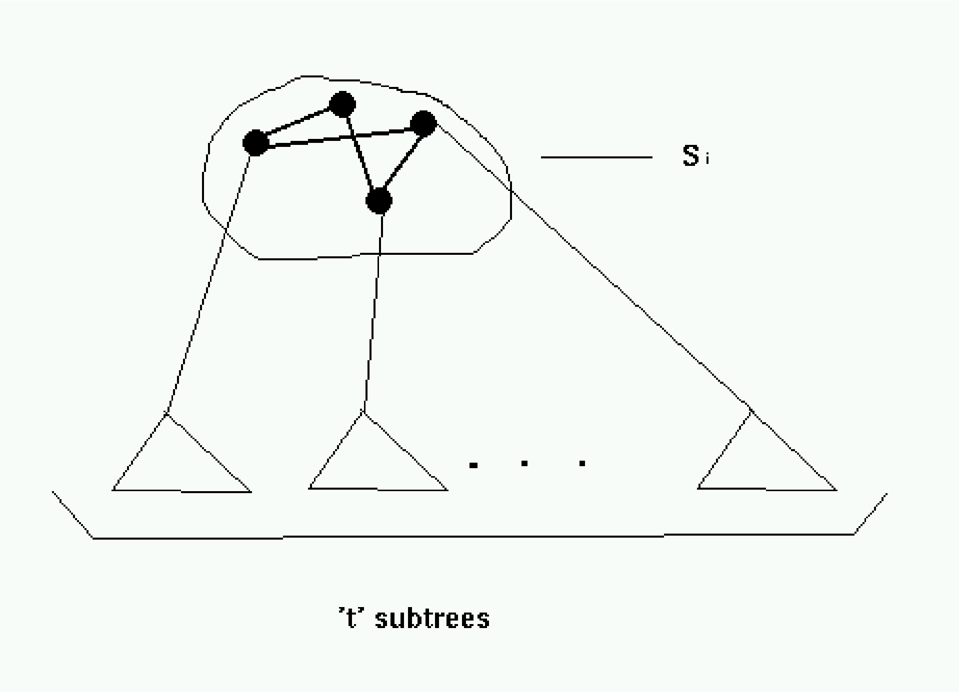

Figure 2: The components S_i splitting the graph into 't' subtrees

For the i satisfying the condition in the previous lemma, consider the set of vertices S_i of degree i in the tree. Let the rest of the vertices be divided into t subtrees (Figure 2).

We can make the following sequence of observations: The number of edges incident on the vertices in S_i is at least $i|S_i|$. Out of these, only $2(|S_i|-1)$ edges are edges within the set S_i. This implies the number of edges out of S_i is at least $i|S_i| - 2(S_i| - 1)$. This implies removing the set S_i results in at least $t \geq i|S_i| - 2(|S_i| - 1) \geq (i-2)|S_i| + 2$ subtrees.

For any edge (u, v) between two different subtrees the degree of either u or v is atmost $i-1$, as else the solution is not locally optimal. Therefore, if we remove the set S_{i-1}, the graph G disconnects into at least t parts where $t \geq (i-2)|S_i| + 2$. Set $w = |S_{i-1}|$.

We therefore have $\Delta^* \geq \frac{w+t-1}{w}$, which implies:

$$\begin{aligned}
\Delta^* &\geq \tfrac{|S_{i-1}|+(i-2)|S_i)}{|S_{i-1}|} \\
&= 1 + (i-2)\tfrac{|S_i|}{|S_{i-1}|} \\
&\geq 1 + \tfrac{(i-2)}{2} \\
&\geq \tfrac{i}{2}
\end{aligned}$$

But, $i \in [\delta - \log_2 n, \delta]$. This implies $\delta \leq 2\Delta^* + \log n$, completing the proof of the theorem.

The problem with the local search procedure described is that it spends a lot of time reducing the degree of small degree vertices. The following corollary gives a better stopping condition.

Corollary 1 *If $\nexists$ a vertex of degree $\geq \delta - \log n$ (where δ is the degree of the current tree), whose degree can be reduced by a valid local swap, then $\delta \leq 2\Delta^* + \log_2 n$*

We can therefore devise a faster local search procedure based on the above observation.

Algorithm 3 Local search procedure for MDST

Input : A graph $G(V, E)$
Output : A spanning tree $T(V, E')$, such that the maximum degree of the tree is minimized

1: Find a spanning tree and let k be its degree
2: **while** $\exists(u, v, w)$ such that

- $deg(w) > \max(deg(u), deg(v)) + 1$
- $deg(w) \geq k - \log_2 n$
- w lies on the path from u to v
- Edge (u, v) exists

do
3: Perform local swap
4: Update k
5: **end while**

Analysis of the algorithm

The running time of this algorithm is calculated by using the potential method. The potential function is defined as follows:

Let $\phi(u) = 3^d$, where d is the degree of the vertex u in the current tree

$$\phi(T) = \Sigma_{u \in V} 3^{d(u)}$$

Initially, let the degrees of w, u and v be i, j and k respectively. After one local swap, the degrees become $i-1$, $j+1$ and $k+1$ respectively, satisfying the inequalities:

$$\begin{aligned}
i-1 &> j \\
i-1 &> k \\
i &\geq \delta - \log_2 n
\end{aligned}$$

The change in potential is given by,

$$\begin{aligned}
\Delta(\phi(T)) = \phi_{INIT} - \phi_{FINAL} &= 3^i + 3^j + 3^k - (3^{i-1} + 3^{j+1} + 3^{k+1}) \\
&= 2 \cdot 3^{i-1} - 2(3^j + 3^k) \\
&\geq 2 \cdot 3^{i-1} - 4 \cdot 3^{i-2} \\
\Delta(\phi(T)) &\geq 2 \cdot 3^{i-2} \\
\Rightarrow \phi(T) &\leq n \cdot 3^\delta \\
\Rightarrow \frac{\Delta(\phi(T))}{\phi(T)} &\geq \frac{2 \cdot 3^{i-2}}{n \cdot 3^\delta} \\
&= \frac{2}{9n} \cdot \frac{1}{3^{\delta-1}} \\
&\geq \frac{2}{9n} \cdot \frac{1}{3^{\log n}} \\
&= \Omega(\frac{1}{n^3})
\end{aligned}$$

A simple argument now shows that the algorithm terminates in at most n^5 steps, as the initial potential is at most $n3^n$, and the final potential is at least n.