600.469 / 600.669 Approximation Algorithms **Lecturer:** Michael Dinitz
Topic: Local Search Algorithm for Max-Cut and Min-Degree Spanning Tree **Date:** 2/10/15
Scribe: Kuan Cheng

5.1 Max-Cut

- Input: a graph $G = (V, E)$, where V is the set of vertexes and E is the set of edges.
- Feasible solution: a set $S \subseteq V$. Here S is also called a cut of G.
- Objective: Maximize $|E(S, \bar{S})|$. Here $E(S, \bar{S}) = \{\{u, v\} : \{u, v\} \in E, u \in S, v \in \bar{S}\}$.

5.1.1 The Algorithm

Algorithm 1 Local Search Algorithm for Max-Cut

Input: A graph $G = (V, E)$
Output: A set $S \subseteq V$

```
Initialize S arbitrarily
while ∃u ∈ V with more edges to the same side than across do
    move u to the other side
end while
return S
```

5.1.2 Time Complexity

Theorem 5.1.1 *The Local Search algorithm for Max-Cut runs in polynomial time.*

Proof:

1. If there exists a vertex u with less than $\frac{1}{2}d(u)$ edges across, we can find u in polynomial time. ($d(u)$ denotes the degree of u)
2. Initially $|E(S, \bar{S})| \geq 0$. Finally $|E(S, \bar{S})| \leq m \leq n^2$. Every vertex switch increases $|E(S, \bar{S})|$ by at least 1.

So the overall running time is polynomial.

■

5.1.3 Approximation Factor

Theorem 5.1.2 *The local search algorithm is a 2-approximation.*

Proof: Say S is a local OPT if there is no improving step. No improving step means that $\forall u \in V$, there are more $\{u, v\}$ edges across the cut than connecting the same side.

Suppose S is a local OPT. Let $d_{across}(u)$ denote the number of edges incident on u that cross the cut. The since S is a local optimum, we know that

$$|E(S, \bar{S})| = \frac{1}{2} \sum_{u \in V} (d_{across}(u)) \geq \frac{1}{2} \sum_{u \in V} \frac{1}{2} d(u) = \frac{1}{4} \sum_{u \in V} d(u) = \frac{1}{4} \cdot 2m = m/2,$$

where $m = |E|$. We know $OPT \leq m$, and hence the algorithm is a 2-approximation. ■

5.2 Min-Degree Spanning Tree

- Input: a connected graph $G = (V, E)$.
- Feasible solution: A spanning tree T.
- Objective: Minimize $\max_{u \in V} d_T(u)$. Here $d_T(u)$ is the degree of u in the spanning tree T.

Theorem 5.2.1 *Min-Degree Spanning Tree is **NP**-hard.*

Proof: There is a reduction from Hamiltonian Path to Min-Degree Spanning Tree: given a graph G, it contains a Hamiltonian path if and only if it contains a spanning tree with maximum degree at most 2. ■

5.2.1 Local Search 1

We first define a local move. A local move is a pair (e, e'), where e is a non-tree edge and e' is a tree edge on the fundamental cycle of e (the cycle created in the tree by adding e).

Algorithm 2 Local Search Algo 1 for Min-Degree Spanning Tree

Input: A graph $G = (V, E)$
Output: A spanning tree T

```
Find a spanning tree T of G
while There is a local move which decreases the max degree of the current T do
  do the move
end while
Output T
```

As shown in figure 5.2.1, the algorithm may not work well. There is a local optimum with maximum degree d as shown in the left figure (each non-leaf node in the tree has degree d) while the OPT is 3 as shown in the right figure.

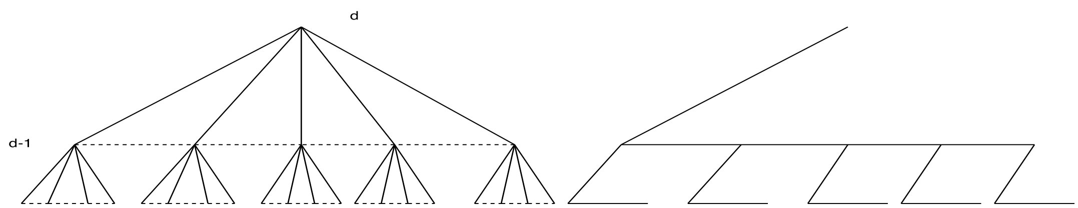

Figure 5.2.1: Example

5.2.2 Local Search 2

Let's improve Local Search 1.

Definition 5.2.2 *Let* $u \in V$*. Then* $(u, \{v, w\})$ *is a* u-improvement *if there is an edge* $\{u, x\}$ *on the fundamental cycle of* $\{v, w\}$ *so that we can swap* $\{v, w\}$ *for* $\{u, x\}$ *to get* T' *so that* $\max\{d_{T'}(v), d_{T'}(w)\} \leq d_{T'}(u) = d_T(u) - 1$

However, the running time is not polynomial if we just keep finding u-improvement for all vertices until we can't find any. Thus we will only perform u-improvements on nodes which have large degree. This gives us the algorithm Local Search 2.

Definition 5.2.3 $\Delta(T) = \max_{v \in V} d_T(v)$.

Algorithm 3 Local Search Algo 2 for Min-Degree Spanning Tree

Input: A graph $G = (V, E)$
Output: A spanning tree T

```
Find a spanning tree T of G
while There is u-improvement with d_T(u) ≥ Δ(T) − log n for the current T do
    do the improvement
end while
Output T
```

5.2.3 Time Complexity

Theorem 5.2.4 *The running time of Local Search 2 is polynomial.*

Proof: The proof works by analyzing a potential function. Let $\Phi(v) = 3^{d_T(v)}$, and let $\Phi(T) = \sum_{v \in V} \Phi(v) = \sum_{v \in V} 3^{d_T(v)}$.

First note that $\Phi(T) \geq \sum_{v \in V} 3 = 3n$, since all nodes have degree at least 1 in T. Also, $\Phi(T) \leq n \cdot 3^n$. The following claim states the decrease on $\Phi(T)$ after each improvement.

Claim 5.2.5 *Suppose we make a u-improvement $(u, \{v, w\})$ with $d_T(u) \geq \Delta(T) - \log n$, obtaining a new spanning tree T'. Then $\Phi(T') \leq (1 - \frac{2}{9n^3})\Phi(T)$.*

Proof: Suppose $d_T(u) = i$ with $i \geq \Delta(T) - \log n$. Then $d_{T'}(u) = i - 1$, so the decrease in $\Phi(u)$ is $3^i - 3^{i-1} = 2 \cdot 3^{i-1}$. The increase of $\Phi(v)$ is $3^{d_{T'}(v)} - 3^{d_T(v)} \leq 3^{i-1} - 3^{i-2} = 2 \cdot 3^{i-2}$, and the same is true for $\Phi(w)$.

So the overall decrease in Φ is at least

$$\begin{aligned}
2 \cdot 3^{i-1} - 4 \cdot 3^{i-2} &= \frac{2}{9} \cdot 3^i \\
&\geq \frac{2}{9} \cdot 3^{\Delta(T) - \log n} \\
&= \frac{2}{9 \cdot 3^{\log n}} 3^{\Delta(T)} \\
&\geq \frac{2}{9n^{\log 3}} 3^{\Delta(T)} \\
&\geq \frac{2}{9n^2} \cdot \frac{1}{n} \Phi(T) \\
&= \frac{2}{9n^3} \Phi(T).
\end{aligned}$$

Note that x (the other endpoint of the edge incident on u that we removed to add $\{v, w\}$) might also have a different degree in T' than in T, but in this case its degree will be smaller so this only helps us. ■

Suppose we run $\frac{9}{2}n^4 \ln 3$ iterations. Then $\Phi(T) \leq (1 - \frac{2}{9n^3})^{-\frac{9}{2}n^4 \ln 3} \cdot n3^n \leq n$. As $\Phi(T) \geq 3n$, this means that the algorithm must stop in less than $\frac{9}{2}n^4 \ln 3$ iterations. ■

5.2.4 Approximation Parameter

We will prove this next class.

Theorem 5.2.6 *The output spanning tree of Local Search 2 has max-degree at most $2 \cdot OPT + \log n$.*

The best known (and possible) result is a different algorithm which is still based on local search:

Theorem 5.2.7 *[FR94] There is a polynomial time algorithm which returns a spanning tree with max-degree at most $OPT + 1$.*

References

FR94 M. Furer and B. Raghavachari. Approximating the minimum-degree Steiner tree to within one of optimal, *Journal of Algorithms 17.3*, 1994, pp. 409–423.