6.896 Sublinear Time Algorithms	October 14, 2004
Lecture 10	
Lecturer: Ronitt Rubinfeld	*Scribe: Victor Chen*

1 Introduction

In this lecture, we show how to approximate Max-Cut on dense graphs. This is one of the most studied problem in this area. The algorithm that we present here follows more of a combinatorial approach.

2 Max-Cut

We first define some notations.

Definition 1 *Let $G = (V, E)$ be a graph and (V_1, V_2) a partition of V. $\mu(V_1, V_2) = \frac{|e(V_1,V_2)|}{n^2}$, where $e(V_1, V_2)$ denotes the set of edges with one endpoint in V_1 and one in V_2. $\mu(G) = \max_{V_1,V_2} \mu(V_1, V_2)$.*

$\mu(G)$ is simply the edge density of the largest cut in G. We will prove the following theorem due to Goldreich, Goldwasser, and Ron in this and the next lecture.

Theorem 2 *Given ϵ and a graph G, there exists an algorithm with query complexity $1/\text{poly}(\epsilon)$ and running time $2^{1/\text{poly}(\epsilon)}$ which outputs an approximate max cut density μ such that*

$$\Pr(|\mu - \mu(G)| \leq \epsilon) \geq 2/3.$$

Furthermore, there exists an algorithm with running time

$$2^{1/\text{poly}(\epsilon)} + O(n/\text{poly}(\epsilon))$$

that finds a partition V_1, V_2 such that $\mu(V_1, V_2) = \mu(G) - \epsilon$.

We build gradually toward the theorem.

2.1 Basic Ideas

We start with a simple idea that can augment an existing cut. Let $\Gamma(v, S)$ denote the set of neighbors of v in S. Suppose we have a partition (H_1, H_2) of V. If $|\Gamma(v, H_1)| < |\Gamma(v, H_2)|$, then moving v to H_1 increases the cut.

Suppose we are given an oracle that gives the number of neighbors on each side of the cut for each node. Then we can decide which side of the partition each node should be placed. The question is, how do we obtain such an oracle? We use approximation.

Here is our first attempt. Take a random small sample S of V. For any partition (S_1, S_2) of S, it induces a partition (V_1, V_2) of V where $v \in V_2$ if $|\Gamma(v, S_1)| \geq |\Gamma(v, S_2)|$. Else, put it in V_1. We hope that a max-cut partition of S induces a good partition. This approach works well for all nodes that have a significantly higher number of edges going to one side of S than the other. However, this approach neglects to take edges within V_1 and V_2 into account. For instance, suppose S is partitioned into (S_1, S_2) where S_1 is a clique, S_2 is an independent set such that all edges crossing the cut are present. This is a max-cut of S. Nevertheless, there can still be a clique in V_2, and this is not a good partition of V.

2.2 A linear time algorithm

Here we describe a linear time approximation algorithm, which is a first step at achieving the result of the theorem. Arbitrarily partition V into $V^1, \ldots, V^\ell$ for $\ell = 4/\epsilon$. Partition each V^i separately, using all partitions of sample from $V \backslash V^i$ to give "approximate oracle."

For all combinations of approximate oracles, one for each V^i, evaluate the partition of V and output the max cut found. Now we describe in detail this partitioning algorithm running in time $O(2^{\text{poly}(1/\epsilon)} n^2)$.

Partitioning Algorithm:
0. Partition V into $V^1, \ldots, V^\ell$ arbitrarily of roughly equal size, where $\ell = 4/\epsilon$ and $|V^i| = n/\ell$.
1. Let $U = (U^1, \ldots, U^l)$, where U^i is of size $t = \Theta(\frac{1}{\epsilon^2} \log \frac{1}{\epsilon})$ chosen uniformly in $V \backslash V^i$.
2. For each possible partition $\pi(U) = \langle (U_1^1, U_2^1), \ldots, (U_1^\ell, U_2^\ell) \rangle$, where for each i, (U_1^i, U_2^i) is a partition of U^i, obtain a partition of V as follows:
   For $i = 1, \ldots, \ell$, partition V^i into (V_1^i, V_2^i) so that for each $v \in V^i$,
   if $|\Gamma(v) \cap U_2^i| > |\Gamma(v) \cap U_1^i|$, then put v in V_1^i. Else, put v in V_2^i.
3. Output the partition of V with largest the cut.

Since there are $2^{\text{poly}(1/\epsilon)}$ possible partitions of U and $O(n^2)$ time to check the size of the induced partition of V, the total running time is $2^{\text{poly}(1/\epsilon)} n^2$. Now the following lemma will establish the correctness of the algorithm.

Lemma 3 (Correctness) *Let H_1, H_2 be any partition of V. Then with probability at least 5/6 over the choice of U, there exists a partition sequence $\pi = \langle (U_1^1, U_2^1), \ldots, (U_1^\ell, U_2^\ell) \rangle$ such that π induces a partition of V into (V_1^π, V_2^π) and $\mu(V_1^\pi, V_2^\pi) \geq \mu(H_1, H_2) - \frac{3}{4}\epsilon$.*

Proof (sketch of proof) For $i = 0, \ldots, \ell$, the i-th hybrid partition (H_1^i, H_2^i) has the vertices in $V^1, \ldots, V^i$ placed by the algorithm and the vertices in $V^{i+1}, \ldots, V^\ell$ partitioned as in (H_1, H_2). Note that (H_1^0, H_2^0) is simply (H_1, H_2) and (H_1^ℓ, H_2^ℓ) is (V_1^π, V_2^π). Think of H_1, H_2 as a max-cut. We want to argue that at each step, the edge density of the hybrid partition does not get too far from the edge density of the max-cut.

Now let A_i be the event that there exists a partition (U_1^i, U_2^i) of U^i such that $\mu(H_1^i, H_2^i) \geq \mu(H_1^{i-1}, H_2^{i-1}) - \frac{3\epsilon}{4\ell}$, where $U^1, \ldots, U^{i-1}$ are fixed. We want to show that $\cap_{i=1}^{\ell} A_i$ happens with high probability.

Main Claim: For fixed $U^1, \ldots, U^{i-1}$,

$$\Pr_{U_i}[A_i] \geq 1 - \frac{\delta}{2\ell}.$$

Think of δ as 1/6. The Main Claim will be proved next lecture. Notice that if the Main Claim holds, then by the Union Bound, $\Pr[\cup_{i=1}^{\ell} A_i^c] \leq \delta/2$, so $\Pr[\cap_{i=1}^{\ell} A_i] \geq 1 - \delta/2$. Hence, with probability at least 5/6, $\mu(V_1^\pi, V_2^\pi) \geq \mu(H_1, H_2) - \frac{3\epsilon}{4}$.

■