Approximation Algorithms - Tools and Techniques	**Spring 2002**

Lecture 15 : Mar 01, 2002

Lecturer: Kamesh Munagala [1] *Scribe: Siddartha Y R*[2]

In the previous lecture[3], we saw an $O(\log k \log D)$ approximation for the SPARSEST CUT problem. In the present lecture, we will use that result to establish an $O\left(\log^2 n\right)$ approximation algorithm for the MINIMUM BALANCED CUT problem.

The MINIMUM BALANCED CUT Problem

Given a graph $G(V, E)$ and weights $c : E \rightarrow \Re$ on the edges, to find the minimum cost set of edges to delete such that the graph is divided into two 'balanced' components, i.e; each component contains atleast $\frac{n}{3}$ vertices. This may be generalised as a minimum k-balanced cut, for $0 < k < 1$ and MINIMUM BALANCED CUT is the case $k = \frac{1}{3}$

We present some related problems first.

Related Problems

All-Pairs Unit Demand SPARSEST CUT Problem

The sparsest cut of a graph $G(V, E)$ with a unit demand assigned to every pair of vertices in V. Using the notations of the previous lecture,

$$\min_{F \in E} \frac{\sum_{e \in F} c_e}{|S||V \backslash S|}$$

where F is a cut dividing the vertex set into S and $(V \backslash S)$. It has applications in clustering, Markov chain (mixing times), etc. From the sparsest cut algorithm of the previous lecture, we have, with

$$k = D = \binom{n}{2}$$

an $O\left(\log^2 n\right)$ approximation algorithm.

[1]Dept of Comp. Sci., Stanford Univ.

[2]Dept of Comp. Sci. and Automation, Indian Institute of Science, Bangalore.
siddu@csa.iisc.ernet.in

[3]see notes on Lecture 14 for notations used in the following

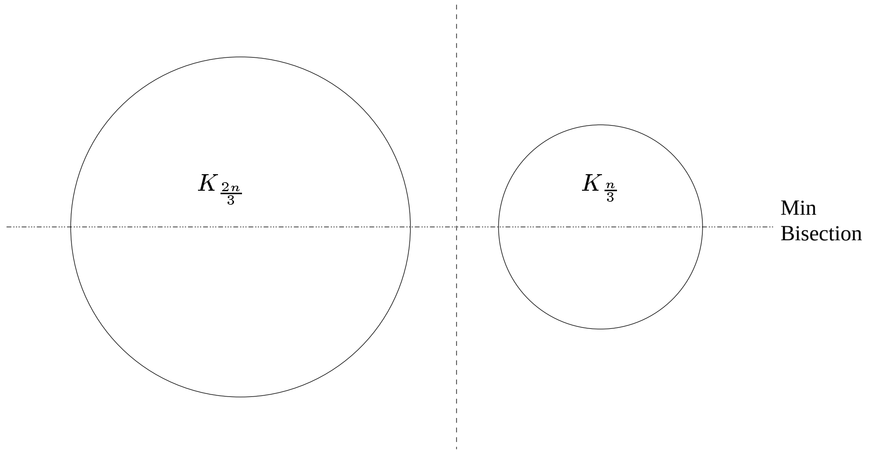

Figure 1: Graph with zero cost balanced cut ($C_{\frac{1}{3}}$) but costly min bisection C_b

The Minimum Bisection Problem

Given a graph $G(V, E)$ with even number of edges, Find the min cost of edges to delete such that it divides the graph into two components of size $\frac{n}{2}$. The min balanced cut is a relaxation of this problem.

Algorithm for MINIMUM BALANCED CUT Problem

Let $C_{\frac{1}{3}}$ and C_b be the cost of the minimum balanced cut and the minimum bisection respectively. Returning to balanced cuts, We now show the main result of this lecture.

Theorem 1 *There exists a poly-time algorithm for computing the balanced cut of cost* $O\left(\log^2 n\right) C_b$.

Note that we have "cheated" here since the approximation ratio does not relate the balanced cut returned by the algorithm to the min balanced cut; instead it relates the returned balanced cut to the min bisection. Figure. 1 shows an example where this does not lead to a true approximation, since the min balanced cut is zero but the min bisection is large (and our algorithm returns a cut which is $\log^2 n$ larger than this bisection).

Algorithm 1 MinBalancedCut

Input : Graph $G(V, E)$ over n vertices, with weights c_e on edge $e \in E$.
Output : A balanced cut $\{U, W\}$ with $|U|, |W| \geq \frac{n}{3}$

set $U \leftarrow \phi, W \leftarrow V$
repeat
  $H \leftarrow$ Subgraph of G induced by W
  $\{X, W\backslash X\} \leftarrow SparsestCut(H,$ "All pairs Unit Demand"$)$
  if $|X| \leq |W\backslash X|$ **then**
    $S_i \leftarrow X$
  else
    $S_i \leftarrow W\backslash X_i$
  end if
  $U \leftarrow U \bigcup S_i, W \leftarrow W\backslash S_i$
until $|U| \geq \frac{n}{3}$
return $\{U, W\}$

Note that the algorithm always produces a U, W such that $\frac{n}{3} \leq |U| \leq \frac{2n}{3}$ or $\frac{n}{3} \leq |W| \leq \frac{2n}{3}$. Before the final iteration, $|U| < \frac{n}{3}$ and $|W| > \frac{2n}{3}$. In the final iteration U is augmented with the *smaller* subset of W. This can occur in two ways.

1. The size of the smaller subset of W is $\leq \frac{n}{3}$ and this gives the final U, W with $\frac{n}{3} \leq |U| < \frac{2n}{3}$ and thereby $\frac{n}{3} \leq |W| \leq \frac{2n}{3}$; or

2. The size of the smaller subset of W is $> \frac{n}{3}$ and this results in a final U, W satisfying $\frac{n}{3} < |U| \leq \frac{2n}{3}$ and thereby $\frac{n}{3} \leq |W| \leq \frac{2n}{3}$

The algorithm is also poly-time since in each iteration atleast one vertex is added to U and the iterations stop when $|U|$ reaches $\frac{n}{3}$ vertices.

Let C_b be the min bisection. At any iteration, $|W| > \frac{2n}{3}$ and hence W has non-null intersections with both partitions of C_b say, A_1 and A_2, with A_1 being the larger, as shown in Figure. 2.

We have,

$$\begin{aligned} |A_1| \leq & \quad \tfrac{2n}{3} - \tfrac{n}{6} & = \frac{n}{2} \\ \Rightarrow |A_2| \geq & \quad \tfrac{2n}{3} - \tfrac{n}{2} & = \frac{n}{6} \end{aligned}$$

Then

$$\rho(C_b) \leq \frac{C_b}{|A_1||A_2|} = \frac{C_b}{|A_1||V - A_1|}$$

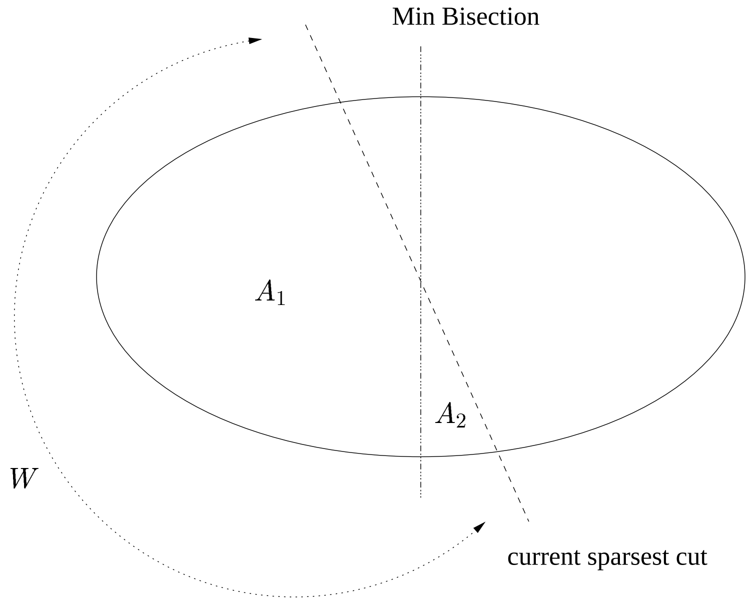

Figure 2: Relation between the min bisection C_b and the sparsest cut

Note that the the denominator is minimum when $A_1 = \frac{n}{6}$ and it equals $\frac{n^2}{12}$. Thus, there exists a cut in W of sparsity $\rho = \frac{12C_b}{n^2}$. From the algorithm for sparsest cut, we know that

$$\begin{array}{rrcl} & \frac{C(S_i)}{|S_i||W\backslash S_i|} & \leq & \frac{12C_b}{n^2}O\left(\log^2 n\right) \\ \Rightarrow & \frac{C(S_i)}{|S_i|} & \leq & \frac{12C_b}{n^2}O\left(\log^2 n\right)|W\backslash S_i| \\ & & \leq & \frac{12C_b}{n}O\left(\log^2 n\right) \end{array}$$

since $|W\backslash S_i| = O(n)$. This gives

$$C(S_i) \leq \frac{12C_b}{n}O\left(\log^2 n\right)|S_i|.$$

We also have

$$\begin{array}{rcl} C(U_{\text{final}}) & \leq & \sum_i C(S_i) \\ & \leq & \frac{12C_b}{n}O\left(\log^2 n\right)\sum_i |S_i| \\ & \leq & \frac{12C_b}{n}O\left(\log^2 n\right)|U_{\text{final}}| \\ & \leq & \frac{12C_b}{n}O\left(\log^2 n\right)\frac{2n}{3} \\ & \leq & O\left(\log^2 n\right)C_b \end{array}$$

which gives us the required result. Note that the above approximation ratio is completely dependant on the best available result for the sparsest cut. Thus, a better approximation for sparsest cut immediately results in a similar improvement for balanced cut.

The Min-Cut Linear Arrangement Problem

Given graph $G(V, E)$, arrange the vertices on a st. line such that at any point on the line, the maximum number of edges of G crossing that point (going from a vertex to the left of the point to one to the right of the point, or vice versa) is minimised.

Algorithm 2 MinCutLinArrangement

Input : Graph $G(V, E)$ over n vertices.
Output : An ordering of the n vertices satisfying Min-cut Linear Arrangement Property

```
set H ← G
if H has only a single node. then
    return node as arrangement
else
    compute balanced cut (H_1, H − H_1) of H.
    recursively call MinCutLinArrangement on H_1 and H − H_1 separately.
    Lay out arrangement returned for H_1 and H − H_1 side-by-side.
    return arrangement
end if
```

Let OPT be the optimal min-cut linear arrangement. Then $OPT \geq C_b$. From the algorithm for balanced cut we just saw,

$$\text{Cost of Cut}(H_1, H - H_1) \leq O\left(\log^2 n\right) C_b \leq O\left(\log^2 n\right) OPT.$$

For the recursive procedure, we have

$$\begin{aligned} OPT(H_1) \leq & \; OPT(H) \\ OPT(H - H_1) \leq & \; OPT(H) \end{aligned}$$

and the depth of the recursion is $\log_{1.5} n$. In going up one step of the recursion, say from H_1 to H, the number of edges added to the arrangement cost is $O\left(\log^2 n\right) OPT(H)$. Then, it follows, loosely, that

$$\#\text{edges in arrangement cost } = O\left(\log^3 n\right) OPT(G).$$