COM S 783 / OR&IE 634 Approximation Algorithms Fall Semester, 2003

Lecture 2: Sept 3, 2003

Lecturer: Yuval Rabani *Scribe: Aleksandrs Slivkins*

Recall that an r-*approximation algorithm* for a minimization (maximization) problem is a polynomial-time algorithm that produces a solution at most (at least) r times the optimum.

1 Maximum Cut

The problem Maximum Cut (MAXCUT) is, given an undirected graph G with weights w on edges, to find the edge-cut of maximum weight. We will see that it is easy to approximate MAXCUT within a factor of $\frac{1}{2}$. We will use MAXCUT to illustrate three different algorithmic approaches: local search, greedy algorithms, and random cuts.

First we consider the unweighted version of this problem, i.e. the case when all edge-weights are 1. Then we have a simple *local search* algorithm.

Algorithm 1.1 *Start with an arbitrary cut, repeatedly move a node to the other side if this improves the solution.*

We claim this algorithm is a $\frac{1}{2}$-approximation. Indeed, the size of the optimal solution is at most $|E|$. Each iteration improves the solution by at least 1, so #iterations is bounded by $|E|$. After the algorithm terminates, at least half of the edges incident to each node are cut (else moving this node improves the solution), so the weight of the solution is at least $\frac{1}{2}|E|$. Claim proved.

For the weighted case the above algorithm does not necessarily have a polynomial #iterations since it might be the case that each iteration improves a solution only by a small amount. However, the following *greedy* algorithm works.

Algorithm 1.2 *Scan nodes in arbitrary order, place each node on the side that this node has fewer edges to (ties broken arbitrarily).*

We claim that this algorithm outputs a solution of weight at least $\frac{1}{2}w(E)$ (and hence is a $\frac{1}{2}$-approximation). Indeed, let $v_0, v_1, v_2, \ldots$ be the order in which the nodes are scanned. Then E can be partitioned into sets $E_j = \{v_i v_j \in E : i < j\}$ so that when each v_j is placed, the weight of the edges from E_j that are cut is at least $\frac{1}{2}w(E_j)$. Claim proved.

Can we do better than a $\frac{1}{2}$-approximation? Note that we used $w(E)$ to upper-bound the optimum. It turns out that we cannot prove a better approximation factor using this upper bound. Indeed, consider unweighted K_{2n}, a complete graph on $2n$ vertices. It is easy to check that the max-cut in this graph is given by any set of n vertices and, therefore, has size n^2, whereas $|E| = n(2n-1)$. So the ratio of any solution and $|E|$ is (asymptotically) at most $\frac{1}{2}$.

Can we hope to prove a better approximation ratio for our algorithms, maybe with a better upper bound? It turns out that we can't. Consider unweighted $K_{2n,2n}$, a complete bipartite graph with $2n$ vertices on each side. The maximum cut in this graph has size $|E| = 4n^2$. Let A, B, C, D be disjoint sets of n vertices each, such that the partition is $(A+B, C+D)$. It is easy to see that both algorithms can output configuration $\mathcal{C} = (A + C, B + D)$ which is only a $\frac{1}{2}$-approximation. Indeed, our local search algorithm gets stuck in $\mathcal{C}$ (e.g. if it is the initial configuration), and the greedy algorithm can output $\mathcal{S}$ if the order of the scan is (A, B, C, D). Therefore for both algorithms the approximation guarantee is tight.

We conclude this section with yet another simple $\frac{1}{2}$-approximation.

Algorithm 1.3 *Pick a random cut: for each vertex, pick a side independently uniformly at random. Repeat if all nodes lie on one side, or if the weight of the cut is less than* $\frac{1}{2}w(E)$.

The probability p_e that a given edge e is cut is equal to $\frac{1}{2}w_e$, so by linearity of expectation the expected weight of the resulting cut is $\sum_{e\in E} w_e p_e = \frac{1}{2}w(E)$. Using Chernoff bounds one can prove that a polynomial #iterations suffices with an arbitrarily high probability.

2 Set Cover

In this section we consider the Set Cover problem. Denote $[n] = \{1 \ldots n\}$. The input of Set Cover is the ground set $[n]$ and a collection of sets $S_1 \ldots S_m \subset [n]$ with weights $w_1 \ldots w_m$. The goal is to find a min-weight sub-collection of sets that *covers* $[n]$, i.e. whose union is equal to $[n]$. We'll show a simple greedy algorithm which is a $O(\log n)$-approximation. Later in the course we'll prove that this approximation guarantee is essentially the best possible for Set Cover.

Theorem 2.1 *The following algorithm is a* $O(\log n)$*-approximation for Set Cover: pick a set* S_i *that maximizes the cost-effectiveness (#elements covered in this step divided by* w_i*), repeat until all elements are covered.*

To prove Thm. 2.1 we'll need to consider the IP-formulation of Set Cover and its the primal and dual LP-relaxations.

IP formulation. For each set S_i we have a variable x_i. The meaning of x_i is that it is equal to 1 if S_i is included in the cover, and 0 otherwise. Accordingly, Set Cover can be restated in terms of x_i's as follows:

$$\begin{array}{lll} \text{minimize} & \sum_{i=1}^{m} w_i x_i & \\ \text{subject to} & \sum_{i: j \in S_i} x_i \geq 1 & \text{for all } j \in [n] \\ & x_i \in \{0,1\} & \text{for all } i \in [m]. \end{array}$$

LP-relaxation. For the (primal) LP-relaxation we relax the constraint $x_i \in \{0,1\}$ to $x_i \in [0,1]$. Note that for this LP, this is equivalent to $x_i \geq 0$. The dual LP is as follows.

$$\begin{array}{lll} \text{maximize} & \sum_{j=1}^{n} y_j & \\ \text{subject to} & \sum_{j \in S_i} y_j \leq w_i & \text{for all } i \in [m] \\ & y_j \geq 0 & \text{for all } j \in [n]. \end{array}$$

Let W be the (weight of) the greedy solution. Our goal is to get a feasible dual solution D such that $W \leq \alpha D$ for some $\alpha = O(\log n)$. Then Thm. 2.1 follows by the weak LP-duality:

$$W/\alpha \leq D \leq \text{LP-dual} \leq \text{LP-primal} \leq \text{Set Cover}.$$

Let $S_1, S_2, \ldots$ be the sets picked by the greedy algorithm, let $T_i = S_i \setminus (S_1 \cup \ldots \cup S_{i-1})$. For each $j \in T_i$ assign $\tilde{y}_j = w_i/|T_i|$. Since T_i is the set of vertices covered at step i, we can think of $\tilde{y}_j$ as a share that each element pays to be covered by S_i. Note that all selected sets are fully paid for: their weight is $\sum_{j \in n} \tilde{y}_j$. To get a feasible dual solution, it remains to *scale* the $\tilde{y}_j$'s.

Claim 2.2 *$\sum_{j \in S_i} \tilde{y}_j \leq w_i \mathcal{H}(|S_i|)$ for each selected set S_i, where $\mathcal{H}(n) = \sum_{i=1}^{n} \frac{1}{i}$.*
Proof: We'll give the proof for the unweighted case; for the weighted case the proof is similar. Fix a set $S = S_k$ selected by the greedy algorithm. Let $\phi_i = |S \setminus (S_1 \cup \ldots \cup S_i)|$, $\phi_0 = |S|$. Note that $\phi_{i-1} - \phi_i$ is #elements of S covered in step i. Also note that by greediness $|T_i| \geq \phi_{i-1}$, since otherwise S would have been chosen instead of S_i. Then

$$\sum_{j \in S} \tilde{y}_j = \sum_{i=1}^{k} \frac{\phi_{i-1} - \phi_i}{|T_i|} \leq \sum_{i=1}^{k} \frac{\phi_{i-1} - \phi_i}{\phi_{i-1}} = \sum_{i=1}^{k} \sum_{t=\phi_i+1}^{\phi_{i-1}} \frac{1}{\phi_{i-1}} \leq \sum_{i=1}^{k} \sum_{t=\phi_i+1}^{\phi_{i-1}} \frac{1}{t} = \sum_{t=1}^{|S|} \frac{1}{t} \equiv \mathcal{H}(|S|).$$

$\square$

Let M be the size of the largest set picked by the greedy algorithm, let $\alpha = \mathcal{H}(M)$. Note that $\alpha = O(\log M) = O(\log n)$. By the above claim $\{y_j = \tilde{y}_j/\alpha\}$ is a feasible dual solution. Therefore, the weight of the greedy solution is $\alpha \sum y_j$, which completes the proof of Thm. 2.1.