Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 12

Lecture 58

Lecture 58 : SDP Based Approximation Algorithm for Max Cut Cont

welcome. So, in the last class we have started seeing the SDP based algorithm for max cut problem. We have seen that a we have written a vector program mean relaxation for it and then did a randomized rounding of the optimal solution of the vector program using a random hyperplane passing through origin and then we started doing the analysis. So, let us continue the analysis and prove the approximation guarantee of that algorithm in this lecture. So, in the last class we were proving this lemma that the probability that an edge $\{i,j\} \in E$ belongs to the cut output by the algorithm is $\frac{1}{\pi} arc\cos(v_i \cdot v_j)$ proof. So, let us briefly recall we were looking at the unit circle around origin in the plane spanned by v_i and v_j they are unit vectors and r' is the projection of r.

on the plane spanned by v_i and v_j and then we were arguing where does if where for what r_i' v_i and v_j fall belong to different cut different parts of the vertices in the cut. So, for that we did two lines one is perpendicular to v_i and another is perpendicular to v_j. and suppose the angle between v_i and v_j is θ and then we observe that only if r' which is uniformly distributed on the circle on the circle falls in this black region then only i and j will be in different components and that is equivalent to the condition that $\{i,j\}$ this edge is a cut edge. So, this is θ then this 2 green angles each of them are $90-\theta$ because they are perpendicular and hence these angles are also θ.

So, the probability that $\{i,j\}$ is cut is $\frac{2\theta}{2\pi}$ which is $\frac{\theta}{\pi}$. Now, what is θ? If I do the inner product of v_i and v_j that is length of v_i times length of v_j times $\cos(\theta)$. Now, v_i and v_j are unit vectors. So, this is equal to $\cos(\theta)$ So, θ is $arc\cos(v_i \cdot v_j)$ ok.

So, this completes the proof of this lemma. So, now, we prove the approximation guarantee theorem. The approximation factor of the SDP based max cut algorithm is at least 0.878. So, let us prove it.

So, for each edge $\{i,j\} \in E$ let x_{ij} be the indicator random variable for the event that $\{i,j\}$ is cut. So, alg which is the random variable a taking the value the cut size of the algorithm which is nothing, but $\sum_{\{i,j\}\in E} x_{ij}$. So, expectation of ALG is expectation of sum of the edges sum over the edges $\{i,j\}$ x_{ij}. Now, apply linearity of expectation this is expectation of x_{ij}. So, this is summation sorry there is a w_{ij} term here because of the weights.

So, this is $\sum_{\{i,j\}\in E} w_{ij} P[\{i,j\} is\, cut]$ which is $\sum_{\{i,j\}\in E} w_{ij} \frac{1}{\pi} arc\cos(v_i \cdot v_j)$. Now, we need to connect this with the objective function of the vector program which is a linear function of $v_i \cdot v_j$. So, for that we need a lemma which can be proved using elementary calculus that for x in you see inner product of v_i and v_j they are unit vectors and the inner product can vary from -1 to $+1$. So, we need to we need an expression which can bound $arc\cos(x)$ for x in between -1 to 1.

So, for every x in -1 to 1, $\frac{1}{\pi} arc\cos(x) \geq 0.878 \frac{1}{2}(1-x)$ you see $\frac{1}{2}(1-x)$ this is like a objective function. In the objective function we have this $\frac{1}{2}$ because each edge gets counted twice. So, recall in vector program it was like maximize $\sum_{\{i,j\}\in E} w_{ij}$ and then we have an expression $1 - v_i \cdot v_j$.

Now, this if you sum over all $\{i,j\}$'s that is fine, but if you sum over all $\{i,j\}$'s then it boils down whenever there is an edge between i and j a that the weight of that edge is w ij as usual, but if there is no edge between i and j we define an edge between i and j with weight 0. This way we have w_{ij} for every pair of i and j. So, this can be equivalently written as $\sum_{1 \leq i,j \leq n} w_{ij}(1 - v_i \cdot v_j)$, but here you see each edge contributes twice one as i j and another as j i. So, we need a factor of half. So, with this let us continue the analysis from here.

we get this is less greater than equal to 0.878 times sorry we need a half. Because, if v_i if i and j are in the same component then $v_i \cdot v_j$ is 1 which is 0 which $1 - v_i \cdot v_j = 0$, but on the other hand if $v_i \cdot v_j$ are in different component $v_i \cdot v_j$ is -1 and this becomes 2 for that we need we need half sorry we do not need this things. Now, we can continue from here. So, we missed this half in the earlier lectures on in the last lecture on max cut based on semi definite program, there in the optimization function we will be there will be an half factor because this is 0 if i j is in cut and 2 if I if this if this is 0 if i j is not in cut and 2 if i j is in cut, but we want a contribution of 1 that is why we need to multiply with half ok.

So, with this $\sum_{1 \le i, j \le n} w_{ij}(1 - v_i \cdot v_j)$. Now, this is VP-opt. So, this is equal to 0.878 VP-opt which is greater than equal to opt.

0.878 optimal. So, this shows that the approximation guarantee of our algorithm is 0.878 which turns out to be the best possible under a standard complexity theoretic assumption which is called unique games conjecture. Of course, it is the this is the best known approximation algorithm for the max cut problem. So, here is the theorem. Assuming unique games conjecture which is a complexity theoretic conjecture regarding a computational task in the context of a game.

this assumption is weaker than $P \neq NP$ assumption. In particular, if P is equal to NP then unique games conjecture is false, but if unique games conjecture is false that does not imply P equal to NP. So, in that way this is a weaker assumption, but this assumption often allows us to prove stronger inapproximability bound for various combinatorial optimization problems. So, under this conjecture, if we assume it to be true, we cannot have a better than 0.878-factor approximation algorithm for Max Cut.

Assuming the unique games conjecture, there is no alpha-factor polynomial-time approximation for the max-cut problem for any alpha that is greater than the minimum of x. Essentially, in this lemma, the best number you can place here is the maximum approximation guarantee you can achieve for the max-cut problem, assuming the unique games conjecture. Now, if you do not want to assume such a strong conjecture—about which, you know, even the research community is divided; some groups believe that this unicorn's conjecture is true, whereas there are some theoretical computer scientists who believe that it is not true.

So, if we only want to assume that P is not equal to NP, then we have the following inapproximability theorem: assuming P is not equal to NP, there is no polynomial-time α-approximation algorithm for the max cut problem. For any $\alpha > \frac{16}{17}$, which is roughly 0.941, okay? So, we obtain a weaker inapproximability guarantee assuming $P \neq NP$. In the remaining time of our course, we will examine some inapproximability bounds and how to prove certain inapproximability results. So, let us provide a brief overview.

The next topic is inapproximability, or polynomial-time inapproximability. These are typically assumptions; I assume that often $P \neq NP$. That is the weakest assumption on which we can base our theorem, and under this assumption, we want to show various inapproximability guarantees or proofs like this. For example, we have already seen a couple of such theorems in the lectures.

For example, we have seen the theorem that for any computable function, if there is a polynomial-time computable function α, then there is no polynomial-time α-approximation algorithm for the traveling salesman problem. So, what we have already seen is that if we have such an algorithm, we can solve the Hamiltonian circuit problem in polynomial time. So, another theorem that we will prove in the next class is scheduling jobs on multiple parallel, non-identical machines.

What is the problem statement? We have n jobs and m machines; it takes p_{ij} time units for machine j to process job i. Compute a non-preemptive scheduling. That means a job assigned to a machine should be allowed to run until it finishes. That means if job i is assigned to machine j, it should be allowed to run for e will prove in the next class is sch amount of time. Our goal is to compute a preemptive schedule that minimizes the maximum completion time of any job, which is also known as the makespan of the schedule.

We will see this inapproximability result in the next lecture: assuming $P \neq NP$, there is no polynomial-time α-approximation algorithm for scheduling jobs on multiple non-identical machines for any α. Less than $\frac{3}{2}$, because it is a minimization problem; the approximation factors are greater than 1, okay. So, we will see the proof of this theorem in the next class, okay