Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 01

Lecture 08

Lecture 08 : Randomized Rounding Algorithm for Weighted Set Cover

Welcome. So, in this lecture we will see yet another powerful approximation algorithm design technique called randomized rounding of linear programs and again we will understand this through the example of set cover. So, this is called randomized rounding of LPs. Recall what was the linear programming relaxation for the set cover problem? For each set we had a variable called x_j and we have minimize $\sum_{j=1}^{m} w_j x_j$ subject to. that every element is covered which is equivalent to saying for every $i \in [n]$ $\sum_{j \in [m]: e_i \in S_j} x_j \geq 1$ and for all $j \in [m], 0 \leq x_j \leq 1.$

And again we can remove this constraint because the optimal solution will never pick never set any x_j value greater than 1 because it is a minimization problem, but we can retain it no harm. So, in randomized rounding we solve the linear program to get an optimal solution and whenever this variables take value in between 0 1 we can treat them as probabilities. So, we solve LP let $(x_j^*)_{j \in [m]}$ be an optimal solution. Then the idea is we pick the j-th set with probability x_j^* independent of everything else.

So, we pick the set S_j with probability x_j^* independent of everything else ok. So, let I be the indices of the sets picked. we will show that the total the expected value of the weight total weight of the sets picked is actually LP-opt. So, for that define an indicator random variable X_j for $j \in [m]$ for the event that S_j is picked. by the algorithm that is X_j takes value 1 if $j \in I$ and 0 otherwise.

So, expected value of the solution is $\sum_{j=1}^{m} w_j X_j$ is the weight of the solution and solution picked by the algorithm and we are taking expectation there is something called linearity of expectation with which we can push the this expectation inside. So, this is same as $\sum_{j=1}^{m} w_j E[X_j]$. And, what is expectation of X_j? It is the probability that S_j is picked. x_j^* this is probability that S_j is picked which is same as saying j belongs to I.

So, this is the probability expectation of an indicator random variable is the probability of the event for which the indicator random variable is defined. You can prove also expectation of X_j is 1 times to probability that X_j takes value 1. So, this is this term plus 0 times probability that X_j takes value 0, but because it is 0 times that k that term is not there. So, this is we are picking j-th set with probability x_j^*. So, this is $\sum_{j=1}^{m} w_j x_j^*$ which is nothing, but LP-opt.

So, we are picking a solution of some sets whose total weight in expectation is LP-opt. But, what is but is it a valid solution, what is the probability that the set speed by the algorithm is indeed a set cover. So, for that probability that an element e_i is covered. by the sets index by I, I is the set of indices indices of the sets picked by the algorithm. So, what is the probability? This is nothing, but product over $j \in [m]$ such that $e_i \in S_j$ and this is the probability that e_i is the say e_i is not covered that is easy to compute and then we will subtract this number from 1 and get the probability that e_i is covered.

So, e_i is not covered if none of the sets e_i is contained is picked. So, this is $\prod_{j \in [m]: e_i \in S_j} (1 - x_j^*)$. Now $1 - x \le e^{-x}$. So, we are using that

$$\prod_{j \in [m]: e_i \in S_j} (1 - x_j^*) \le \prod_{j \in [m]: e_i \in S_j} e^{-x_j^*} = e^{-\sum_{j \in [m]: e_i \in S_j} x_j^*} \le \frac{1}{e}$$

So, the probability that the element e_i is not covered is at most e to the power minus 1 that means, with probability at least $\left(1 - \frac{1}{e}\right)$ it is covered, but this is not high probability. We want to cover all elements in the universe with probability $1 - \frac{1}{poly(n)}$, I want the probability which is little o of 1 minus little o of 1. So, for that the trick is simple this is quite prevalent in randomized algorithm this is called boosting your probability by repetition. You repeat this many times and then you see the probability then we will see the error probability goes down probability goes up. So, the idea is we the remedy is we repeat the algorithm $c \ln n$ times we will find the value c, $c \ln n$ times let the indices of the sets be $I_1, I_2, \ldots, I_{c \ln n}$ in the algorithm that they are $I_1 \cup I_2 \cup \ldots \cup I_{c \ln n}$ ok.

So, now, we can see that expected is at most in one times it is at most in one iteration the expected weight of the sets picked by the algorithm is at most LP-opt and we are we are repeating it for c log n time and returning all the sets picked. So, this is at most $c \ln n \times LP-opt$. Now find out what is the probability that some element is not covered probability that e_i is not covered. the probability that e_i is not covered in one iteration is

$(1-\frac{1}{e})$. And so, it does it is not covered in c log n iterations is $(1-\frac{1}{e})^{c\ln n}$.

Now, we apply union bound probability that there exists an $i\in[n]$ such that e_i is not covered is less than equal to $\frac{1}{n^{c-1}}$ ok. So, this is called good probability. So, if we if we choose choose c to be say 2, then the algorithm outputs collection of sets of total weight at most twice log n times of and it is a set cover with probability at least $1-\frac{1}{n}$ ok. So, this is the statement using this we can also show that let me write that this theorem which is easy to prove that the algorithm is a randomized $O(\ln n)$ factor randomized $2\ln n$ approximation algorithm which outputs a set cover with high probability ok. In the randomized algorithm in randomized algorithms typically high probability means probability at least $1-\frac{1}{poly(n)}$ and very high probability means probability at least $1-\frac{1}{\exp(n)}$ say c^n for some c greater than 1. this shows the thing and now we have seen various algorithms for set cover some of them their approximation factor is f and some others have approximation factor $O(\log n)$. It turns out that there is substantial evidence to believe that there is no polynomial time algorithm for set cover problem which achieves approximation ratio better than log n. So, let me state the results without proof.

These are some important results well known which you can use in your research theorem. If there exists a $c\ln n$ approximation algorithm for even the unweighted set cover problem for the unweighted set cover problem for some constant c less than 1, then there is a $O(n^{O(\log\log n)})$ time deterministic algorithm for each NP complete problem ok, which is which we believe to be quite unlikely. $O(n^{O(\log\log n)})$ this sort of running time is very close to polynomial time running time in for all practical purposes. $\log\log n$ is a very slow growing function even if n is the number of atoms in the you know in the earth then also $\log\log n$ is a very small number and n to the power $\log\log n$ is definitely tractable, but so that is why we believe that if there is a we believe that for NP complete problems there is no such algorithm. And, this result shows that even if there is a 0.9 log n approximation algorithm even for unweighted set cover problem, then for all NP complete problems there are there will be $O(n^{O(\log\log n)})$ time algorithm this one. this is slightly stronger assumption that NP completeness. If you just want to assume NP completeness then we have slightly weaker result known this is this theorem. There exists some constant c greater than 0 such that if there exists a c lon n. approximation algorithm for the unweighted set cover problem then p equal to n p ok. So, there exists some constant c beyond which or some constant c greater than 0 such that if there is a $c\log n$ factor approximation algorithm for even unweighted set cover problem then P=NP. That means, you can have the approximation ratio is theta log n. So, you can have a $O(\log n)$ factor approximation algorithm and there is a lower bound of big omega of log n factor approximation assuming $P \neq NP$ ok. Now, using the f factor approximation algorithm we can show a two factor approximation algorithm for vertex cover.

the standard vertex cover to set cover reduction with the f factor approximation algorithm for the set cover problem. gives a two factor approximation algorithm for weighted vertex cover. where vertices have weights. It turns out that these also based known definitely and tights assuming something called UGC conjecture. So, theorem assuming UGC conjecture unique games conjecture assuming UGC if there exists an alpha approximation for the vertex cover problem even unweighted for some constant this is very important constant.

alpha greater less than 2. There exist algorithms better than 2, but that is $2-o(1)$ is 1 by some function some growing function of n. This sort of algorithm exists for vertex cover, but for if you have an alpha factor approximation algorithm for some alpha less than 2 which is constant then P=NP. ok. This is very important this is assuming UGC which is stronger assumption that that $P \neq NP$. So, if you if you want to assume only $P \neq NP$ then we know weaker result that if there exists an alpha factor approximation algorithm for vertex cover even unweighted for some again constant $\alpha < 10\sqrt{5} - 21$ which is approximately 1.36, then we have P equal to So, these sort of results show that you know the approximability of set cover problem is quite understood assuming UGC there is no better than 2 factor approximation algorithm for vertex cover there is no $2-\epsilon$ factor approximation algorithm. for any epsilon constant greater than 0. This in turn implies that you cannot have an approximation algorithm alpha factor approximation algorithm for set cover unweighted set cover for any α constant alpha strictly less than f ok. So, let us stop here. So, we will continue from here next class. Thank you.