Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 01

Lecture 02

Lecture 02 : Deterministic Rounding of Linear Program: An Approximation Algorithm for Weighted.....

Welcome. So, in the last lecture we have seen what is an approximation algorithm and what is approximation ratio and various ways to tackle NP complete problems. So, from today onwards we will delve into designing approximation algorithms for various problems. So, let us start. we will look at the set cover problem to demonstrate our techniques for designing approximation algorithms.

So, what is set cover problem? We will use this problem many times in this course to demonstrate how a particular technique for designing approximation algorithm is applied. So, what is this problem? So, the input is a universe U with n elements a collection $S_1, S_2, \ldots, S_m$ of subsets of U with costs $c_1, c_2, \ldots, c_m$. So, each set has a cost and these costs are greater than equal to 0. ok and that is it and the goal output set cover of minimum cost. minimum total cost.

What is a set cover? So, a collection of sets $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ everyone belongs to $\{S_1, \ldots, S_m\}$ is called a set cover for U. union of $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ equal to this universal set U ok. And the cost of this set cover is the sum of the costs of these sets which is $c_{i_1} + c_{i_2} + \ldots + c_{i_k}$ ok. So, in this problem we are given a universe and a collection of sets each with some cost and we need to compute a set cover of minimum cost. So, it is a well known fact and you can also prove it yourself take it as a homework that the set cover problem is NP complete.

Now when I say the set cover problem is NP complete recall the NP framework is defined only for decision version of the problem. On the other hand the set cover problem as described here it is an optimization problem. So, when I say that the set cover problem is NP complete that more formal or correct statement would be the decision version of this set cover problem is NP complete. What is the decision version of this problem? the input is as usual the universe U and the collection of sets and also a target cost K and the question is does there exist a set cover of total cost at most K and then by defining this way it will be a yes or no answer and it will be a decision version of the problem that version needs to be can be shown that it is NP complete. Here also sets have weights if all sets have same weight say same cost equal to 1, then this set cover problem is called unweighted set cover problem and even that version is also NP complete.

Now, we will see an approximation algorithm for this weighted set cover problem let us change this costs to weights. $W_1, w_2, \ldots, w_m$ here $c_{i_1}, c_{i_2}, \ldots, c_{i_l}$ ok. Next we design a polynomial time approximation algorithm for set cover problem and for that we use what is called a linear program. Now what is a linear program? Let me give a brief very high level idea, but if you do not know linear programming please understand learn linear programming before go ahead with this course. So, and there are various textbooks for example, in the appendix section of the approximation algorithm book of Williams and Shmoys and has a good enough exposition for linear programming which is enough for our purpose.

So, what is a linear program? Let me give a very high level idea. We have a set of decision variables or simply variables which can take. real values ok. Then there is a we will see an example shortly there is a system of linear equal linear inequalities two kinds of linear inequalities are allowed less than equal to or greater than equal to strictly less than or strictly greater than is not allowed for linear programming. Linear inequalities that the decision variables must satisfy and then and finally, there is a linear objective function which we want to maximize or minimize ok. So, what we will do we first and there is another programming which is called integer linear programming. And, here it is same as linear programming and on top of this. So, we can specify some variable to take only integral values integer values ok. And very important fact linear program can be solved in polynomial-time.

However, solving an integer linear program ILP for short is NP complete. So, in most cases we will see that our problems can be formulated as an integer linear program and then what we will do we will relax something called relaxation of integer linear program to linear programs and crucially use the fact that linear programs can be solved in polynomial time to design our polynomial time approximation algorithms. So, for that let us see how the framework is used using set cover. So, formulating set cover as an integer linear program ILP. So, in set cover problem what is the goal? The goal is to pick sets.

So, we have sets $S_1, S_2, \ldots, S_m$ and we will pick sets so that their union is U. So, we have variables say $x_1, \ldots, x_m$ where $x_i = 1$ encodes the fact that S_i is picked in the solution and $x_i = 0$ indicates that S_i is not picked in the solution. So, here we will define each x_i is takes only integral values 0 or 1 and this sort of constraint is allowed only in integer linear programs. In linear program this sort of constraints are not allowed this for all $i \in [m] = \{1, 2, \ldots, m\}$ ok. So, what is our goal? Our goal is to minimize the total weight of the sets covered.

So, minimize the goal in terms of these variables the goal is to minimize the sum of the weights of the sets picked. that is $\sum_{i=1}^{m} w_i x_i$. ok, and what is the constraint thus the sets that we pick that must form a set cover their union must be u. Now, how do we write that constraint as a linear constraint using greater than equal to or less than equal to. So, that we can write as that the same constraint is equivalent to for every element in the universe all the sets that contain that element at least one such set should be picked.

So, let u be these are the elements $e_1, e_2, \ldots, e_n$. I need to look at the sets where e_i belongs. So, $j \in [m]$ such that $e_i \in S_j$ ok. Those sets at least one of them must be picked that means, at least one of those x_j 's must be set to 1 which is equivalent to writing this is greater than equal to 1. So, this is the constraint.

So, what is the integer linear program? ILP formulation of set cover. minimize $\sum_{j=1}^{m} w_j x_j$ subject to the constraints. What are the constraints? For all $i \in [n]$ $\sum_{j \in [m]: e_i \in S_j} x_j \geq 1$ ok. And what else each variable $j \in [m], x_j \in \{0, 1\}$.

So, this is the ILP integer linear programming formulation of set cover that means, whatever the minimum value here is that is the minimum value of the set cover. This is not an LP because this is the last constraints this is not LP constraint. ok, but this ILP cannot be solved in polynomial time. So, what we do we relax the ILP relax the ILP to an LP. what is the relaxation? The constraints where the constraints where which are allowed in LP that means, this the optimization and this constraints will stay as it is minimize $\sum_{j=1}^{m} w_j x_j$ subject to $i \in [n]$ $\sum_{j \in [m]: e_i \in S_j} x_j \geq 1$ and we replace this constraint $j \in [m], x_j \in \{0, 1\}$ to the best possible linear constraint which is for all $j \in [m], 0 \leq x_j \leq 1$.

So, this is the relaxed LP. Now, you can see or this is also you can take it as a homework that this part that we can get rid of we can delete. $x_j \leq 1$ for all $j \in [m]$. These constraints we can get rid of without affecting the value of optimal solution without affecting LP opt. which is the value of an optimal LP solution ok.

Also you see that every feasible solution of ILP is a feasible solution of LP. So, observe that every feasible solution of LP feasible solution means which satisfies this is every feasible solution of ILP sorry that means, which satisfies this constraints is also a feasible solution of LP is also a feasible solution of ILP. Hence, $ILP-opt \geq LP-opt$ ok and $ILP-opt=opt$ because we are minimizing in LP we have a larger set to minimize. Now, let us see the first approximation algorithm we solve the relaxed LP let $(x_j^*)_{j\in[m]}$ be any optimal solution ok. This we can do in polynomial time. Now, our goal is to output a set cover. So, define say Z to be you know those sets S_j such that $x_j^* \geq \frac{1}{f}$ Now, what is f? f is the maximum frequency of any item, f is the maximum number of sets any element in U can belong. So, here is a easy exercise show that Z is a set cover.

why Z is a set cover? The reason is that let me tell it for informally and then you prove it formally. The reason is that the number of terms here is at most f because each element appears in at most f sets. Now, because $(x_j^*)_{j\in[m]}$ is a feasible solution there must exist for every $i\in[n]$ there must exist at least one x_j^* where $e_i \in S_j$ whose value is greater than $\frac{1}{f}$ otherwise this sum will be strictly less than n strictly less than 1. So, this value is at least one set have value greater than equal to $\frac{1}{f}$. hence for every element $i\in[n], e_i$ we must we are picking at least one set that contains that element.

So, that is why it is a set cover. Now, why it is a it is a good or approximate solution? So, for that let us alg is the by alg we denote the we denote the cost or the value of the solution output by the algorithm which is summation weight j of the sets that the algorithm outputs this is this. Now, this is less than equal to $f\sum_{j=1}^{m} w_j x_j^*$. Why this is true? Because $x_j^* \geq \frac{1}{f}$. So, $f x_j^* \geq 1$.

This is less than equal to $f\sum_{j=1}^{m} w_j x_j^*$, but this is nothing, but LP opt F times LP opt and we have seen that LP-opt which is less than equal to ILP-opt which is equal to opt. So, we get LP opt is less than equal to ILP opt which is opt. So, this shows that the value of the weight of the set cover is at most f times the weight of the optimal set cover. So, hence our algorithm has an approximation factor of f ok. So, let us stop here. Thank you.