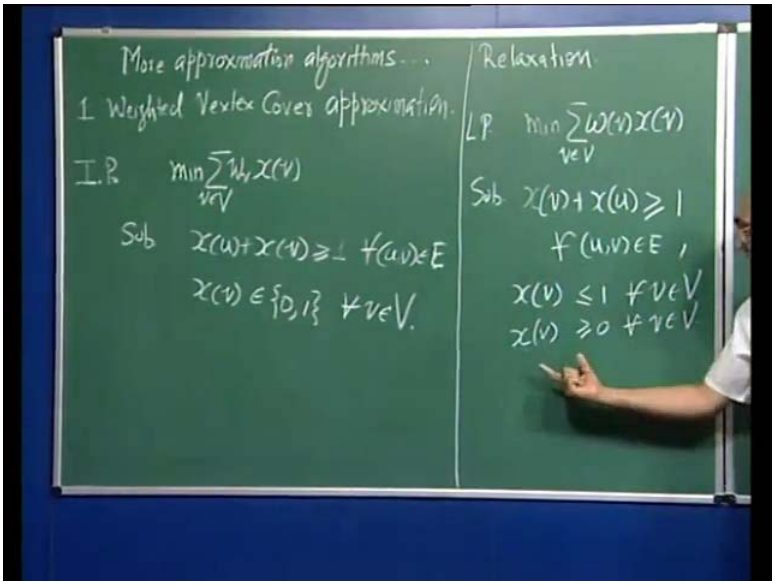
Computer Algorithms - 2
Prof. Dr. Shashank K. Metha
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 33
Approximation Algo II

(Refer Slide Time: 00:21)



Hello. So, we will continue with more approximation algorithms. So, today's first problem is weighted vertex cover approximation. Recall that we discussed the vertex cover approximation in the last lecture; and just to remind you, this is given a graph, a vertex cover is a subset of vertices such that every edge has at least one of its end points in this set. Now, the only twist in this case is that every vertex has a weight and our objective is to approximate the smallest cover with collective way to the smallest. So, the purpose of doing this problem is we are going to use the technique which has a many general applications in approximation.

So, one way to describe this problem is writing an integer program and here we are trying to minimize W v x, V x v will represent here a characteristic function which is 0. If vertex is not in the vertex cover v is not in the vertex cover, otherwise it should be 1. Hence, this weight will be actually the cumulative weight of the vertex cover and this we want subject to the following that x u plus x v should be greater than or equal to 1 for all edges u v. This ensures that at least one of the two end vertices of this edge is in the

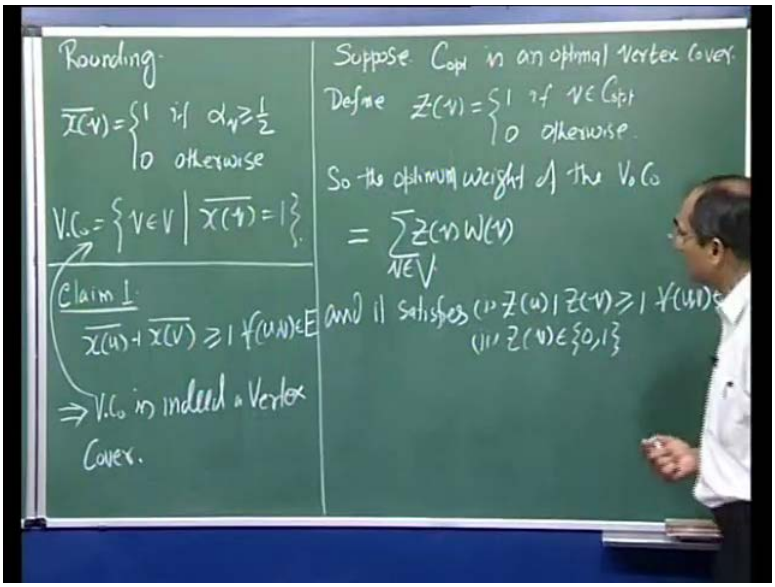
cover. Of course, we want to ensure that x of v is either 0 or 1 for every v in the vertex edge.

This guaranties a meaningful value for these functions. If it is 0, then it is not in the cover or else it is present in the cover when it has value 1. So, many problems can be written in this kind of linear program where in the discrete domain we often end up having an integer program. An integer program simply means that the values of the variable should be integral, such programs are known to be no hard. So, we do not expect it to be solvable in polynomial time. So, what one does is one relaxes this relaxation, simply means that we will drop this integrality condition will allow values, fractional values any value between 0 and 1 and then it becomes a linear program.

The linear program corresponding to this is to minimize subject to same condition here, for all u v and E, we replace this condition by x of v should be less than equal to 1 for all v in vertex at V as well as should be greater than equal to 0 for all v in V. So, this is almost the same, but now we have fairly meaningless values to these variables, because they can vary anywhere from 0 to 1. But, on the positive side is a linear program which is known to be polynomial solvable, although in lecture series we had discussed solution which was not polynomial time.

Recall that we had a discussion of simplest algorithm for this, but there are polynomial time solutions for this, so we expect this that we can solve this out, then what do we do? So, then the next step is to extract what it is based on some information that we have and for vertex cover, the vertex cover must be meaningful. There is no approximation about the vertex cover; the only approximation is that it may not be the minimum in terms of this.

(Refer Slide Time: 06:40)



So, the next step is rounding and what we do is we define x v bar as 1 if our, so lets let us assume that this ends up giving us some solution x v solution is x v equal to some alpha v. This is a number between 0 and 1. So, we will say alpha v is greater than equal to half, zero otherwise. Now, this satisfies in this integrality condition and we are going to interpret a vertex cover as v of V such that x v bar is 1. Well, we have to show that this is indeed vertex cover for 1 and then you have to see what is the weight of this cover and how does it compare with the optimum solution? So, these are the two steps we have to now do.

So, firstly notice that for every edge this plus this is at least 1, so claim 1 is that x u bar plus x v bar is still greater than equal to 1 for all u comma v and E. The reason is very simple that whenever the sum is at least one of these at least must be half. So, one of the two values alpha v or alpha u has to be half or more. Hence, the corresponding variable will pick value one in this case; hence this sum cannot be less than 1. What does that imply? It implies that for every edge, one of the two end points, at least one of the two end points must be in this cover, because that is what we do. We select those vertices for which x v bar is 1.

Hence, this implies that vertex covered computed here is indeed a vertex cover. So, we certainly have that in object that we have wanted to compute, but how good is it? Well,

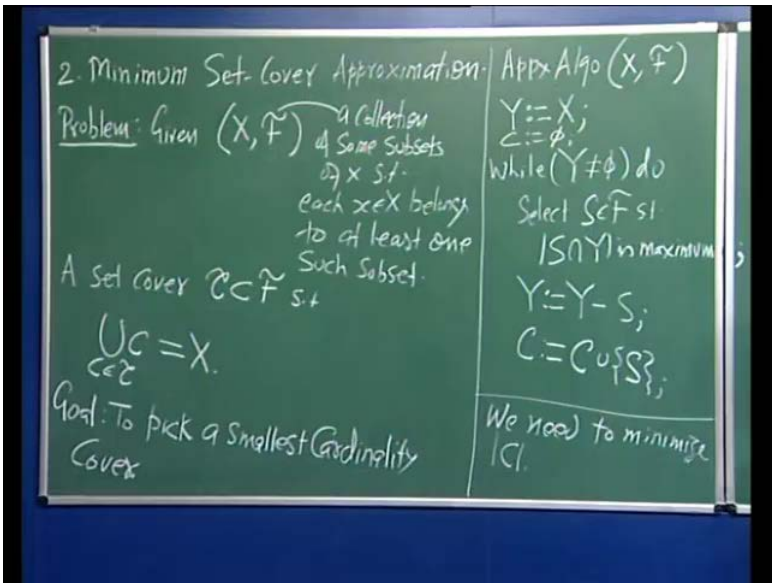
let us suppose C opt is an optimal vertex cover, so let us now define z v to be 1. If v is in C opt and 0 otherwise. So, we defined these characteristics functions for those for that set. So, the optimal weight of the vertex cover that is the weight of this optimal vertex cover is sum z v and w v. So, this is the weight of the optimum, weight cover than check this in contact of the integers program written, this being a vertex cover has to satisfy. It satisfies z u plus z v greater than equal to 1 for all u v in the cover. It also satisfies the fact that z v, all z v belong to the set of 0's and 1's.

So, everyone is either 0 or 1, hence it is not a surprise that this is the optimum solution of the integer program that is z v is an optimum solution of the integer program. Now, let us get back to this linear program, it is clear that the solution of integer program is automatically a solution of linear program, because every variable takes value 0 or 1, hence this is true. This condition is identical and we are optimizing the same object function. So, it is clear that the optimum solution of the linear program has to be less than equal to the optimal solution of the integer program.

So, what we do is we realize that the optimum solution of L P has in terms of the object function is less than equal to the optimum solution of the I P, because every solution of I P is a solution of I P. Now, let us just write down this the solution of I P sum w, the object function value was this which from this in equality is greater than equal to sum w v alpha v; this was the solution of the linear program. Now, let us go back and recall what we did is that we picked up a those vertices which had this value greater than equal to half.

So, this is greater than equal to one half of sum w v x v bar, notice that every value that was half or more was pushed up to value 1, but there might be some other values which are less than half which were set to 0. So, this value cannot be less than one half of this value, well this is the weight or this is the weight of the vertex cover computed by the approximation algorithms. Hence, the object function hence phi computed is less than equal to two times object function of the optimal solution. So, once again the performance ratio is a constant here two and that was the same when we considered the case without weight.

(Refer Slide Time: 14:21)



But, the thing to learn here is that this technique is a very powerful one and in used in many cases second problem today is the another mp hard problem called set cover problem. We would like to compute an approximation to this smallest set cover.
I will begin with the definition of the problem, suppose we are given X and a family of subsets a collection of some sub sets of x, such that each x e x belongs two at least one such subset. So, we may not have all the subsets of x but, sufficient in a way that every member of x is present somewhere.

Now, a cover, a set cover is a subset of F such that union c in of the sets in C is equal to whole of X which means we want to pick sufficient members sufficient number of sets from F such that their union covers every element of X. The objective is to compute these smallest in terms of cardinality of smallest such cover, so goal is to pick a smallest cardinality cover. Well as I told you this is also a well known n p complete problem and we do not except that this will be solvable polynomial time unless p is equal to n p. So, we would like to approximate the cover, well this time we will see that the approximate the performance ratio is not a constant.

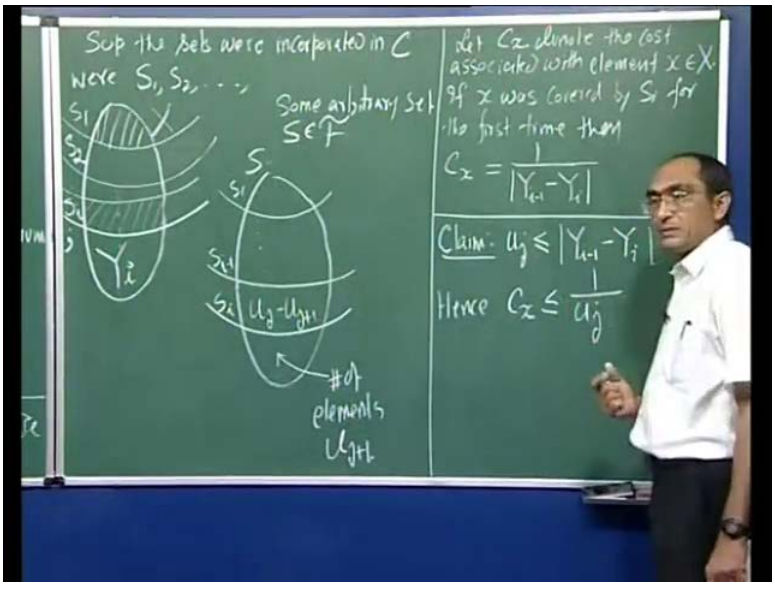
So, I will give a very simple algorithm for approximating a cover, it is the analysis which is interesting, so let us write down the algorithm. So, approximation algorithm given x and a family of a sets of x, I will take Y to be a set variable initialize at X and idea is simply to pick, it is a greedy algorithm, it picks that member of x which covers

maximum number of new elements, which are not yet covered in this head. So, while Y is not empty. I initialize my cover to be empty, initially this is the cover, select s from F such that S intersection Y is maximum y will always be the set of element not covered initially. Of course, everybody is not covered so pick that set from F which gives me maximum mileage in covering the elements not covered yet and then vary move than from Y is Y minus S.

We put that set in the cover this is it, this is a very simple trivial algorithm, so we want to know how good it is in terms of the optimum solution. So, let us do one thing, notice that every time I pick a new set, because if I have I pick up one of the old sets which had already been put in C. Then I will be really having empty set here. So, it is going to pick up a new set every time I am inuring a cost of one, because our object here is you know our goal is to minimize the size of the cover set.

Our goal, we need to minimize C cardinality, that was the goal so what is happening is in this algorithm every time I picked up a set I have a incurred one additional cast. If I do that than in my analysis, I am going to distribute that cost over the uniformly over those elements which have newly been covered these are the new elements which are covered, so instead of taking the cost associated with S. I am going to put that unit cast on those elements which were newly covered. In this, now let us try to let us try to define and deter the rather the cost of the elements in some random set as so.

(Refer Slide Time: 25:22)



Let us just say suppose the sets were incorporated in C were S 1 S 2 and so on these were the sets which actually went in to the cover, think of this is my set X, the universal set x. So, these were covered by S, answer were covered by S 1 union, S 2 as a whole. So, actually might confuse, so let us say these were covered by S 2. I put down here it is perfectly possible some of these elements are also covered by S 2, but that is not so important. See this is so the last one is a S. So, let us say after these elements are eliminated this is y I.

So, the value of y after I sets have been put in to the cover. Now, I am taking some arbitrary set S belonging to the cover. This may or may not belong to this the cover sets. So, this is that set s and we will see how this sets cover the elements of S also. So, this is S 1, S I may minus 1 so on, now let me denote this, the cardinality the number of elements left in S, after removing those covered by S 1 through S. We will denote that by u j plus 1, so these are u j minus u j plus 1 is in it, because everything after this is u j everything up to this is u j plus 1.

So, this is what is covered when we selected S for the cover, one more notation. Let C x denote the cost associated with element x in x, what is exactly this suppose x was first time covered by S. At that time all these were the new elements that were covered and the price we paid to cover them was to incorporate S i. Hence, we incurred 1 unit cost, so if x was covered by S i for the first time, then that 1 unit crossed evenly distributed among these elements. So, if there are ten elements in this, then each will carry one tenth units of cost; that would be C x. So, what we know is that C x, then C x will be how much? Well, it will be 1 over the cardinality of this section, which is Y the set here is Y i minus 1 minus y I.

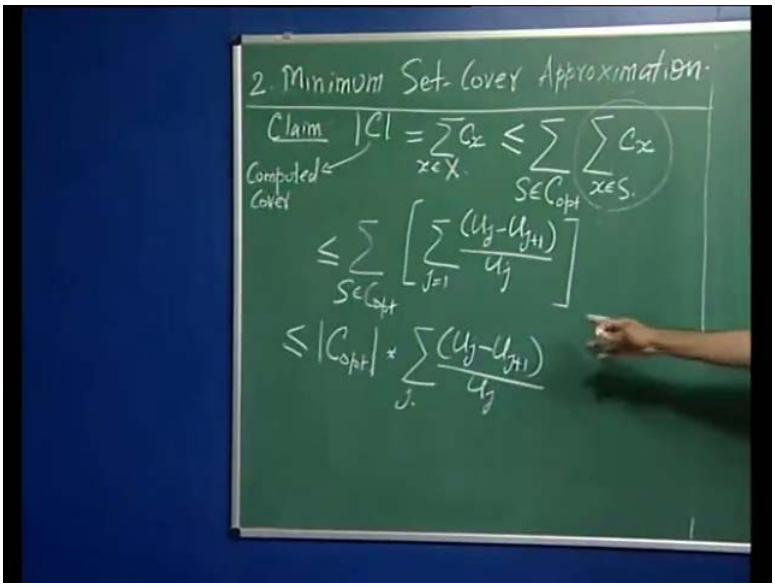
So, this is 1 over cardinality of Y i minus 1 minus Y I, is that clear, these were the freshly covered elements when S i was incorporated and x was one of them. They all got collectively a cost of 1. So, we distributed 1 over cost this much to each of them, hence x also got the cost C x namely this. Now, with this we are going to make one observation, well before S i was chosen, what we did is we cheeked with every set how much is that set is going to cover from the remaining elements, indeed S i was promising this much. This vary set S at that time had we taken S if S is S I.

Then of course, there is no issue, but of course, different from S I, then S was promising this much coverage because, so many elements are already covered. But, S was going to cover this many additional elements. But we chose S i, so even if this is same as S i, this is the cardinality of this set has to be less than equal to the this number or the cardinality of this set. So, claim is that and this set is of course, u we are saying that U j is less than or a equal to at best Y i minus 1 minus Y i. So, let me repeat quickly if I had chosen S instead of S I, I would have covered U j new elements, I chose to cover I chose to incorporate S i in the cover and that covered this many new elements.

So, this had been less than this, otherwise I would have chosen rather than this set, hence this inequality. So, from this inequality we conclude that C x has to be less than equal to 1 over U j. Now, given this can I determine the cumulative costs of the elements of this set? Notice that this is absolutely any set we did not particularly chose, the element belonging to be covered. It could be absolutely anybody our x belong to this section because it was first time covered by S i. So, each of these elements have a cost the same as of x. So, sum C z where z belongs to set S, you mean the total cost of the elements of S, well this has to be this times this.

As I am looking for the upper bound at best and that makes it sum U j minus U j plus 1. These many elements each had a cost of 1 over U j at most and our j start from 1 and go on for all the values of that cover. So, this is one useful inequality that we have, now let us see how we are going to estimate the size of the cover we have computed here. Now, one claim that is if you remember that how we had done the costing is that the size of our cover.

(Refer Slide Time: 37:16)



The cover that we have computed, cover is precisely sum C x, x belonging to X, why the reason is every time I picked a set, I covered some new members of capital x. The cost 1 unit of cost I had incurred for picking that set was evenly distributed among all those members which are those covered newly from this. So, if I add up the cost of, remember that C being a cover every member of X is already covered, everybody is covered exactly once for the first time for some reason. Hence, if I just add up the cost of every element of x, I am getting precisely the number of sets put inside is that, now let us say C opt is the optimum cover.

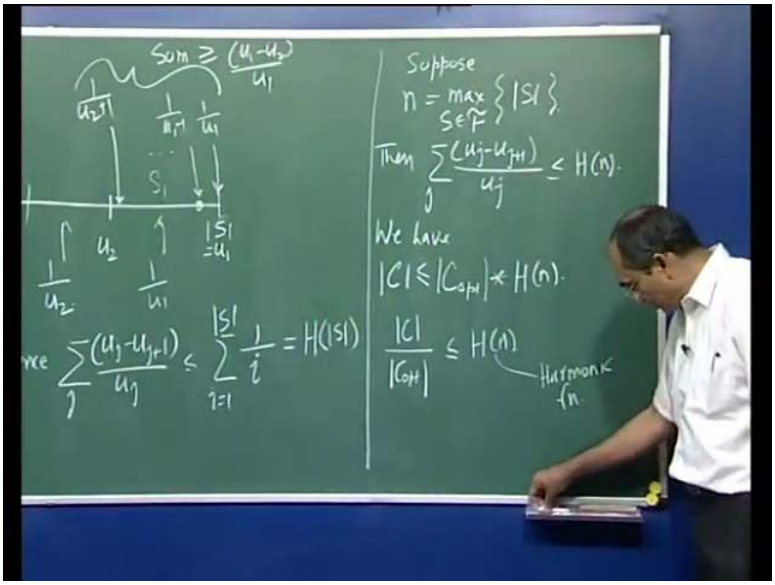
Now, optimum cover is a cover, hence it also covers every member of X, so I am just rewriting this in the following fashion S belonging to the C opt. So, this is the optimum cover, every member of this if you take and union them, you get x, so this x, sorry, we have C x, So C x, x belonging to S. So, every member of this set I take, I add up their cost and then if I add up over all the sets in C opt, I have definitely taken care of every member of X, but it is also possible that I have taken care of some elements more than once.

That may happen and hence this is an upper bound to this, some elements of X might be present in more than one set. So, that will covered more than once, all the costs are non negative, in fact all the costs are positive. So, this is going to exceed this and then I am going to use that inequality. We had for this S is just any set happens to be a member of

the optimal cover. So, we can see this sum S belonging to C opt and for this I have sum j going from 1 to whatever to be the number of sets in the cover U j minus U j plus 1 over U j. This we have derived earlier, the bound for and after bound for this sum.

Now, this number is independent of this, it does not matter what I say. Hence, we can say this is less than or equal to the size of C opt times sum U j minus U j plus 1 over U j. For one thing, you notice that you already have a situation that you can compare C with C opt, this is your performance ratio. So, find out what is this, so let us start to estimate this of course, there is one thing that I actually did not tell you correctly; that this actually depends on this. Because these are the numbers of elements of S which were covered between S i minus 1 and S j minus 1 and S j. But, we will avoid that dependence very soon will get over that. So, let us try to understand what this number is.

(Refer Slide Time: 42:38)



So, let us say we write down the elements of S as element number 1 2 and so on. Over here, we have element number mode S, so I am just looking at our definition says that this is U 1, U 1 is the elements that are not covered and nothing was taken into an account. Then at some point, here this is U 2. So, these were covered by S 1, so these were, this is S 2 and so on. So, let us say somewhere over here we have U j plus 1 and here it is U j. So, these elements up to this these were covered by S j, now what this is saying is that for each of this, you charge 1 over U j. Then you will get this term

associated with this section, you charge 1 over U j. Further, so you will charge here 1 over U 1 over here, 1 over U 2 and so on.

So, we will say that we are looking for a upper bound for this, so we will just modify the charges as follows: will charge for this 1 over U 1 for this, will do that with 1 over U minus 1 and so on. For the last, this will be 1 over U 2 plus 1 because after U 2 this was the element. Now, notice that each of these numbers is less than each of the denominators is less than 1. Hence, these numbers are greater than 1 over U 1, so if I just add up these numbers, their sum has to be greater than equal to U 1 minus U 2 divided by U 1. Just take a look at this term, the first term was U 1 minus U 2 by U 1. This sum was viewed as associating 1 over U 2 of each of these elements, instead I am associating slightly larger values, but that helps. Now, this is 1 over U 1, this is U 1 minus 1 and so on, 1 over U 2 plus 1 for this. Then over here will be 1 over U 2, 1 over U 2 minus 1 and so on.

Hence, j U j minus U j plus 1 over the U j is less than equal to sum 1 over i, i going from one to mode of S because this is the total number of elements in S such sum is known as harmonic function of A. This is the discrete variant of the log of a number because in log you integrate 1 over x. So, this is an upper bound for this expression. Suppose, E n is the maximum over all S and f for mode S maximum cardinality of members of F, then see I wanted to avoid this dependence on S, so I am just replacing this by n. Then, we have sum j minus U j plus 1 divided by U j and that is now less than equal to H of n.

Hence, we have mode of C equal to mode of C opt into H of n, means we have there is an equal to H of n harmonic function and a performance ratio is actually this H n. This is the first time we have seen a case when it is not a constant, when the last lecture. In the next one, I am going to give you an example of a pitas a polynomial time approximation scream. These were actually algorithm, we are giving as a fixed performance. But in pitas you can achieve arbitrary performance and that is where we will close this lecture series.