

Design Verification and Test of Digital VLSI Designs
Prof. Dr. Santosh Biswas
Prof. Dr. Jatindra Kumar Deka
Indian Institute of Technology, Guwahati

Module - 3
Logic Optimization and Synthesis
Lecture - 3
Two level Boolean Logic Synthesis - 3

Welcome to the last and the third lecture on module 3. In first 3 lectures, I mean so, we started in this three part lecture, is the idea of two level Boolean logic synthesis. So what was the idea? The idea was that, we have taken cycle second of specification from high level synthesis tools. We have generated the RTL and from the RTL we represent them as Boolean functions, and then we wanted to represent them as a in terms of minimal terms of gates.

So if you think in terms of Boolean functions, then it will be the minimum functions have to be represented in such a form? So the minimum number of terms and minimum number of product terms and in each of the product terms, the number of literals should be equal and should be as minimum as possible. So, that they can be represented in so, that they can be represented in minimum number of gates.

Then what we have discussed? Then we have founded that any Boolean function can be represented as prime implicants integers. So what are the prime implicants? Prime implicants are the product terms in which the product term is not totally included in any other term; that means, prime implicants are all necessary to form a function.

Then the next function what we saw, when you have seen that some of implicants essential implicants, that are in minterm or some term or that actually contain some which is not available in any of the implicants; that means, so they are prime implicants are essential implicants, implies that representing the Boolean function that essential prime implicants must there. Then next thing what we have to do? We have to find out, we have to select the subset of the prime implicants such that they cover the function. So what you mean by covering of a function?

Covering of function means that for all the minterms of functions, that the prime implicants contain it. So this is the rational what we have seen in the last two lectures, and the last lecture of the three part, we will see how we will find out the subset of the

prime implicants that covers the entire function and we have also said that when we mapped the problem for that minimum two level Boolean logic synthesis, that is implementation using minimum number of gates. So once we have found out the prime implicants and finding out the minterms and we make the constraint matrix that we have seen in the last class and it remains. Just find out the unit covering problem or a subset, subset finding kind of a problem or you have to find out the subset of the prime implicants products, such that they all cover the minterms in future.

So now no longer remains the problem of VLSI or circuit. It will become the problem of synthesis. When you have to find actually which is the unit covering problem, such that we have to find out the minimum number of prime implicants or prime implicants of minimum weights, whatever when you can forget about the calling them prime implicants, finding minimum number of columns of the constraint matrix such that it covers all the rows and that should be, and there should be multiple solutions for that. And you have to take the solution which is minimum cost. So you can give cost of each of the columns, for example, if the number of terms in the number of literal in the prime implicants, which have a high weightage of so on.

Then what we say found out that, they represent the whole problem that constraint matrix and we found out the matrix is very large and it can be preprocessed to bring it into the reasonable size, and you can apply the unit covering problem or prime subset problem. So how do you do? That first you have to find out the essential columns so if you have essential columns you have essential implicants, you have to take them, you have to include those rows. By including those rows you have to find out the rows, rows will have already in covering. So that you can eliminate those rows and column has been taken into account. So, next you find out the row dominance and column dominance by applying the row dominance and column dominance.


What you will get? Some more rows are getting covered and some rows can be eliminated, and some columns can be eliminated by using the concept of row dominance and column dominance. Finally, we will have the small matrix where more preprocessing can be done to minimize it, there you will try to apply some algorithms to find out which is the minimal subset of a h . Those h columns to be taken to cover the matrix.

(Refer Slide Time: 04:19)

Branch-and-Bound Algorithm for selecting cover

The unate covering problem can be solved efficiently using Branch-and-Bound Algorithm. Now we will present the scheme and then illustrate the same with examples.

- There may be multiple optimum solutions to the problem and our intention to find one of them.
- The search space can be defined as a subset of selected columns; if there are n columns then solution space is $O(2^n)$.
- In the branch and bound procedure for this problem the solution space is enumerated in form of a binary search tree.
- A node corresponds to a column and the left (right) edge represents the column being (not being) considered in the cover.
- It may be noted that for all columns there may not be a node in the tree, because considering some column may result in elimination of some other due to dominance or inclusion due to essentiality.



So that, we will be basically landing in that algorithm, how to do that? We all have the technique which will actually call the branch-and-bound which will see that is the branch-and-bound. To find selecting a algorithm to find the cover that is what the at the end of this class how do you do this?

(Refer Slide Time: 04:44)


Selecting a Subset of Primes

The switching function (P_2+P_3) for the 1st row evaluates to one if either $P_2=1$ or $P_3=1$ or both $P_2=1, P_3=1$.

We interpret $P_i=1$ as "column i is selected," and all j "rows covered" for which the matrix has $c_{ij} = 1$.

We can proceed similarly for the other rows. The expressions thus obtained are switching functions that must all be 1 for a solution to be valid. Hence, their product must be 1. We can therefore write the following equation as an equivalent to the constraint matrix $(P_2+P_3)(P_1+P_2)(P_1+P_4)P_3P_4$

P1	P2	P3	P4	Result
0	0	0	0	0
0	0	1	0	1
0	1	1	0	1
1	0	1	0	1
1	1	1	0	1
1	1	1	1	1



So before you go there, go what exactly branch and bound. So we today solve those branch and bound. So before going to the exact algorithm just we will give the idea behind that, for example, we can represent as you have seen in the last class lecture, that

the unit covering problem we can represent using the equation like this. So; that means, p 1, p 2, p 3, p 3, p 2 all these. So you can have all the 1s p 1 1 to 1 equal to all or p 1, sorry p 4 all of them dot, dot all of them which this can be one solution this can be one solution which can be definitely covered. There can also be a solution to that which is not and other solution is that all of them can be 0s.

If all of this are 0s, obviously this is not a problem but, you can think all the possible values to the all the columns. It can be all 0s to all 1s, obviously all 1s is a solution, all 0s is not a solution but, you have to find out the proper subset, for example, 0, 1; that means, select first, third and sixth, so forth there can be a solution. So this equation or the unit column can also be represented in terms of the tree.

(Refer Slide Time: 05:48)

Branch-and-Bound Algorithm for selecting cover

The unate covering problem can be solved efficiently using Branch-and-Bound Algorithm. Now we will present the scheme and then illustrate the same with examples.

- There may be multiple optimum solutions to the problem and the intention to find one of them.
- The search space can be defined as a subset of selected columns; if there are n columns then solution space is $O(2^n)$.
- In the branch and bound procedure for this problem the solution space is enumerated in form of a binary search tree.
- A node corresponds to a column and the left (right) edge represents the column being (not being) considered in the cover.
- It may be noted that for all columns there may not be a node in the tree, because considering some column may result in elimination of some other due to dominance or inclusion due to essentiality.

HPTTEL

That is if you consider each node in each in the first level you can consider the root node corresponds to the first column, second level nodes corresponds to the second column such that so forth. So the leaf nodes corresponds to one of the solution. So if you take the left most edges throughout for example, this is your root and you come down to this, this is your leaf node and you can think this all the left edges first one corresponds to PI 1 this one corresponds to PI 2 and this one corresponds to PI 3 and this to say last p n.

So if you take the left edge you say that I have not taken PI 1, PI 2 and so on this will represent all zero in this cases last one that one is also PI n and this one will corresponds to all the 1s for this you have taken all the edges. In other words what I am saying is

solution problems solution speaks can we represent in terms of trees and we have to find out we have to take the path to a leaf. So not all the leaf paths will be valid because it will not cover it so, some of the valid nodes you have to take, so it corresponds to leaf nodes inside it will be multiple solutions and you have to take which is having minimum number of weight and how are weights assigned weights are assigned in term of you can say that number of weights can be number of prime implicants involved all the all the prime implicants are compactable with and you count them the solution which is having the minimum number of 1s will have all 0.


Means no prime implicants is selected all one means, all the prime implicants are selected. So you can think that the more number of prime implicants selected you add 1 to the weights 1 prime implicants means, weight 1, 2 prime implicants means, weight 2 and so on. So you add up with the weights and you will get it so, solutions which will have the minimum number of implicants will be your solution and you can go for more finer way of thinking prime implicants can be individually put away like number of literals in the prime implicants like x is a prime implicant weight one this is a one c and t if x is a if f if prime implicant is a b c d it is having I mean that four kind of the thing. So you can devise new extra weights.

(Refer Slide Time: 07:31)

Branch-and-Bound Algorithm for selecting cover

The unate covering problem can be solved efficiently using Branch-and-Bound Algorithm. Now we will present the scheme and then illustrate the same with examples.

- There may be multiple optimum solutions to the problem and our intention to find one of them.
- The search space can be defined as a subset of selected columns; if there are n columns then solution space is $O(2^n)$.
- In the branch and bound procedure for this problem the solution space is enumerated in form of a binary search tree.
- A node corresponds to a column and the left (right) edge represents the column being (not being) considered in the cover.
- It may be noted that for all columns there may not be a node in the tree, because considering some column may result in elimination of some other due to dominance or inclusion due to essentiality.



So that some of the technique we will see for the basic core idea is that whole problem can be represented in terms of the tree multiple optimal solutions are intentions is to find

one of the; that means, we can find 3 or 4 solutions 3 or 4 prime implicant and all corresponds to 3 of the equal value and you are deeming to find out the number of each literals so that it can give you a final value.

So again if you can host another binary tree you can easily understand the order of the order of the complexity is exponential number of paths or number of key floats are 2 to the power of n. If n is the prime implicant so again this problem is not a simple problem it is in terms of complexes then it is a exponential order of complexity through solve the problem. So bound and branch algorithm if you take in a complete manner I mean that the you explore all the parts of the tree then exponential solutions will be there but, we can see this we can minimize this of course, worst case will be the problem solution is the exponential but, let us try to how to minimize the how complexity reduces update again we will come back to heuristic.

So in essence this what you say in essence this branch and bound tree is an exponential kind of problem. So what we will do? We will not try to explore all the problems put some heuristic and some of the branches we will stop at the some level and this branch we are not going to try and most it will not going to give a very good solution and at this point some heuristic are coming into picture and they are estimate what can be the solution? And if you explore the solution for this branch so if you can find out has this branch has no exponent meaning which is not going to give you a give a good solution, then do not explore the branch to go to another branch and try out at this point heuristic are coming into picture. So heuristic will tell you do not explore this parts it is going to give you a solution.

So that heuristic will actually estimate solutions for a branch in a very quick way. If i go on exploring the branches I will get an exact answer. So that will actually going to take a very long time for it. So what I will do is that for each of the nodes I can try to estimate what can be the possible solution or what ways of the solution? If I explore that path and that can be done in a very fast manner following the heuristic finally, solution may not be attractive I mean exactly will give you a value on best go that value you can decide you can explore the path or drop the path and try on another path. So complexity will come done if you are not exploring the all the paths of the tree. So that is the idea.

So branch and bound procedure for this the solution is the binary tree actually that is becoming a much unnecessary and complexity is exponential are actually as I told you correspond to a column left and right correspond to the column being considered in the corner column is not taken if you go for the standard this way is not taken. If I come out by this path but, at node or corresponding column is taken obviously you have noticed that for some columns there may not be node for some tree may not be a node in the tree considering that resulting in elimination of some nodes due to dominance inclusion or essentiality just like I told you always I there is a mis by take path by path by force for example, if I select if I select a node it has a column if I do not select a column for some of the nodes or some of the I mean some of the minterms or some rows or some of the minterms are represented in rows.

So they may become essential because there will be a 1 1 in a row all that will see in examples. So it will become an essential row that will be sorry that implies a column to the essential columns by eliminating some of the nodes some of the nodes and land up in some of the situations like some of the prime implicant will be essential prime implicant and that force you have to take corresponding prime implicant. So in that café there will not be two solution possible take any one of them that is some of the solutions so will solutions will start becoming invalid just for example, you take prime implicate one you say that node number one any doing that you eliminate some of the rows how you will eliminate that columns will have three columns.


So all the rows corresponding to three gets eliminated. So and in reduced matrix because these some of the other may corresponds to having a single one in that row that will make some columns h essential columns and then you have to by force you have to take the some of the some of the exponential space becomes flexible and become fixed. So that is what said here but, overall it becomes an exponential problem. So by using the heuristic I mentioned above try to reduce the complexity.

(Refer Slide Time: 11:49)

Branch-and-Bound Algorithm for selecting cover

Therefore, if we can determine that a given part of the search space (i.e., sub-tree rooted at a non-leaf node) does not contain any solution better than the one we have found so far, then we can avoid exploring that part of the search space altogether.

We start searching another branch by considering a column that was considered not to be taken in the cover. In addition, we may search other branch by not considering a column that was considered to be taken in the cover. Therefore, in branch and bound algorithm for unate covering problem, we resort to two basic ideas—organize the search space in from of a binary search tree and explore the branches of the tree.




(Refer Slide Time: 12:32)

Branch-and-Bound Algorithm for selecting cover

Therefore, if we can determine that a given part of the search space (i.e., sub-tree rooted at a non-leaf node) does not contain any solution better than the one we have found so far, then we can avoid exploring that part of the search space altogether.

We start searching another branch by considering a column that was considered not to be taken in the cover. In addition, we may search other branch by not considering a column that was considered to be taken in the cover. Therefore, in branch and bound algorithm for unate covering problem, we resort to two basic ideas—organize the search space in from of a binary search tree and explore the branches of the tree.



Now you can determine the given part of the search space that does not contain any solution better than the one we have found. So far, we can avoid exploring that part that search space altogether actually that the heuristic. So, what is the heuristic? Say if you can determine the given part a of the search space, a sub tree routed on the non-leaf node for example, it may have a lot of trees over here, it may have the intermediate node and that and if you explore down these paths no better solution currently which you are having better you can drop it all together. Now who can do that? Who can find out in a

very quick time exploring the sub tree that is where heuristic will come into play. So that what is the idea?

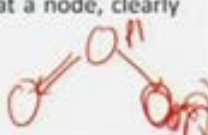
So basically what you want to say is branch-and-bound algorithm branch-and-bound algorithm got a we are branching and finding out, we got any better solution and backtracking other spaces other other other parts. So we resort the two basic ideas, organize the search space in form of a binary search tree and explore the branches of the tree and also I can say you can put a heuristic over here or what will the heuristic do? Which branch of the tree is not ideal not essential to it is not fruit full to explore and you try other paths, it has the three paths, search path in the tree binary explores the paths is actually branch-and-bound and complexity is very high.

So you put the heuristic over here and heuristic will you tell which you will tell it is not good to explore. So that you can branch and track back like now; that means, what? How you solve the problem?

(Refer Slide Time: 13:29)

Branch-and-Bound Algorithm for selecting cover

- For efficiency, we need some scheme to estimate the lower bound cost of a solution (given a constraint matrix) without fully exploring the search space.
- When we find that exploring a given path would lead to more expensive solution than expected, we retract to other branches without exploring the path under question.
- In other words, at any given node of the search tree, we have selected and rejected some columns. These columns are identified by the path from the root of the tree to that node. Hence, at that node we have a partial solution. If the cost of a partial solution (from that node) exceeds the expected solution at a node, clearly we can abandon that path and track back.



So what we do for efficiency? We need to some scheme to estimate the lower bound cost of a solution without fully exploring the search space that is the heuristic. So what is the heuristic? Heuristic is the scheme that will estimate the lower bound of the solution the mean of this much amount you have to take meaning this minimum amount of prime implicant you have to take without exploring the search space actually that is very important without that we have to estimate the prime implicant required; that means,

solution cannot go beyond this solutions are some of the approximate thing we have to have a front kind of thing lower bound.

So for example, minimum number of requirements will be three that is the exact value. If you are heuristic you might land up into a case minimum number of requirement is two minimum number is one and like that or your heuristic is like you can say minimum number of requirement is four or something like that so, some deltas will be there, with those deltas you have to find out you have to make out it will help you with exploring or some of places the when we find that exploring a given path would lead to more expensive solution and retract to other branches to the tree.


In other words at any given node of the search tree we have selected and rejected some columns right so, the root node you take left path, you have not taken the root you have taken the right path and root node then these columns are identified by the path from the root of the tree to that node. Hence, at a node we have a partial solution if the cost of the partial solution exceeds the expected value clearly we can a bound abandon the path abandon the path and track back. So what it is saying you take up the left path so left path is you do not take it.

So right path is you take it, next node is you take this you are taking the h column corresponding to the root node that each intermediate node you have a sub solution you have taken prime implicant a, b, c, d also both are taken not taken prime implicant a taken prime implicant b and c everything. Now your estimate algorithm will tell you that after your sub tree starting from the from this node root the sub tree here what will be actually sub tree here? Which is the sub tree? So what can be the minimum cost of the sub tree? So you should add up the minimum cost of the cost of the sub tree you have incurred still now like for example, here p 1 you should take this node; that means, you have already taken p 1 current value is 1 then you add up with the estimated value of the sub tree.

(Refer Slide Time: 16:43)

Branch-and-Bound Algorithm for selecting cover

- Branch and Bound then tries to establish whether a new best solution can still be found by proceeding from the current node in a different branch.
- The way of computing the lower bound depends on the particular problem. It is obvious that a careful choice of the lower bound criterion is important. Ideally, the criterion should provide an accurate estimate of the real minimum cost incurred in completing the current solution. At the same time, the computation of the bound should be fast. So, most of the lower bound estimation algorithms are fast heuristics, that may not provide a sharp bound.
- So the branch and bound algorithm for unate covering problem generally provides near optimal solutions. However, due to the lower bound estimation heuristics the execution time is low.



So if you add here and find out that solution is so much greater than the I mean estimated cost or cost estimation you have or the cost estimation worst than the present solution, then the stop traversing this path and try to take this path that is how you do it? So therefore, actually so what we do in branch-and-bound essentially we will try out different path and find out the estimate whether it is going to be good or not. If there is a possibility the solution is better than my solution, better than my quality of estimate I will use otherwise I will not use so branch-and-bound tries to estimate the best solution but, still be found by proceeding from the current node then I have to progress to other branch or you apply the same thing if you do not take the path, take the other path again you have to repeat the algorithm and find out whether the solution is greater than the path which is I have is discarded.


If again you have to go back with that discarded path and this path is most costlier than the other. Now a very important part of branch-and-bound is you have to pick a answer you need not go to exponential algorithm, then the way your computing the upper bound in a particular column more of your competition you can tell me this number of prime implicant is a very very important point in this case of solution or many of the other heuristic. You need to calculate the lower bound or an approximation or many of the heuristic algorithms. What you do is you think of this branch-and-bound are proposing a solution or mapping a solution by binary tree or a binary search tree very very important paradigm which is used in many other places of engineering and computer science.

(Refer Slide Time: 18:35)

Branch-and-Bound Algorithm for selecting cover

Before we turn our attention to the computation of the lower bound for the unate covering problem, we now look at the steps for the branch-and-bound algorithm for the unate covering problem.

1. Execute the lower bound computation algorithm on the constraint matrix. The "Initial Lower Bound" is the Lower Bound. Current Cost is set to 0.
2. Select or de-select a column P_i (i.e., a path in the solution space corresponding $P_i=1$ or $P_i=0$). If a column P_i is selected then add P_i to the cover and add cost 1 to the Current Cost. Apply reduction techniques (essential columns, row and column dominance) to the constraint matrix. For all essential columns, add them to the cover and increase Current Cost by the number of essential columns. If the problem is now reduced to a terminal case (because of selection of a column), then check whether the solution thus found is better than (or at least at par) the "Initial Lower Bound" (i.e., Initial Lower Bound \geq Current cost). If so, the cover is returned and algorithm is terminated.



What you explore the complexity? It will be the exponential complexity then you have to find out the algorithm which can estimate the minimum to traverse this path that is this lower bound complexity is very very important as well as this as accurate as possible as well as it should be very exhausting algorithm I am sorry highly complex algorithm you should be very fast to do it. So branch-and-bound algorithm is computation algorithm bound should be fast as well as accurate as possible. So branch-and-bound algorithm for unit covering problem generally near prog near actually I know that this are all contradictory but, accurate estimation it should be fast and contradictory.

So Unate covering problem generally provides a near optimal solution because this lower bound estimation execution is low and always accuracy is very high, that is why your branch-and-bound will give a near optimal solution and but, it may not give up the exact misconception instead of explaining all the branches of the tree and exponential algorithm you solution will always be solution will always be the most optimal solution because again I told you in had told you in the beginning in starting in every lecture I am saying that most of the design algorithms problem are exponentially in nature, here also we have proved that we can actually map the algorithm and binary tree which is again the exploring the 2 to the power of n nodes or number of levels you can think or number of prime implicant you can think therefore, each complexity is 2 to the power of n and if you map 100 or more than that. So you can never solve in a particle time therefore, you

are bringing this lower bound heuristic and that is how our solutions are going to be approximate.

So before we turn our attention to this computing this lower bound branch-and-bound let us look at the branch-and-bound algorithm to the Unate covering problem in harsh. Let us see what are the steps we do and we will see? How exactly you can calculate the lower bound? What you do execute the lower bound algorithm? And we assume that the heuristic nature available which will tell you which is minimum bound or minimum tells you given a constant matrix what is the minimum number of nodes required to prime implicant covering?

So first you execute the lower bound computation algorithm of the constraint matrix and generate the initial lower bound which is lower bound and current cost is set to 0. So first you have given a matrix and then you use lower bound algorithm and can find out the minimum number of nodes required to implement it. So you give the lower bound 1 at least 2 at least or 3 at least.


So again lower bound algorithm will not give you the higher estimate so the so for example, I told you something that 3 is the required prime implicant to cover delta may be 4 or 2, 3, 1 generally we will see that we will not get this upper side, we will always go to the lower side for example for example, three implicants are enough to cover it. So we will not report 4, 5, 6 in the delta side we never report $c \geq 1$ and so on. On the lower bound because they are in lower bound saying that minimum number of things are required. So it may form whatever bound you got slightly increase or to we will not to the other way, we will not go to the higher side like 4, 5, 6 and then real algorithm or the exact algorithm will bring it down because our computer is heuristic is generating low fall point that is minimum this much is required. So even they may require more than that initially you generate the lower bound and totally given stand and do all the processing and then you set current cost is 0.

(Refer Slide Time: 21:19)

Branch-and-Bound Algorithm for selecting cover

Before we turn our attention to the computation of the lower bound for the unate covering problem, we now look at the steps for the branch-and-bound algorithm for the unate covering problem.

1. Execute the lower bound computation algorithm on the constraint matrix. The "Initial Lower Bound" is the Lower Bound. Current Cost is set to 0.
2. Select or de-select a column P_i (i.e., a path in the solution space corresponding $P_i=1$ or $P_i=0$). If a column P_i is selected then add P_i to the cover and add cost 1 to the Current Cost. Apply reduction techniques (essential columns, row and column dominance) to the constraint matrix. For all essential columns, add them to the cover and increase Current Cost by the number of essential columns. If the problem is now reduced to a terminal case (because of selection of a column), then check whether the solution thus found is better than (or at least at par) the "Initial Lower Bound" (i.e., Initial Lower Bound \geq Current cost). If so, the cover is returned and algorithm is terminated.



Why the current cost is zero? The current cost is zero, because we are we are not selected the any primary details. Now you select or deselect column p_i first you go for this input i in the solution space you take P_i will be 1 or P_i will be 0 if column is selected add P_i to the power and add 1 to the current process now the weight of the current process is 1.

Now apply reduction techniques for essential rows and all these to the constraint matrix and now you take a column and add one to it arbitrary starting with any column taking or rejecting it. Now rejecting it obviously, some is an example if you take it also it will be processing. So if you take it you have to add it away if you do not take it, you do not add away and again regenerate matrix and generate row dominance and column dominance and reduce the matrix.

If all the essential columns for all the essential columns add them to the cover and increase the current cost but, during the preprocessing some of the columns become essential. So again add them to the current cost and again add them.

If the problem is now reduced to the terminal case everything is over what you have done is you have done with one column and you are not doing or take a column to select a way as 1. Now you do preprocessing means, essential column row dominance all you do and you have to increase the weight a as of as by adding 1 to the each of the column together.


Now if you find out that the terminal case and the cover is complete then you have to find out the what is the total cost? So if the total cost or the current cost is less than or equal to initial bound, then you have found out the approximate or linear bound of the number of what you say lower up of column matrix. So that I told you that initial lower bound is always give you the in the lower side, it will never give you in the upper side even if there is I mean tell me some abstract data.

(Refer Slide Time: 23:49)

Branch-and-Bound Algorithm for selecting cover

3. On the reduced matrix, again compute the lower bound. Add the lower bound value to the Current Cost. If there is still a chance of getting an optimal solution, identify a column to be selected/de-selected from the reduced matrix and go to Step-2; execute step-2 by selecting/de-selecting the column being marked.

4. If the value of lower bound + Current Cost is higher than Initial Lower Bound, then there is no point in exploring that path. Undo selection or de-selection of the column done last (in Step-2), consequent selection/de-selection of columns (due to matrix reduction) and addition in Current cost. Go to step-2 and traverse another path by selecting or de-selecting an alternative column.




For example in real case the number of rows columns will be 4. So you will get the answer c or something you will never get the lower side in the power, if this is happens by the current cost all the columns you have taken is less than or initial equal bound then you will get a good case then lower bound is returned and algorithm is terminated.

(Refer Slide Time: 23:59)

Branch-and-Bound Algorithm for selecting cover

Before we turn our attention to the computation of the lower bound for the unate covering problem, we now look at the steps for the branch-and-bound algorithm for the unate covering problem.

1. Execute the lower bound computation algorithm on the constraint matrix. The "Initial Lower Bound" is the Lower Bound. Current Cost is set to 0.
2. Select or de-select a column P_j (i.e., a path in the solution space corresponding $P_j=1$ or $P_j=0$). If a column P_j is selected then add P_j to the cover and add cost 1 to the Current Cost. Apply reduction techniques (essential columns, row and column dominance) to the constraint matrix. For all essential columns, add them to the cover and increase Current Cost by the number of essential columns. If the problem is now reduced to a terminal case (because of selection of a column), then check whether the solution thus found is better than (or at least at par) the "Initial Lower Bound" (i.e., Initial Lower Bound \leq Current cost). If so, the cover is returned and algorithm is terminated.




So because the heuristic will never report it in the plus side, it will always report in the negative side for example, three prime implicant is equal to the covering side your current cost is 3 and obviously it is reduced because you never get less than that but, in the practical case you may require 4, 5 or 6. So, if your algorithm is saying that the prime implicant you are covered till now is actually equal to a less than the initial bound then that solution is done you have reached a very good solution.

(Refer Slide Time: 24:13)

Branch-and-Bound Algorithm for selecting cover

Before we turn our attention to the computation of the lower bound for the unate covering problem, we now look at the steps for the branch-and-bound algorithm for the unate covering problem.

1. Execute the lower bound computation algorithm on the constraint matrix. The "Initial Lower Bound" is the Lower Bound. Current Cost is set to 0.
2. Select or de-select a column P_j (i.e., a path in the solution space corresponding $P_j=1$ or $P_j=0$). If a column P_j is selected then add P_j to the cover and add cost 1 to the Current Cost. Apply reduction techniques (essential columns, row and column dominance) to the constraint matrix. For all essential columns, add them to the cover and increase Current Cost by the number of essential columns. If the problem is now reduced to a terminal case (because of selection of a column), then check whether the solution thus found is better than (or at least at par) the "Initial Lower Bound" (i.e., Initial Lower Bound \leq Current cost). If so, the cover is returned and algorithm is terminated.





(Refer Slide Time: 24:36)

Branch-and-Bound Algorithm for selecting cover

3. On the reduced matrix, again compute the lower bound. Add the lower bound value to the Current Cost. If there is still a chance of getting an optimal solution, identify a column to be selected/de-selected from the reduced matrix and go to Step-2; execute step-2 by selecting/de-selecting the column being marked.

4. If the value of lower bound + Current Cost is higher than Initial Lower Bound, then there is no point in exploring that path. Undo selection or de-selection of the column done last (in Step-2), consequent selection/de-selection of columns (due to matrix reduction) and addition in Current cost. Go to step-2 and traverse another path by selecting or de-selecting an alternative column.




What it happens? That if the reduced matrix that is the current cost is becoming larger than your lower bound then try exploring and in the reduced matrix again add the lower bound; that means, what that you have reserved that ordinary case, if you can check it if this is greater than this one, then try exploring other part if you try terminal cases are not being used that then what you do? Processing will not be there in the reduced matrix, again the count will be in lower bound and the lower bound to the current cost and the how it has to be done again a new nominal case and done. So again you add a 1 to lower bound and add it to the corresponding. So; that means, that you are in an intermediate row. So the cost is 2 k, so this one path over here, one path over here is not a terminal. Now check that this route is included on reduced matrix and the reduced matrix over here. So there are I am sorry I am very very sorry so it can take or cannot take it.

(Refer Slide Time: 25:08)

Branch-and-Bound Algorithm for selecting cover

3. On the reduced matrix, again compute the lower bound. Add the lower bound value to the Current Cost. If there is still a chance of getting an optimal solution, identify a column to be selected/de-selected from the reduced matrix and go to Step-2; execute step-2 by selecting/de-selecting the column being marked.

4. If the value of lower bound + Current Cost is higher than Initial Lower Bound, then there is no point in exploring that path. Undo selection or de-selection of the column done last (in Step-2), consequent selection/de-selection of columns (due to matrix reduction) and addition in Current cost. Go to step-2 and traverse another path by selecting or de-selecting an alternative column.



Now actually here also will be there a reduced matrix after taking this node sorry I mean that we have list here saying that we have taken node if it is better in this way, now you have reached here, now you have taking the code like node p 1 and then taken one over here and print 1.

There may be other solutions that they p q and how it should the explore but, at this point you have a reduced matrix by taking it 1 you have you have row dominance, column dominance essential everything and finally, you got a reduced matrix and not a terminal case got it in terminal cases you can find it by taking cone c 4 columns. What is the cost of the columns bounds? Then you are very good, you are reached the optimal solution and here stop. It is not the terminal case, it is reduced by and see if the lower bound is say that is in this reduced matrix it will be there one of the implicants covering and you add it to the current cost, the current cost is 1 over here, 1 plus 3 is equal to 4. Now you have to check that the 4 is less than or equal to the initial bound.

So if you find that the greater than the initial bound you may know that even if you explore the place minimum explores from this sub tree, then it is minimum tree, you have to implement it and one you are already you are adding 1 is constant which is higher than the bound. So then you have to say than no I do not have to take p 1 I have to start taking the other path where I will not take p 1 will not take a p 1 you will get a. So, you will so, you will another node over here which corresponds to p 2 but, you have not

taken p 1 you this is what you have covered again? You found out in this reduced matrix, you are not taking p 1 we will get another reduced matrix and again you find out what is the minimum lower bound in the reduced matrix? And then tally to the current cost in this case; current cost will be 0 because p 1 you have not taken.

(Refer Slide Time: 27:05)

Branch-and-Bound Algorithm for selecting cover

3. On the reduced matrix, again compute the lower bound. Add the lower bound value to the Current Cost. If there is still a chance of getting an optimal solution, identify a column to be selected/de-selected from the reduced matrix and go to Step-2; execute step-2 by selecting/de-selecting the column being marked.
4. If the value of lower bound + Current Cost is higher than Initial Lower Bound, then there is no point in exploring that path. Undo selection or de-selection of the column done last (in Step-2), consequent selection/de-selection of columns (due to matrix reduction) and addition in Current cost. Go to step-2 and traverse another path by selecting or de-selecting an alternative column.

MPTEL

1-327

Then if you find out it is lower than the lower initial bound and there can be a path if you explore this solution and start exploring the path. So this is actually what we are been saying. So there is the 3 the value is lower than the current cost still there is a chance of getting an optimal solution identify a column to be selected from the reduced matrix and to step 2 and read it.

If the value of lower bound current cost is higher than initial lower bound, then there is no point in exploring the path undo the selection of the column last done that is what we are saying, undo the one in our example, undo the column in our last step consequent selection that is you undo everything you do not take p 1 you taking by p 1 by not taking p 1 you have done some actions you undo everything and actually you go to step 2 and traverse another the path by selecting or deselecting another path.

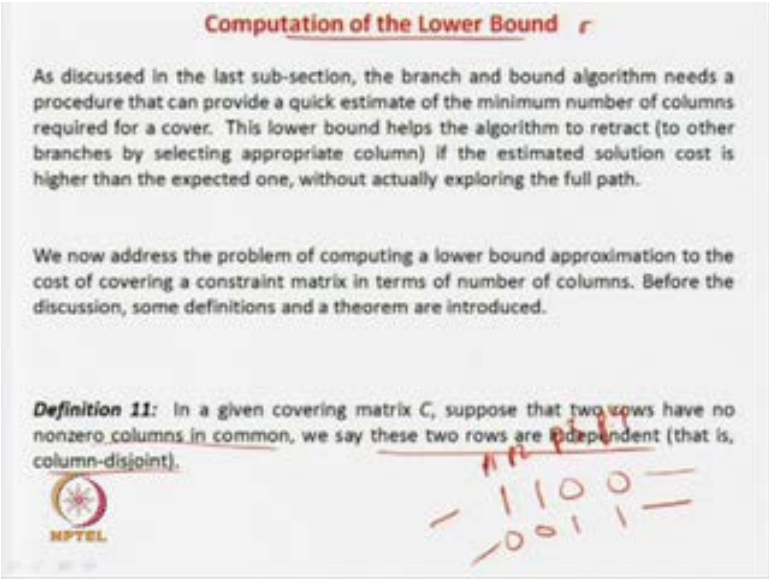
(Refer Slide Time: 27:44)

Computation of the Lower Bound r

As discussed in the last sub-section, the branch and bound algorithm needs a procedure that can provide a quick estimate of the minimum number of columns required for a cover. This lower bound helps the algorithm to retract (to other branches by selecting appropriate column) if the estimated solution cost is higher than the expected one, without actually exploring the full path.

We now address the problem of computing a lower bound approximation to the cost of covering a constraint matrix in terms of number of columns. Before the discussion, some definitions and a theorem are introduced.

Definition 11: In a given covering matrix C , suppose that two rows have no nonzero columns in common, we say these two rows are **θ -dependent** (that is, column-disjoint).



The slide includes a logo for NPTEL in the bottom left corner. In the bottom right, there is a handwritten matrix in red ink:

$$\begin{pmatrix} - & 1 & 1 & 0 & 0 \\ - & 0 & 0 & 1 & 1 \end{pmatrix}$$

So essentially what we have done what we have done actually we have selected a solution, we have deselected the solution first we have found the initial column and next we select the row or column based on that one if you select a column you add a weight to 1 then you reduce the matrix by dominance equivalence and essential matrix you take essential columns and it to the weight. Now in matrix you again add the approximation lower bound to columns then add it to the current cost.

If the current is cost actually lower than the initial exploration then you can say that it is profitable that you can do it, if you not if the value is higher than the estimated path you discarded the path and what you mean by discard the path? You go back to step 2 and undo everything, undo means? If you have taken column one you do not take column one, if you have not taken column one, you take column one and some actions you have done you have to undo all everything and again you have to explore the path and you keep on doing it. You will find out the path where your solution is equal to or less than the current bound.

If you can do it now where is the approximate path here because some of the points you are using a lower bound computation. So, lower bound is you find out the heuristic algorithm and always may not give you the very correct solution. So if you are not getting the correct exact bound some of the paths you may leave saying that the lower bound is 2 and if fact you and very much attracted by saying that saying that the lower

bound 2 to find start exploring the path, when you are going to the path you may actually say that the cost will be 7 or 8 it may happen something like that. So because the lower bound algorithms are very tight, it will never give you the additional value delta will give you the value.

So you can identify that this path exploring the sub tree will give you the solution with the cost of three prime implicant explore that path in the end, you find out that the cost is 5 and 4, 5 or 6 something like that is not a very good idea for exploring the path and similarly, some of the path reported as 2 but, in fact if you really explore the path the value will be really 2 only 2 or 3 maximum 1 delta will be there. So you could have explored that path and got the better solution because of this heuristic you may land up in a path which is showing that you get the value of 3 but, becoming plus 4 or plus 5. So you have gone to a different bad direction. So you are getting a solution which is higher than the value. So all this may things may happen.

So now we will see how can you compute the lower bound. So the algorithm is very simple start exploring every time you lower computation tool and found out the value adding to the current path, if you find out the path is good exploring good exploring you undo the path and try other path.

Otherwise simple algorithm you follow that there is actually branch-and-bound algorithm with heuristic. Now we will see heuristic, how to calculate the lower bound? Because in this branch-and-bound algorithm which is very straight and simple the heart of the algorithm is that the how accurately you compute the lower bound lower bound is accurate algorithm, you will find out the solution and you better lower bound is not good your solution will not be good.

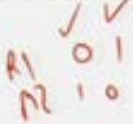

(Refer Slide Time: 31:21)

Computation of the Lower Bound r

As discussed in the last sub-section, the branch and bound algorithm needs a procedure that can provide a quick estimate of the minimum number of columns required for a cover. This lower bound helps the algorithm to retract (to other branches by selecting appropriate column) if the estimated solution cost is higher than the expected one, without actually exploring the full path.

We now address the problem of computing a lower bound approximation to the cost of covering a constraint matrix in terms of number of columns. Before the discussion, some definitions and a theorem are introduced.

Definition 11: In a given covering matrix C , suppose that two rows have no nonzero columns in common, we say these two rows are independent (that is, column-disjoint).



(Refer Slide Time: 31:37)


Computation of the Lower Bound

It is obvious that we need two different columns to cover these two rows.

Generalizing this argument, if a matrix has n rows that are disjoint (pair wise), we need at least n columns to cover the whole matrix.

In this case, the rows are said to form an independent set of rows.

In the constraint matrix given below, rows corresponding to minterm1, minterm3 and minterm5 are independent as they do not have a common column that has 1. Similarly, rows corresponding to minterm2, minterm4 and minterm6 are independent. So, we need at least three columns to cover the matrix. The lower bound in this case is 3.



So before we do that some definitions are there. So given the covering matrix constant matrix or whatever suppose that two rows have non-zero columns in common we say that these rows are independent, that is column is disjoint for example, if in one row you have 1 1 0 0 and in second row you have 0 0 1 1; that means, these rows have no places in common; that means, this is p 1, this is p 2, this is p 3 and this is p 4. So they are actually to cover this you have p 1, p 2 to cover the second row you require p 3 or p 4. So

if this happen what is the these two rows are? These two rows are column independent; that is row disjoint column disjoint that is 0 1 1 0.

This is row one, this is row two and so again see to cover, row one you require this column to cover row two, you require this column; that means, they are column disjoint; that means, that they have no common in both row. So if the more number of columns disjoint rows more number of prime implicant has to be taken. So obviously, you require two different columns to cover these two rows in general the argument has n row that are disjoint pair wise we require n columns to cover the matrix. So that is obviously in this case the rows are said to be independent set of rows, each of the rows is in the that is prime implicant row or the column are independent of each other.

(Refer Slide Time: 32:15)

Computation of the Lower Bound

	PI1	PI2	PI3	PI4	PI5	PI6
minterm1	1	0	0	0	0	1
minterm2	1	1	0	0	0	0
minterm3	0	1	1	0	0	0
minterm4	0	0	1	1	0	0
minterm5	0	0	0	1	1	0
minterm6	0	0	0	0	1	1

For cyclic matrices, lower bound is 2 or more, as shown in the following theorem.

Theorem 3: The lower bound for a cyclic matrix is at least 2.
 There is only one case when the lower bound is 1; constraint matrix contains a full column of ones. In this case, the matrix cannot be cyclic because the column with all ones dominates all the others. The matrix can therefore be reduced to one column, which is obviously essential. Thus, such a matrix is not cyclic.

NPTEL

So in the constant matrix given below correspondingly to minterm 1, this is your matrix this is there, so you can see that in the constraint matrix given below row corresponding to minterm 1, minterm 2 and minterm 5 are independent and so are 2, 4 and 6. So if you see this if you take row number on in this sorry if you take row number 1 just consider this row you can see that row number, you can see that you can see that it does not have anything in common once, in common row 3 in 1 row 3 have 1 in 2 and 3 and row 1 has 1 in row 1 and 6 row number 6 has a sorry h 1 3 and 5 sorry 1 3 and a 5. If you take 5 if you just find out a 1 in place of 4 and 5 h excuse me. So that is what, so if you have to cover the 1 3 and 5 3 columns are required. So for example, if you take PI 1 it will cover

minterm 1 and if you take PI 3 it will cover minterm 3. So if you take minterm 4 5 you have take 4.

(Refer Slide Time: 33:07)

Computation of the Lower Bound

	PI1	PI2	PI3	PI4	PI5	PI6
minterm1	1	0	0	0	0	1
minterm2	1	1	0	0	0	0
minterm3	0	1	1	0	0	0
minterm4	0	0	1	1	0	0
minterm5	0	0	0	1	1	0
minterm6	0	0	0	0	1	1

3

For cyclic matrices, lower bound is 2 or more, as shown in the following theorem.

Theorem 3: The lower bound for a cyclic matrix is at least 2. There is only one case when the lower bound is 1; constraint matrix contains a full column of ones. In this case, the matrix cannot be cyclic because the column with all ones dominates all the others. The matrix can therefore be reduced to one column, which is obviously essential. Thus, such a matrix is not cyclic.

NPTEL

So in other words 1 3 and 5 are all column disjoint rows are independent rows similarly, if you take minterm 2, minterm 4 and minterm 6 you can find out that there is no single column common for 3 of the rows. If you have to cover minterm 2 you should have taken PI 2 or 1 minterm 4 you have take 3 or 4 for minterm 6. You have 5 or 6 so 3 so 3 rows are required sorry 3 columns are required to cover column 3 5 and 6 and 3 will require to cover 1 3 and 5 of course, this independent sets 1 3 and 5. So 2 4 and 6 between 2 these two sets there can be obviously overlap but, if you consider 1 3 and 5 3 are required for 2 4 and 6 again three columns are required but, between these two subsets common sets will be there by looking at this matrix, easily you can say that lower bound is 3. So because there is 3 column independent rows so we can say lower inbound is 2.

So that is what is the saying, so we at least need 3 columns to cover the matrix, so the lower bound is 3 in this case.

(Refer Slide Time: 34:43)

Selecting a Subset of Primes

Let us consider an arbitrary constraint matrix as shown below.

	P1	P2	P3	P4	
minterm1	1	1	0	0	1
minterm2	0	1	1	0	1
minterm3	0	0	1	1	1
minterm4	1	0	0	1	1

$f=7$

1. A function has a cyclic core if we cannot identify columns of the constraint that must be part of the solution or that can be eliminated.
2. In the arbitrary constraint matrix, each row is covered by exactly two columns and each column covers exactly two rows. There is no essential primes. There is no apparent reason to prefer one column over another. For this matrix we must proceed by choosing one column arbitrarily and finding the best solution subject to the assumption that the column is selected. We must then assume that the column is not in the solution and find another solution. This process is repeated till the whole solution space is explored.

NPTEL

For cyclic matrix lower bound is 2. So how about lower bound? Because nobody can give you a lower bound less than 3, it can be more nobody is saying that it has to be exactly 3, nobody can give you a solution less than 3 in a cyclic matrix. What is a cyclic matrix? You have seen a cyclic matrix something like 1 sorry. In last class we have seen some example, so cyclic matrix is something like you just remember what are cyclic matrix is so there will be always two 1s, you cannot reduce the matrix as you can see this one, this one, this one and this one these matrix cannot be reduced. So this is a cyclic matrix. So nobody can solve it in the single column. So there is not reduction possible there is no column with 1 so at least the lower bound is 1 plus 1 that is equal to 2.

(Refer Slide Time: 35:03)

Selecting a Subset of Primes

Let us consider an arbitrary constraint matrix as shown below.

	PI1	PI2	PI3	PI4
minterm1	1	1	0	0
minterm2	0	1	1	0
minterm3	0	0	1	1
minterm4	1	0	0	1

H=2

1. A function has a cyclic core if we cannot identify columns of the constraint that must be part of the solution or that can be eliminated.
2. In the arbitrary constraint matrix, each row is covered by exactly two columns and each column covers exactly two rows. There is no essential primes. There is no apparent reason to prefer one column over another. For this matrix we must proceed by choosing one column arbitrarily and finding the best solution subject to the assumption that the column is selected. We must then assume that the column is not in the solution and find another solution. This process is repeated till the whole solution space is explored.

NPTEL

In such away there is no reduction possible there is no column with all 1s. So it will work always 2 but, it does not mean that all will be it says that nobody can find a solution less than 2 column why? Means there is no row which is a all one 1s; that means, the plus 1 plus 1 plus how much i 1 plus 1 which is minimum 2 rows must be required and the further the matrix cannot be simplified.

So for all cyclic matrix what I was saying for all the cyclic matrixes. So the lower bound is 2 so that is what is the thermo 3? The lower bound of the cyclic matrix is 2.

(Refer Slide Time: 35:32)

Computation of the Lower Bound

	PI1	PI2	PI3	PI4	PI5	PI6
minterm1	1	0	0	0	0	1
minterm2	1	1	0	0	0	0
minterm3	0	1	1	0	0	0
minterm4	0	0	1	1	0	0
minterm5	0	0	0	1	1	0
minterm6	0	0	0	0	1	1

3

For cyclic matrices, lower bound is 2 or more, as shown in the following theorem.

Theorem 3: The lower bound for a cyclic matrix is at least 2. There is only one case when the lower bound is 1: constraint matrix contains a full column of ones. In this case, the matrix cannot be cyclic because the column with all ones dominates all the others. The matrix can therefore be reduced to one column, which is obviously essential. Thus, such a matrix is not cyclic.

NPTEL

So why is it so? There is only one case the lower bound is 1. So by just beginning with the contra positivity lower bound is 1. So the lower bound is 1 means, that somebody must be able to find out the solution 1 1 this 1 1 column 1 column solution means, the matrix contains the full columns of 1s and if full columns of 1 then it is not a cyclic matrix and cyclic matrix is a case which I have already shown in the last slide.

(Refer Slide Time: 36:24)

Computation of the Lower Bound

There are several heuristics for computing the lower bound for the covering problem that have been proposed. Steps are as follows.

1. Add a field w to each row, whose value is equal to the number of 1's in the row
2. Choose the row with minimum w . Let it be r_i . If there are multiple rows with same value, choose the one from the top.
3. Delete all rows r_j such that r_i and r_j have at least one column where both of them have a 1. Also, delete r_i .
4. Repeat step 2 and 3 until no more rows remain.

The number of rows selected in Step-2 is the lower bound of the cover.

So in the it has 1s but, it has no row with all 1s. So the matrix cannot be reduced. So minimum 2 columns are required to solve the problem. So that is why the column is the lower bounds 2. So there is why the column matrix is not a cyclic matrix? So lower bound is actually 1, so lower bound is 2 means 2 plus nobody can give you less than 2 you have to add something. So these are some of the thermos. Now we are going to see why heuristic to complete the problem? So again it will not give you a accurate solution, it will give you some solution it says that add a field w to each row which is equal to the number of 1s in the row.

Choose the rows with minimum number of 1s, choose the minimum number of weight w . Let it be r_i . So multiple values rows choose the one form the top. So for each of these rows you are adding you count the number of rows in that row and we are arranging it in an ascending order. So we are arranging it in an ascending order. So then what you do? You start taking it from the top. So delete all rows r_j so at less at r_j have at least 1 column in common 1 row in common also delete r_j . So what we are doing r_i is false. So

we now first delete this row and then if you have common 1s, if in between in other rows r_j . So this one common over r_j say r_j . So in this example another r_j is there one in common there. So another r_j having one common. So you delete r_i along with it you delete r_j also repeat this two until not more no remains.

(Refer Slide Time: 37:49)

Computation of the Lower Bound

Let us consider the constraint matrix given below. The weights w are also shown in the matrix.

	P11	P12	P13	P14	P15	P16	w
minterm1	1	0	0	0	1	0	$w=2$
minterm2	1	1	0	1	0	0	$w=3$
minterm3	0	1	1	0	0	0	$w=2$
minterm4	0	0	0	1	1	1	$w=3$
minterm5	0	0	1	1	0	0	$w=2$
minterm6	0	1	0	0	0	1	$w=2$

In this case, we start with row1 (minterm1) which has the minimum weight and occurs first in chronological order among all other rows having equal weight. Now, rows minterm2 and minterm4 are also eliminated because of common columns column1 (P11) and column5 (P15), respectively. In the next iteration, row3 is selected. Now, rows minterm6 and minterm5 are also eliminated because of common columns column2 (P12) and column3 (P13), respectively. As there are no more rows, Lower Bound = 2 and comprises {1, 3}. It may be noted that in this case the bound is not sharp, as we require at least three columns in the cover.

How many times you have selected r_2 sorry r_i ? That is the actually your lower bound. So we will see with the example that is written and then we will again come back this one tell the motivation of the structure because in this you see w is equal to 2, since there are two 1. So here is also 3, 1, 2, 3, 1 less than 1, 2 and so forth. So in this case this are the ways, now this is the least weight 2.

So you start up with some other 1, 2, 3 1, 2, 3, 4 four rows are having the weight of 2 but, again I have told you to start from top these are the keys. So this is the what you have said. Now you eliminate this, now tell my lower bound is 1. So what you have to do? The row 1 and a 1 and 2 same one is a common. So again you read is 2 they again it is common with this one. So again you reduce this one right this is no commonality I do not think there is no more commonality here. So these two are rows are reduced, now row number 2 and row number 4 gets deleted. So next what we do? Sorry.

(Refer Slide Time: 39:55)

Computation of the Lower Bound

Let us consider the constraint matrix given below. The weights w are also shown in the matrix.

	P11	P12	P13	P14	P15	P16	
minterm1	1	0	0	0	0	0	$w=2$
minterm2	1	1	0	1	0	0	$w=3$
minterm3	0	1	1	0	0	0	$w=2$
minterm4	0	0	0	1	0	1	$w=3$
minterm5	0	0	1	1	0	0	$w=2$
minterm6	0	1	0	0	0	1	$w=2$

In this case, we start with row1 (minterm1) which has the minimum weight and occurs first in chronological order among all other rows having equal weight. Now, rows minterm2 and minterm4 are also eliminated because of common columns column1 (P11) and column5 (P15), respectively. In the next iteration, row3 is selected. Now, rows minterm6 and minterm5 are also eliminated because of common columns column2 (P12) and column3 (P13), respectively. As there are no more rows, Lower Bound = 2 and comprises {1, 3}. It may be noted that in this case the bound is not sharp, as we require at least three columns in the cover.

Now it is done. It is left with this matrix. Now you can think about this, you reduced matrix you are having. So again you take this the third row third row you can say that this row you delete and you add a 1 over here. So if you are deleting a row 3 what are the commonalties you are having? So it has two rows it is a common, so again row number there you have deleted you have a common of you have taken this row you have taken this row a common which is 6 6 also you can delete because this one is having a common with this one, this is gone, this again you take this one again it have common with this one. So this one is gone there is no more this one. So the lower bound is two and it comprises 1 and 3.

So let us quickly see? What we have done so we started with this one we started with one this one is having with this one this one can be eliminated this one is having a common with this one. So four gets eliminated and any more commons no and actually you add this one this row is added. So you take row this is common with this one. So this guy is eliminated. So this guy is having a column with this one this guy is eliminated. So 1 and 3 are remaining.

So this we have included so the cost is 2 1 plus 1 is 2 these are the way of saying. If you take this matrix nobody can solve it or give a solution less than 2 prime implicant. So this is the idea.

(Refer Slide Time: 40:40)

Computation of the Lower Bound

The key feature of this algorithm is it just chooses the "shortest" row, that is, the row with the fewest nonzero columns, and breaking ties in ascending order.

The basic motivation of choosing the "shortest" row first, comes from the fact—shorter the row is, higher is the probability of involving a column to cover it. If there is a row of $w=1$, then one column is mandatory to cover it. All rows with at least one common column of 1 are deleted because, all the rows can be covered with the common columns. This makes the solution faster because we eliminate many rows quickly.

However, this also leads to inaccuracy explained as follows. Let row1 be selected to be deleted. Let row2 has common column as column1 (with row1) and row3 has common column as column2 (with row1). So the lower bound estimate in this case is 1; deletion of row1 will also eliminate row2 and row3 because of common columns having 1. Also, let row2 and row3 be singletons. To cover row1 and row2 if we select column1, then selection of column2 is mandatory for row3. Similarly, to cover row1 and row3 if we select column2, then selection of column1 is mandatory for row2. Therefore, we require two columns in this case, which is not reflected in the estimate.

(Refer Slide Time: 40:50)

Computation of the Lower Bound

Let us consider the constraint matrix given below. The weights w are also shown in the matrix.

	P1	P2	P3	P4	P5	P6	
minterm1	1	0	0	0	1	0	$w=2$
minterm2	1	1	0	1	0	0	$w=3$
minterm3	0	1	1	0	0	0	$w=2$
minterm4	0	0	0	1	1	1	$w=3$
minterm5	0	0	1	1	0	0	$w=2$
minterm6	0	1	0	0	0	1	$w=2$

In this case, we start with row1 (minterm1) which has the minimum weight and occurs first in chronological order among all other rows having equal weight. Now, rows minterm2 and minterm4 are also eliminated because of common columns column1 (P1) and column5 (P5), respectively. In the next iteration, row3 is selected. Now, rows minterm6 and minterm5 are also eliminated because of common columns column2 (P2) and column3 (P3), respectively. As there are no more rows, Lower Bound = 2 and comprises {1, 3}. It may be noted that in this case the bound is not sharp, as we require at least three columns in the cover.

So now heuristic is this is the heuristic now what is the modulation? So actually modulation is written in this slide modulation is written in this slide then we will see what we have done we will see and again we will come back and explain. So what is the modulation? Modulation n says that why we are given weight? The minimum weights are two, minimum weight is one; that means, what is this row will be covered by minimum number of columns. So to include this it is more difficult to get this in included compared to this one because this is covered by prime implicant, it has the highest

probability of selecting this row or this row of covering prime implicant but, here is the number is 2 some 1.

So 1 1 in the row, so it becomes an essential prime implicant the less number of 1s, it is more difficult to cover this row less the probability more the difficulty we should the probability all the rows are provided. Idea is that, it is very difficult to cover that row compared to a row which is all 1s number of 1s. So that is that is why we do the ordering. We start with this one we take the number of we say that we select this row. So how do you select this row? We select this row it may either selected by this one or selected by this one and why you have started with minimum number of one? Because it is more difficult to compare with of the other case where more number of 1s.

Now we say if row number 1 is covered by pi 1 and automatically minterm gets covered right. If you cover it with PI 5 so in turn 4 case so; that means, if may it is covering that it is having a very high chance minterm 3 and minterm 4 gets covered. So it is actually cutting out the problem. So you quick reducing the matrix we had but, in essential what we happening you will cover minterm 1 either by PI 1 or the by PI 5 but, not both by this case we have a choice if you cover by PI 1 you can eliminate minterm 2 or row 2 gets eliminated. If i if i use PI 5 you can eliminate row 4, row number 4 can be eliminated by or but, you do this you are doing it with a tree else portion if I cover PI 1 in minterm you remove r 2 row 2 or if I give by 5 you remove 5 and again you see r 3 is coming into picture.

But, we do not want to do that, what we want to do? We want to do is that we want to it for a quick solution in this case, it will cover by either pi 1 or PI 5 but, not both but, it is a good chance that this will be covered or this will be covered. So if we take row number 3 so row 3 so this 1 you do by PI 2 you can cut minterm 6 if you do by PI 3 you can cut PI 5 but, not both in this case, we will cut both because the heuristic will going to give you the lower bound.

(Refer Slide Time: 43:08)

Computation of the Lower Bound

Let us consider the constraint matrix given below. The weights w are also shown in the matrix.

	P1	P2	P3	P4	P5	P6	
minterm1	1	0	0	0	1	0	$w=2$
minterm2	1	1	0	1	0	0	$w=3$
minterm3	0	1	1	0	0	0	$w=2$
minterm4	0	0	0	1	1	1	$w=3$
minterm5	0	0	1	1	0	0	$w=2$
minterm6	0	1	0	0	0	1	$w=2$

In this case, we start with row1 (minterm1) which has the minimum weight and occurs first in chronological order among all other rows having equal weight. Now, rows minterm2 and minterm4 are also eliminated because of common columns column1 (P1) and column5 (P5), respectively. In the next iteration, row3 is selected. Now, rows minterm6 and minterm5 are also eliminated because of common columns column2 (P2) and column3 (P3), respectively. As there are no more rows, Lower Bound = 2 and comprises {1, 3}. It may be noted that in this case the bound is not sharp, as we require at least three columns in the cover.

It is saying that if you cover PI 1 it has a good chance minterm 6 will go and if you do it by minterm 5 it has a good chance row 5 will go but, not both but, there is a solution. That we are going to cut down very fast. So we are just seeing that we have a co-ability there if you take minterm 1 you cover by PI 1 or you do not say anything, you just say if first row is covering there is a good chance of covering minterm 2 or say minterm 4 directly minterm 4 directly. So we cut out both of them and so, this speed is coming in this algorithm because we are seeing or and more number of rows in the iteration and there is no if else condition.

So you are quickly going to get a lower bound obviously, you are going to get the lower bound but, not any positive delta why positive delta is not there? Because we I am covering this row by PI 1 and automatically another row gets covered by 1 you either eliminate row 2 or but, not both that is very much clear over here. If you take PI 1 you can cover minterm 1 and minterm 2 and if you take PI 5 you can cover minterm 5 and minterm 4 but, both of them are not going to happen but, in this heuristic in one row I am covering minterm 1, minterm 2 and minterm 4. So you are taking more number into picture, you are going to get the lower bound downside negative side not to the upper side.

(Refer Slide Time: 44:45)

Computation of the Lower Bound

The key feature of this algorithm is it just chooses the "shortest" row, that is, the row with the fewest nonzero columns, and breaking ties in ascending order.

The basic motivation of choosing the "shortest" row first, comes from the fact—shorter the row is, higher is the probability of involving a column to cover it. If there is a row of $w=1$, then one column is mandatory to cover it. All rows with at least one common column of 1 are deleted because, all the rows can be covered with the common columns. This makes the solution faster because we eliminate many rows quickly.

However, this also leads to inaccuracy explained as follows. Let row1 be selected to be deleted. Let row2 has common column as column1 (with row1) and row3 has common column as column2 (with row1). So the lower bound estimate in this case is 1; deletion of row1 will also eliminate row2 and row3 because of common columns having 1. Also, let row2 and row3 be singletons. To cover row1 and row2 if we select column1, then selection of column2 is mandatory for row3. Similarly, to cover row1 and row3 if we select column2, then selection of column1 is mandatory for row2. Therefore, we require two columns in this case, which is not reflected in the estimate.

(Refer Slide Time: 45:06)

Computation of the Lower Bound

Let us consider the constraint matrix given below. The weights w are also shown in the matrix.

	P1	P2	P3	P4	P5	P6	
minterm1	1	0	0	0	1	0	$w=2$
minterm2	1	1	0	1	0	0	$w=3$
minterm3	0	1	1	0	0	0	$w=3$
minterm4	0	0	0	1	1	1	$w=3$
minterm5	0	0	1	1	0	0	$w=2$
minterm6	0	1	0	0	0	1	$w=2$

In this case, we start with row1 (minterm1) which has the minimum weight and occurs first in chronological order among all other rows having equal weight. Now, rows minterm2 and minterm4 are also eliminated because of common columns column1 (P1) and column5 (P5), respectively. In the next iteration, row3 is selected. Now, rows minterm6 and minterm5 are also eliminated because of common columns column2 (P2) and column3 (P3), respectively. As there are no more rows, Lower Bound = 2 and comprises {1, 3}. It may be noted that in this case the bound is not sharp, as we require at least three columns in the cover.

(Refer Slide Time: 45:28)

Computation of the Lower Bound

Let us consider the constraint matrix given below. The weights w are also shown in the matrix.

	P1	P2	P3	P4	P5	P6	
minterm1	1	0	0	0	1	0	$w=2$
minterm2	1	1	0	1	0	0	$w=3$
minterm3	0	1	1	0	0	0	$w=3$
minterm4	0	0	0	1	1	0	$w=3$
minterm5	0	0	1	1	0	0	$w=2$
minterm6	0	1	0	0	0	1	$w=2$

In this case, we start with row1 (minterm1) which has the minimum weight and occurs first in chronological order among all other rows having equal weight. Now, rows minterm2 and minterm4 are also eliminated because of common columns column1 (P1) and column5 (P5), respectively. In the next iteration, row3 is selected. Now, rows minterm6 and minterm5 are also eliminated because of common columns column2 (P2) and column3 (P3), respectively. As there are no more rows, Lower Bound = 2 and comprises {1, 3}. It may be noted that in this case the bound is not sharp, as we require at least three columns in the cover.

We want explicitly to do that, because we want a faster solution that we are quickly going to reduce the number of matrix sizes and it is actually going to give you a good solution I mean I am sorry the it is going to give the fast solution but, bound is going to be in the negative side. So real solution is higher than the bound I get that is how your heuristic is designed or branch-and-bound heuristic is designed. If you can reach the; that means, what whatever the algorithm will give you is this one, this is the real solution can be here or real solution can be here or heuristic solution real solution can be above this or at the level of heuristic but, the or never be down this side.

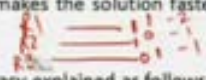
So that is a good sign if you somehow get your solution by branch-and-bound some solution you are doing if you actually reaching the reaching the value you have got you know that my solution is the most optimal one and cannot go beyond that one, because this heuristic is going to give you a solution at least the here may go above the lower bound but, nobody can get the solution less than that value. So if you are doing a branch-and-bound and if you are going to given by the heuristic. So you have reached the solution, where nobody can give the very very less value, this is how the heuristic is. So that is I told you this is how you going to by saying that taking minterm 1. So I covered minterm 1 also minterm 4 also how it is possible? It is possible because it says that nobody can do less than that nobody is saying that you need not go more than 1 what the heuristic is suggesting.

(Refer Slide Time: 46:20)

Computation of the Lower Bound

The key feature of this algorithm is it just chooses the "shortest" row, that is, the row with the fewest nonzero columns, and breaking ties in ascending order.

The basic motivation of choosing the "shortest" row first, comes from the fact—shorter the row is, higher is the probability of involving a column to cover it. If there is a row of $w=1$, then one column is mandatory to cover it. All rows with at least one common column of 1 are deleted because, all the rows can be covered with the common columns. This makes the solution faster because we eliminate many rows quickly.



However, this also leads to inaccuracy explained as follows. Let row1 be selected to be deleted. Let row2 has common column as column1 (with row1) and row3 has common column as column2 (with row1). So the lower bound estimate in this case is 1; deletion of row1 will also eliminate row2 and row3 because of common columns having 1. Also, let row2 and row3 be singletons. To cover row1 and row2 if we select column1, then selection of column2 is mandatory for row3. Similarly, to cover row1 and row3 if we select column2, then selection of column1 is mandatory for row2. Therefore, we require two columns in this case, which is not reflected in the estimate.

That is why some inaccuracy is there. So basic motivation is that taking the shortest row is that because shorter the row higher the involving the covering the probability other words more difficult it is to be selected. So in other words shortest rows minimum number of rows, minimum number of rows means? That it is going to become a essential prime implicant column corresponding to it we have to implement it. So give some probability to selecting of the row told you it is also in accuracy. So very simple example is that let row one be selected or deleted, so that can be the case later row two has a common column, let row two has a common column as column one with row one row three as common column one with row two just read this paragraph and do this we will finally, happen.

(Refer Slide Time: 48:30)

Computation of the Lower Bound

The key feature of this algorithm is it just chooses the "shortest" row, that is, the row with the fewest nonzero columns, and breaking ties in ascending order.

The basic motivation of choosing the "shortest" row first, comes from the fact—shorter the row is, higher is the probability of involving a column to cover it. If there is a row of $w=1$, then one column is mandatory to cover it. All rows with at least one common column of 1 are deleted because, all the rows can be covered with the common columns. This makes the solution faster because we eliminate many rows quickly.

However, this also leads to inaccuracy explained as follows. Let row1 be selected to be deleted. Let row2 has common column as column1 (with row1) and row3 has common column as column2 (with row1). So the lower bound estimate in this case is 1; deletion of row1 will also eliminate row2 and row3 because of common columns having 1. Also, let row2 and row3 be singletons. To cover row1 and row2 if we select column1, then selection of column2 is mandatory for row3. Similarly, to cover row1 and row3 if we select column2, then selection of column1 is mandatory for row2. Therefore, we require two columns in this case, which is not reflected in the estimate.

(Refer Slide Time: 48:36)

Computation of the Lower Bound

The key feature of this algorithm is it just chooses the "shortest" row, that is, the row with the fewest nonzero columns, and breaking ties in ascending order.

The basic motivation of choosing the "shortest" row first, comes from the fact—shorter the row is, higher is the probability of involving a column to cover it. If there is a row of $w=1$, then one column is mandatory to cover it. All rows with at least one common column of 1 are deleted because, all the rows can be covered with the common columns. This makes the solution faster because we eliminate many rows quickly.

However, this also leads to inaccuracy explained as follows. Let row1 be selected to be deleted. Let row2 has common column as column1 (with row1) and row3 has common column as column2 (with row1). So the lower bound estimate in this case is 1; deletion of row1 will also eliminate row2 and row3 because of common columns having 1. Also, let row2 and row3 be singletons. To cover row1 and row2 if we select column1, then selection of column2 is mandatory for row3. Similarly, to cover row1 and row3 if we select column2, then selection of column1 is mandatory for row2. Therefore, we require two columns in this case, which is not reflected in the estimate.

(Refer Slide Time: 48:54)

Computation of the Lower Bound

The key feature of this algorithm is it just chooses the "shortest" row, that is, the row with the fewest nonzero columns, and breaking ties in ascending order.

The basic motivation of choosing the "shortest" row first, comes from the fact—shorter the row is, higher is the probability of involving a column to cover it. If there is a row of $w=1$, then one column is mandatory to cover it. All rows with at least one common column of 1 are deleted because, all the rows can be covered with the common columns. This makes the solution faster because we eliminate many rows quickly.

However, this also leads to inaccuracy explained as follows. Let row1 be selected to be deleted. Let row2 has common column as column1 (with row1) and row3 has common column as column2 (with row1). So the lower bound estimate in this case is 1; deletion of row1 will also eliminate row2 and row3 because of common columns having 1. Also, let row2 and row3 be singletons. To cover row1 and row2 if we select column1, then selection of column2 is mandatory for row3. Similarly, to cover row1 and row3 if we select column2, then selection of column1 is mandatory for row2. Therefore, we require two columns in this case, which is not reflected in the estimate.

So it is saying that let row one be selected let row has common column one so, sorry this is row one, this is row one, this is row two and this row three, this is 1, 2 and 3 k. Now it is saying that row two has a common column as row one this is both of them have one column one and row three has and row three has a as column two with column one. So in this case it is a one, this is a one, this is a one and zero kind of thing and this is also a zero kind of thing. So lower bound estimate in this case is a one. So how is that? So here the weight is 2 it is a 1 and it is 1 k. Now if you take in this case this how it will look like and now if you find out this whole algorithm. So you can find out the in this case the lower order algorithm will be equal to 1.

So how is that one, let us see we have taken row 1 and selected it with 2 sorry you just represent it just with this way, note it in different way and nicely find out. So let out read form this row two have column one with row one. So with this row 1, row 2 we will take a. So we will just take this point giving an example in this case the point is going to be radiant. So you just remember this point and again we will come back with slide with this example. So how the solution is going to be a wrong estimate? So this one will come back to the point, so just because this heuristic we have taken some times you may get a wrong value. So how it is there we will take an example and come back again to this point. So this is the example we are talking about and the same example will show you that sometimes you are going to get a wrong estimate.

(Refer Slide Time: 49:33)

Example of Branch and Bound applied to Unate covering

Let us consider the constraint matrix given below. It may be noted that this matrix cannot be further simplified. If we apply the quick lower bound estimate, we get Lower Bound = 4 comprising {1, 3, 5, 7}.

	PI1	PI2	PI3	PI4	PI5	PI6	PI7	PI8	PI9	PI10	PI11
minterm1	1	1	0	0	0	0	0	0	0	0	0
minterm2	0	1	1	0	0	0	0	0	0	0	0
minterm3	0	0	1	1	0	0	0	0	0	0	0
minterm4	1	0	0	1	0	0	0	0	0	0	0
minterm5	0	0	0	0	1	1	0	0	0	1	0
minterm6	0	0	0	0	0	1	1	0	1	0	0
minterm7	0	0	0	0	0	0	1	1	0	0	0
minterm8	0	0	0	0	0	1	0	1	0	1	1
minterm9	0	0	0	0	1	0	0	0	1	1	1
minterm10	0	0	0	0	1	0	0	1	1	0	0
minterm11	0	0	0	0	1	0	1	0	0	0	1
minterm12	1	0	0	0	0	0	0	0	0	0	1

So let us just go back and go, so this is the matrix. So this is the working example this will be used by the theory. So if you look at this, this is very big matrix I will not be solve it sorry. This let you can easily do that if you find out the lower bound algorithm in this case. So you will find the lower bound is 4. So it will be equal to 1 3 5 and 7. So you can just put up the rows and find out.

So 1 3 5 7 is your answer. So; that means, lower bound is 4 in the first bound of algorithm lower bound is so nobody can give a solution less than 0 just you have to keep in mind. Now if you are doing branch-and-bound and you find out that sorry 4. Now you are doing branch-and-bound and solution is someone saying that then you are very happy you have got the solution.


(Refer Slide Time: 50:21)

Example of Branch and Bound applied to Unate covering

Now we start exploring the solution space. Let us start with selecting P11 (i.e., $P_{11}=1$). With P_{11} being taken, the following happens in the matrix

- rows minterm1, minterm4, and minterm12 are covered,
- columns P12 (by P13) and P14 (by P13) are dominated
- column P13 becomes essential.

The Binary search tree illustrating the search in the solution space by the branch and bound algorithm is shown in next figure. The left edge (right dotted edge) indicates the column being (not being) considered.




(Refer Slide Time: 50:56)

Example of Branch and Bound applied to Unate covering

Let us consider the constraint matrix given below. It may be noted that this matrix cannot be further simplified. If we apply the quick lower bound estimate, we get Lower Bound = 4 comprising {1, 3, 5, 7}.

	P11	P12	P13	P14	P15	P16	P17	P18	P19	P110	P111
minterm1	1	1	0	0	0	0	0	0	0	0	0
minterm2	0	1	1	0	0	0	0	0	0	0	0
minterm3	0	0	1	1	0	0	0	0	0	0	0
minterm4	1	0	0	1	0	0	0	0	0	0	0
minterm5	0	0	0	0	1	1	0	0	0	1	0
minterm6	0	0	0	0	0	1	1	0	1	0	0
minterm7	0	0	0	0	0	0	1	1	0	0	0
minterm8	0	0	0	0	0	1	0	1	0	1	1
minterm9	0	0	0	0	1	0	0	0	1	1	1
minterm10	0	0	0	0	1	0	0	1	1	0	0
minterm11	0	0	0	0	1	0	1	0	0	0	1
minterm12	1	0	0	0	0	0	0	0	0	0	1




(Refer Slide Time: 51:45)

Example of Branch and Bound applied to Unate covering

Now we start exploring the solution space. Let us start with selecting PI1 (i.e., PI1=1). With PI1 being taken, the following happens in the matrix

- rows minterm1, minterm4, and minterm112 are covered,
- columns PI2 (by PI3) and PI4 (by PI3) are dominated
- column PI3 becomes essential.

The Binary search tree illustrating the search in the solution space by the branch and bound algorithm is shown in next figure. The left edge (right dotted edge) indicates the column being (not being) considered.



So now what we will do now again we will start exploring. Now you have to start with so, start exploring the solution space later start with PI 1 in this case there cannot be any preprocessing done, there will be no essential column, no essential row will be there. So no single term nothing will be eliminated with row dominance or column dominance or nothing or nothing simplification can be of this matrix as of now. Now we start exploring solution you take PI 1 now what we do so? If I take PI 1 so atomically this row will be covered so again PI 1 minterm 3 will be covered.

Let us we nice we remove this because I have taken p 1 so this is also eliminated and again minterm 12 is as well eliminated because I have taken the this no I have taken. Now in this way you can see that what happens? h this is what I have taken automatically PI 1 minterm 4 and minterm 12 gets covered correct? Now you have to just see you have to now what are the replications for that rows 1, 12 are covered column PI 2 and 4 are dominated PI 4 and what you seeing PI 4 are dominated PI 2 so we see PI 2 has a 1 and PI 3 has two 1s in it.

(Refer Slide Time: 51:48)

Example of Branch and Bound applied to Unate covering

Let us consider the constraint matrix given below. It may be noted that this matrix cannot be further simplified. If we apply the quick lower bound estimate, we get Lower Bound = 4 comprising {1, 3, 5, 7}.

	PI1	PI2	PI3	PI4	PI5	PI6	PI7	PI8	PI9	PI10	PI11
minterm1	1	0	0	0	0	0	0	0	0	0	0
minterm2	0	1	1	0	0	0	0	0	0	0	0
minterm3	0	0	1	1	0	0	0	0	0	0	0
minterm4	1	0	0	0	0	0	0	0	0	0	0
minterm5	0	0	0	0	1	1	0	0	0	1	0
minterm6	0	0	0	0	0	1	1	0	1	0	0
minterm7	0	0	0	0	0	0	1	1	0	0	0
minterm8	0	0	0	0	0	1	0	1	0	1	1
minterm9	0	0	0	0	1	0	0	0	1	1	1
minterm10	0	0	0	0	1	0	0	1	1	0	0
minterm11	0	0	0	0	1	0	1	0	0	0	1
minterm12	0	0	0	0	0	0	0	0	0	0	1


(Refer Slide Time: 52:13)

Example of Branch and Bound applied to Unate covering

Now we start exploring the solution space. Let us start with selecting PI1 (i.e., $PI1=1$). With PI1 being taken, the following happens in the matrix

- rows minterm1, minterm4, and minterm12 are covered,
- columns PI2 (by PI3) and PI4 (by PI3) are dominated
- column PI3 becomes essential.

The Binary search tree illustrating the search in the solution space by the branch and bound algorithm is shown in next figure. The left edge (right dotted edge) indicates the column being (not being) considered.



Similarly for PI 3 so no other 1s in this column and no other 1s in column. So obviously this PI 3 is dominating PI 2 and PI 4 by that you can eliminate this and you can eliminate this these two rows get I mean this two columns get dominance by PI 3. Next what is the case? PI 3 become essential why is that? Now you see that the row in row 3 you see it there is only a 1 1 in that. So PI 3 becomes a essential because in this case PI 2 get dominated by PI 3, PI 4 also get dominated by PI 3.

(Refer Slide Time: 52:51)

Example of Branch and Bound applied to Unate covering

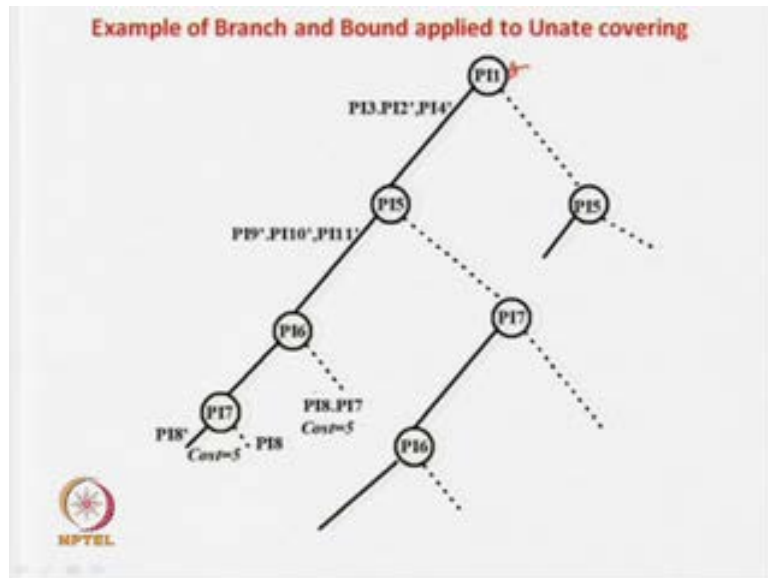
Let us consider the constraint matrix given below. It may be noted that this matrix cannot be further simplified. If we apply the quick lower bound estimate, we get Lower Bound = 4 comprising {1, 3, 5, 7}.

	PI1	PI2	PI3	PI4	PI5	PI6	PI7	PI8	PI9	PI10	PI11
minterm1	1	0	0	0	0	0	0	0	0	0	0
minterm2	0	1	1	0	0	0	0	0	0	0	0
minterm3	0	0	1	1	0	0	0	0	0	0	0
minterm4	1	0	0	0	0	0	0	0	0	0	0
minterm5	0	0	0	0	1	1	0	0	0	1	0
minterm6	0	0	0	0	0	1	1	0	1	0	0
minterm7	0	0	0	0	0	0	1	1	0	0	0
minterm8	0	0	0	0	0	1	0	1	0	1	1
minterm9	0	0	0	0	1	0	0	0	1	1	1
minterm10	0	0	0	0	1	0	0	1	1	0	0
minterm11	0	0	0	0	1	0	1	0	0	0	1
minterm12	0	0	0	0	0	0	0	0	0	0	1

Now you can eliminate PI 3 and PI 4 and you have only 2. What you say this numbers this PI 2 are there obviously, if you look at the minterm 3 there is only a one term they all are eliminated. So this only one is there whole row so obviously, PI 3 becomes essential PI 3 becomes essentially automatically minterm 2 gets covered by this 1 and minterm 3 gets covered by this 1. So our current solution is what we have taken PI 1 PI 3 gets covered

So your initial cost is 2, because PI 2 and PI 3 is there so obviously, minterm 2 is gone and minterm 3 is covered by this one and any more 1s over here. So no PI 3 is obviously taken into picture, this is the now we will find out the so you have got the simplified matrix as this 1. So that is 5, 6, 7, 8, 9, 10 and 11. So this one is you matrix and you have taken a PI 1 and PI 7. So this is the fast type of the branch-and-bound algorithm cost is 2 1 plus 1 2 and the lower bound is find out by using this algorithm this is what the structure.

(Refer Slide Time: 53:40)



(Refer Slide Time: 53:50)

Example of Branch and Bound applied to Unate covering

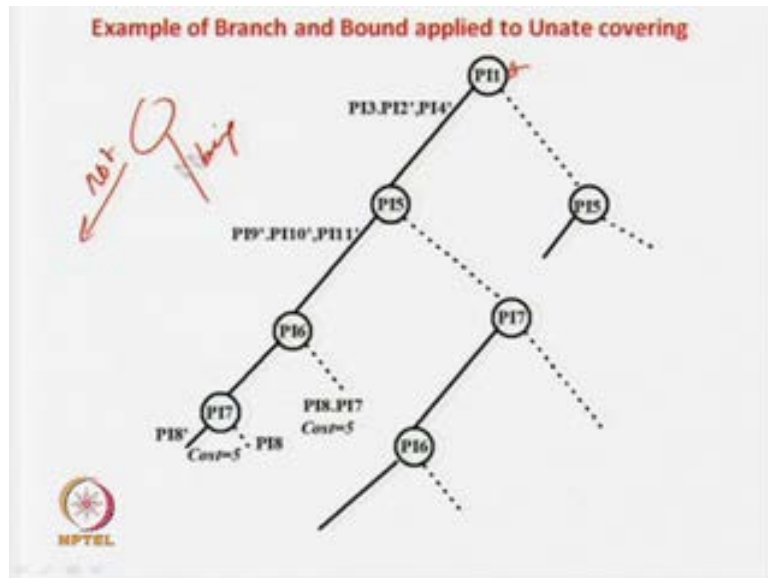
Now we start exploring the solution space. Let us start with selecting P11 (i.e., $P11=1$). With P11 being taken, the following happens in the matrix

- rows minterm1, minterm4, and minterm112 are covered,
- columns P12 (by P13) and P14 (by P13) are dominated
- column P13 becomes essential.

The Binary search tree illustrating the search in the solution space by the branch and bound algorithm is shown in next figure. The left edge (right dotted edge) indicates the column being (not being) considered.

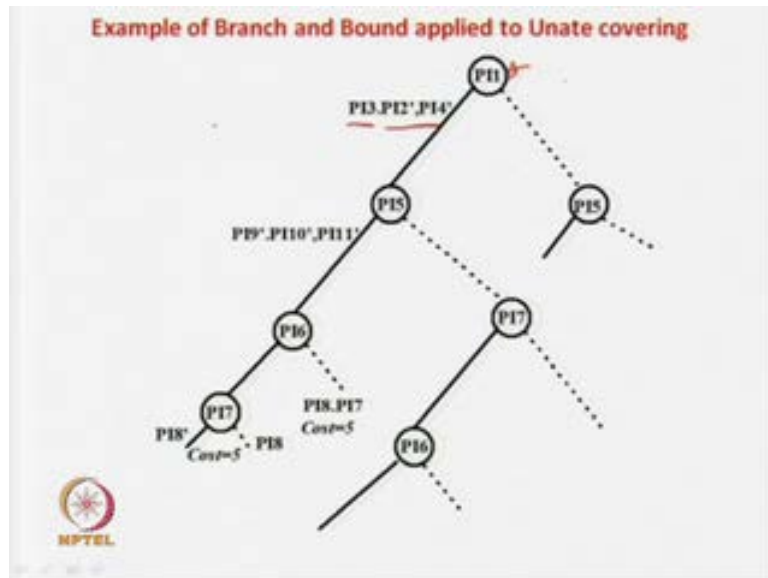
NPTEL

(Refer Slide Time: 53:57)



So now we have this space. Now what is the case? It says that sorry it is done in a reverse way, in this case left edge right dotted edge indicate the column being taken and not being taken. So this one is actually a this one corresponds to taken not be taken. So generally, we write this left edge not been taken this is into been taken and this is taken generally, right in this case it is represented by dotted lines and thick lines. So what is that PI 1 taken obviously and PI 3 comes default into picture because of essentiality and PI 3 and PI 4 gets deleted. So of the column dominance so, that is being represented by this part of the tree.

(Refer Slide Time: 54:16)



(Refer Slide Time: 54:27)

Example of Branch and Bound applied to Unate covering

The left edge from root note indicates P11 being taken. Also in the edge it is marked that "P13, P12, P14"—this indicates that because of taking P11, P13 has to be considered and P12 and P14 get eliminated.

After reduction (i.e., taking P11, P13 and eliminating P12, P13) we get the following matrix. As of now, the cost of the solution is 2 (P11, P13).

	P15	P16	P17	P18	P19	P110	P111
minterm5	1	1	0	0	0	1	0
minterm6	0	1	1	0	1	0	0
minterm7	0	0	1	1	0	0	0
minterm8	0	1	0	1	0	1	1
minterm9	1	0	0	0	1	1	1
minterm10	1	0	0	1	1	0	0
minterm11	1	0	1	0	0	0	1

(Refer Slide Time: 55:32)

Example of Branch and Bound applied to Unate covering


In this matrix Lower Bound = 2 comprising (5,7). So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

The left edge from root node indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11"—this indicates that because of taking P15, P19, P110 and P111 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P110, P11) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1



Now this is the case left edge indicate that PI 1 is taken and this one indicates PI 4 is because of taking PI 1, PI 3 have become dominance essential and PI 3 and PI 4 gets eliminated because of row and column dominance sorry because of the row dominance. So this is what is represented. Now this is your reduced matrix. Now what you do again you have to apply the minimum bound algorithm. So again if you apply the minimum bound algorithm find out that cost of the solution so sorry the intermediate cost is two because you have taken PI 1 and PI 3 because you have taken the reduced matrix again you have to apply the what you have do lower bound algorithm.

So if you do the lower bound computation you will find that 5 and 7 if you take 5 this are all gone and if take 7 7 in this case. So this one is gone what else this one is gone, this one gone. So you can find out the by taking the 5 and 7, this 5 and 7 this find out the whole matrix will be covered. So the so in this case the lower cost is nothing but, this is the lower bound of this algorithm which is two comprising of column 5 and 7 sorry rows 5 and 7 when lower bound is 2 you have add it 2. So does not cost of this two because of column number one and column number is two add two to the lower bound of this add 2 plus 2 is 4.

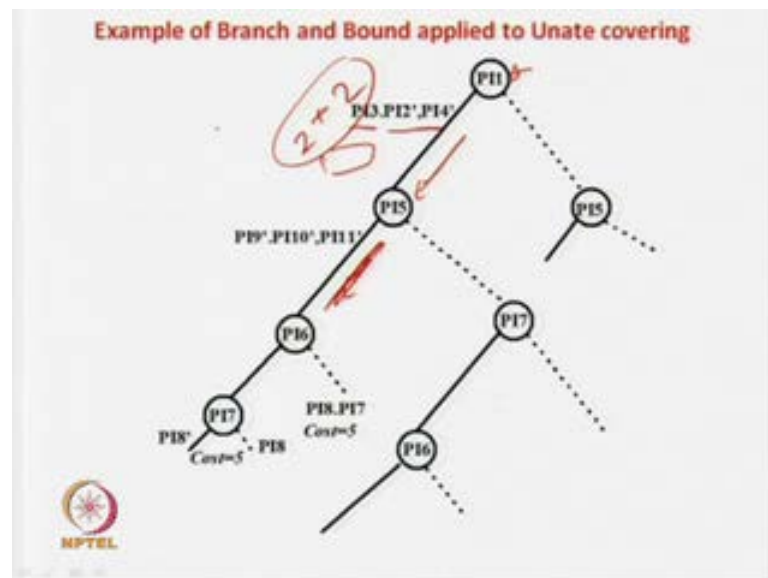
(Refer Slide Time: 55:57)

Example of Branch and Bound applied to Unate covering

Let us consider the constraint matrix given below. It may be noted that this matrix cannot be further simplified. If we apply the quick lower bound estimate, we get Lower Bound = 4 comprising {1, 3, 5, 7}.

	PI1	PI2	PI3	PI4	PI5	PI6	PI7	PI8	PI9	PI10	PI11
minterm1	1	0	0	0	0	0	0	0	0	0	0
minterm2	0	1	1	0	0	0	0	0	0	0	0
minterm3	0	0	1	1	0	0	0	0	0	0	0
minterm4	1	0	0	0	0	0	0	0	0	0	0
minterm5	0	0	0	0	1	1	0	0	0	1	0
minterm6	0	0	0	0	0	1	1	0	1	0	0
minterm7	0	0	0	0	0	0	1	1	0	0	0
minterm8	0	0	0	0	0	1	0	1	0	1	1
minterm9	0	0	0	0	1	0	0	0	1	1	1
minterm10	0	0	0	0	1	0	0	1	1	0	0
minterm11	0	0	0	0	1	0	1	0	0	0	1
minterm12	0	0	0	0	0	0	0	0	0	0	1

(Refer Slide Time: 56:00)



So now this 4 is the initial upper bound computation sorry initial lower bound computation initial lower bound computation is equal to 4. Now we can understand that is what to explore this path there is a chance that or you explore this path or you explore from this area these are lead to the matrix. So cost is equal to the 2 2 plus 2 that is PI 3 and PI 4 is 4 which is equal to the lower bound. So indicate that there is a good chance that if you start exploring this path you may get a solution which is equal to the 4 or 5 or something like that at least you get a solution which it is 4 or you are in a good shape because nobody give a solution in less than 4 for this matrix that is already been proved.

(Refer Slide Time: 56:48)

Example of Branch and Bound applied to Unate covering


In this matrix Lower Bound = 2 comprising {5,7}. So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

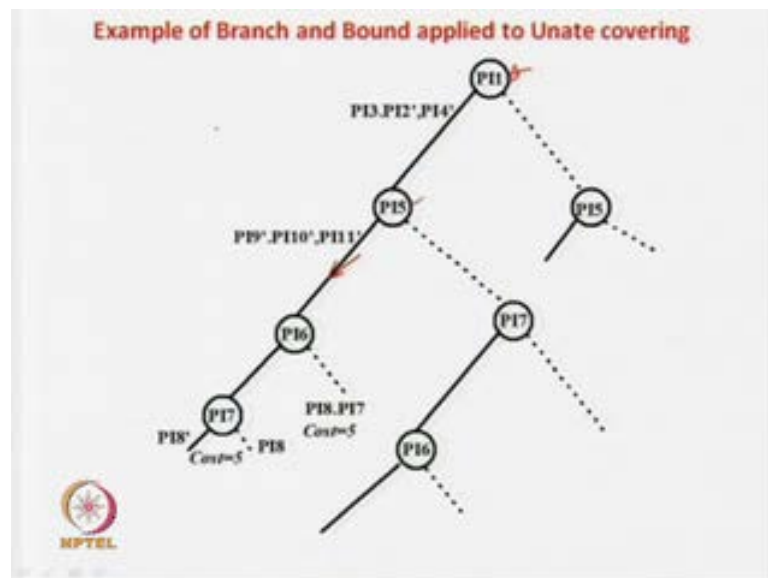
The left edge from root note indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11"—this indicates that because of taking P15, P19, P10 and P11 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P10, P11) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1



(Refer Slide Time: 56:52)



(Refer Slide Time: 57:04)


Example of Branch and Bound applied to Unate covering

The left edge from root node indicates P_{11} being taken. Also in the edge it is marked that " P_{13}, P_{12}, P_{14} "—this indicates that because of taking P_{11} , P_{13} has to be considered and P_{12} and P_{14} get eliminated.

After reduction (i.e., taking P_{11} , P_{13} and eliminating P_{12} , P_{13}) we get the following matrix. As of now, the cost of the solution is 2 (P_{11} , P_{13}).

	P_{15}	P_{16}	P_{17}	P_{18}	P_{19}	P_{110}	P_{111}
minterm5	1	1	0	0	0	1	0
minterm6	0	1	1	0	1	0	0
minterm7	0	0	1	1	0	0	0
minterm8	0	1	0	1	0	1	1
minterm9	1	0	0	0	1	1	1
minterm10	1	0	0	1	1	0	0
minterm11	1	0	1	0	0	0	1

∴



Now, again what you do? Now you eliminate now you start working on the other matrix he has given you the lower bound has said that very good you can explore on this matrix and you may get a solution equal to 4. Now what you do row? Now let us consider column five now P_5 has been taken so trying this one P_5 is been taken. So we are going this path. So let us assume that we are taking this path we are so why we are exploring in this path? We are exploring in this path because this matrix is lower bound 2 and good chance of finding a solution.

(Refer Slide Time: 56:48)

Example of Branch and Bound applied to Unate covering


In this matrix Lower Bound = 2 comprising $\{5, 7\}$. So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P_{15} . With P_{15} being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P_{19} (by P_{16}), P_{110} (P_{16}) and P_{111} (by P_{16}) are dominated

The left edge from root node indicates P_{15} being taken (tree). Also in the edge it is marked that " P_{19}, P_{10}, P_{11} "—this indicates that because of taking P_{15} , P_{19} , P_{10} and P_{11} get eliminated.

After reduction (i.e., taking P_{15} and eliminating P_{19} , P_{10} , P_{11}) we get the following matrix. As of now the cost of the solution is 3 (P_{11} , P_{13} , P_{15}).

	P_{16}	P_{17}	P_{18}
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1



(Refer Slide Time: 58:04)


Example of Branch and Bound applied to Unate covering

The left edge from root node indicates P_1 being taken. Also in the edge it is marked that " P_3, P_2, P_4 "—this indicates that because of taking P_1 , P_3 has to be considered and P_2 and P_4 get eliminated.

After reduction (i.e., taking P_1 , P_3 and eliminating P_2 , P_3) we get the following matrix. As of now, the cost of the solution is 2 (P_1 , P_3).

	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}
minterm5	1	1	0	0	0	1	0
minterm6	0	1	1	0	1	0	0
minterm7	0	0	1	1	0	0	0
minterm8	0	1	0	1	0	1	0
minterm9	1	0	0	0	1	1	1
minterm10	1	0	0	1	1	0	0
minterm11	1	0	1	0	0	0	1

$= 2$



Now let us take P_1 sorry P_5 . So P_5 is taken so minterm 5 is covered minterm 5 is covered, minterm 10 is covered and minterm 11 is covered. So that is a very good thing. Now if you see our reduced matrix will be reduced matrix will be what? So reduced matrix is now this portion right this is all reduced matrix. Now if you see our reduced matrix what is the case? If there is any what happens you see with five been taken the following happens row number 9, 10 and 11 are covered or say 5 is taken 9 10 and 11 are automatically covered columns 9 10 and 11 are all dominated by 6. So you can check 9.

So there is only a one over here nine then what they have said what is the case 9 see 9 10 and 11. If you take 9 sorry 9 this is 10 and this is 11 as you can see 9 will be dominated by column number 7 I think P_6 . So if you look at P_6 taking a one over here and inserting a one over here, so it is obviously dominating column number 9 similarly, column number is also taken there is a one over here and extra one over here. So again P_{10} is covered by P_6 even for P_{11} there is a one over here, one over here and extra one over here so P_6 is domination 9 and 10 so obviously, 9, 10, 11 get eliminated.

(Refer Slide Time: 56:48)

Example of Branch and Bound applied to Unate covering

In this matrix Lower Bound = 2 comprising (5,7). So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

The left edge from root note indicates P15 being taken (tree). Also in the edge it is marked that "P19, P110, P111"—this indicates that because of taking P15, P19, P110 and P111 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P110, P111) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1

(Refer Slide Time: 58:49)

Example of Branch and Bound applied to Unate covering

The left edge from root note indicates P11 being taken. Also in the edge it is marked that "P13, P12, P14"—this indicates that because of taking P11, P13 has to be considered and P12 and P14 get eliminated.

After reduction (i.e., taking P11, P13 and eliminating P12, P13) we get the following matrix. As of now, the cost of the solution is 2 (P11, P13).

	P15	P16	P17	P18	P19	P110	P111
minterm5	1	1	0	0	0	1	0
minterm6	0	1	1	0	1	0	0
minterm7	0	0	1	1	0	0	0
minterm8	0	1	0	1	0	1	0
minterm9	1	0	0	0	1	1	1
minterm10	1	0	0	1	1	0	0
minterm11	1	0	1	0	0	0	1

= 2

(Refer Slide Time: 58:59)

Example of Branch and Bound applied to Unate covering


In this matrix Lower Bound = 2 comprising {5,7}. So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

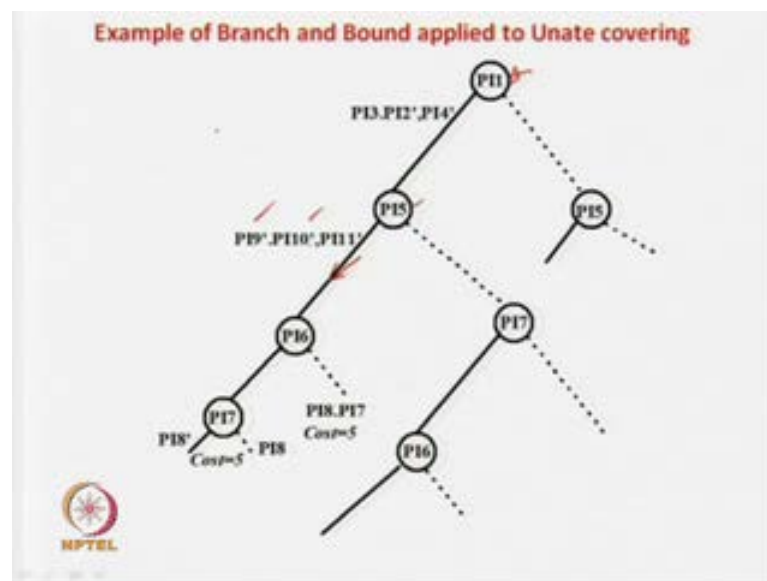
The left edge from root note indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11"—this indicates that because of taking P15, P19, P10 and P11 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P10, P11) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1



(Refer Slide Time: 59:11)



So 9 10 and 11 get eliminated by this one and finally, you are having a matrix of this size. So this one is also reduced not there so obvious p 1 you have taken. So this one is a really small matrix left behind this one only this sub part only. So this one actually this one only 6, 7, 8, 9. So where we are now from left side edge PI 5 is taken. So as you can see PI 5 is taken, so 9 10 and 11 all re primes over here because they get eliminated by dominance rule. So we are at this case. So we are at this case.

(Refer Slide Time: 59:20)

Example of Branch and Bound applied to Unate covering


In this matrix Lower Bound = 2 comprising (5,7). So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

The left edge from root node indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11" —this indicates that because of taking P15, P19, P110 and P111 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P10, P11) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1



Handwritten notes: An arrow points to the matrix, and the words "Lower bound" are written in red.

(Refer Slide Time: 59:48)

Example of Branch and Bound applied to Unate covering


In this matrix Lower Bound = 2 comprising (5,7). So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

The left edge from root node indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11" —this indicates that because of taking P15, P19, P110 and P111 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P10, P11) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1



Handwritten notes: An arrow points to the matrix, and the words "Lower bound" are written in red.

(Refer Slide Time: 60:11)

Example of Branch and Bound applied to Unate covering


In this matrix Lower Bound = 2 comprising {5,7}. So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

The left edge from root node indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11"—this indicates that because of taking P15, P19, P110 and P111 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P110, P111) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18	
minterm6	1	1	0	-2
minterm7	0	1	1	-2
minterm8	1	0	1	-2



So now the question is that, this is a reduced matrix what you have to apply? What you have to apply? The lower bound lower bound algorithm on this one, now this one is taken, now this one in the left edge node taken into the tree again in edge, these are all marked as prime this taking PI 5 and PI 9 actually taking PI 5 it says that if you consider PI 5 eliminates this three what is actually represented by this part. Now the reduced matrix is this one, so again you have to apply the lower bound computation in this. So what is the minimum cost? Now minimum cost is one PI 3, PI 4 now you have to find out the lower cost lower bound and you have to add this one. So in this matrix lower bound is one just let us see what that mean.

So in this case if you put the weight is two in this case weight is two in this case weight is two and this case weight is two. Now you consider this if you take this, this will be getting covered and this two are common and getting covered. So it says that the lower bound is only one but, being fact that you can easily observe that nobody can give you a solution in one, it is always two. If you consider PI 6, PI 7 or PI 7, PI 8 is a cyclic matrix. So the bound is actually two this if you take the lower bound for the heuristic which we are taken. So that it what happen? It is saying that the weight is two, weight is two, weight is two and just take PI 6 right PI 6 this one is the virtue of this one is the common for this one, this is where the inaccuracy comes into the picture and this actually shown by this matrix stated in this.

(Refer Slide Time: 61:06)

Computation of the Lower Bound

The key feature of this algorithm is it just chooses the "shortest" row, that is, the row with the fewest nonzero columns, and breaking ties in ascending order.

The basic motivation of choosing the "shortest" row first, comes from the fact—shorter the row is, higher is the probability of involving a column to cover it. If there is a row of $w=1$, then one column is mandatory to cover it. All rows with at least one common column of 1 are deleted because, all the rows can be covered with the common columns. This makes the solution faster because we eliminate many rows quickly.

However, this also leads to inaccuracy explained as follows. Let row1 be selected to be deleted. Let row2 has common column as column1 (with row1) and row3 has common column as column2 (with row1). So the lower bound estimate in this case is 1; deletion of row1 will also eliminate row2 and row3 because of common columns having 1. Also, let row2 and row3 be singletons. To cover row1 and row2 if we select column1, then selection of column2 is mandatory for row3. Similarly, to cover row1 and row3 if we select column2, then selection of column1 is mandatory for row2. Therefore, we require two columns in this case, which is not reflected in the estimate.

(Refer Slide Time: 61:25)

Example of Branch and Bound applied to Unate covering

In this matrix Lower Bound = 2 comprising (5,7). So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

The left edge from root node indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11"—this indicates that because of taking P15, P19, P110 and P111 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P110, P111) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1

(Refer Slide Time: 62:05)

Example of Branch and Bound applied to Unate covering


In this matrix Lower Bound = 2 comprising (5,7). So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

The left edge from root node indicates P15 being taken (tree). Also in the edge it is marked that "P19, P110, P111"—this indicates that because of taking P15, P19, P110 and P111 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P110, P111) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18	
minterm6	1	1	0	-
minterm7	0	1	1	-
minterm8	1	0	1	-




So this is inaccurate why this leads to inaccuracy is that the example I have said told over just read this what this paragraph actually states what is happening in this case? So it is saying that the requirement is one. So when can the reward be one only when the columns having one it is not the case but, still the heuristic will tell you that in this case the lower bound is one because in this heuristic is simple. So they are not given the factor that given that whenever you are generating that lower bound is one nobody is checking that where the reality is there they are not checking just saying that the lower bound is just following the rates and reduction.

So whatever you get they are not putting an extra check if really lower bound is 1 or can anybody find the solution all this things are there because you have to make the lower bound algorithm fast and quickly you have to get the estimate that is the idea. So this one will give you a bound of one so, what is the present cost P1 1, P1 3 and P1 and if the lower bound is one add one and yes you find out that answer is 4 3 plus 1 is 4 which is very good. So they are still giving you hope that we can find out this one now you are this stage on the tree so.

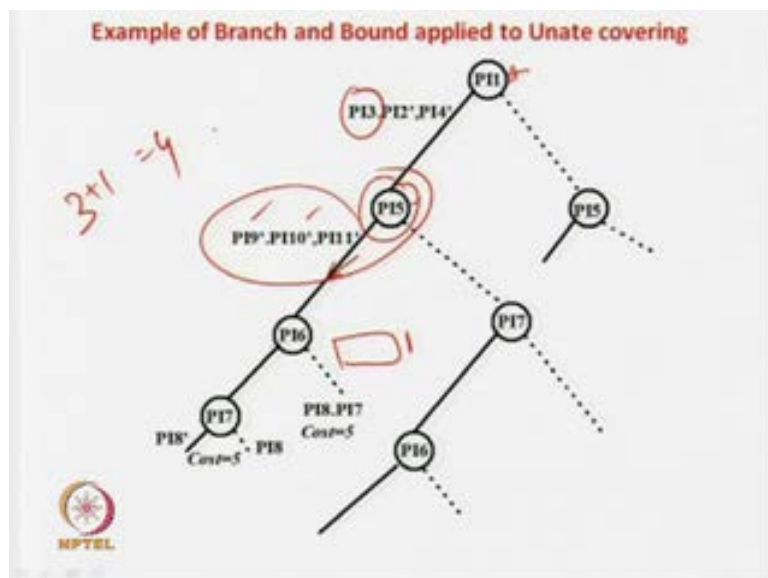
(Refer Slide Time: 62:13)

Example of Branch and Bound applied to Unate covering

- In this matrix Lower Bound = 1 comprising {6}. So if we explore on this matrix the solution cost lower bound is $3+1=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. If we consider P16, the following happens in the matrix
 - row minterm8 is covered,
 - Column P17 or P18 needs to be taken in the cover.
- This is the terminal case as all rows are covered. This is solution is {P11,P13,P15,P16,P17} (left edge of P17) in the tree or {P11,P13,P15,P16,P18} (right edge of P17).
- In both the solutions, the cost is 5, which is higher than the initial expected lower bound. Therefore, we need to retract. We may select an alternative path where, P16=0 (right edge of P16). It is easy to observe that in this case the solution is {P11,P13,P15,P17,P18}; as the cost is 5 we retract. We may select an alternative path where, P15=0 (right edge of P15).



(Refer Slide Time: 62:23)



(Refer Slide Time: 63:13)

Example of Branch and Bound applied to Unate covering

In this matrix Lower Bound = 2 comprising (5,7). So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

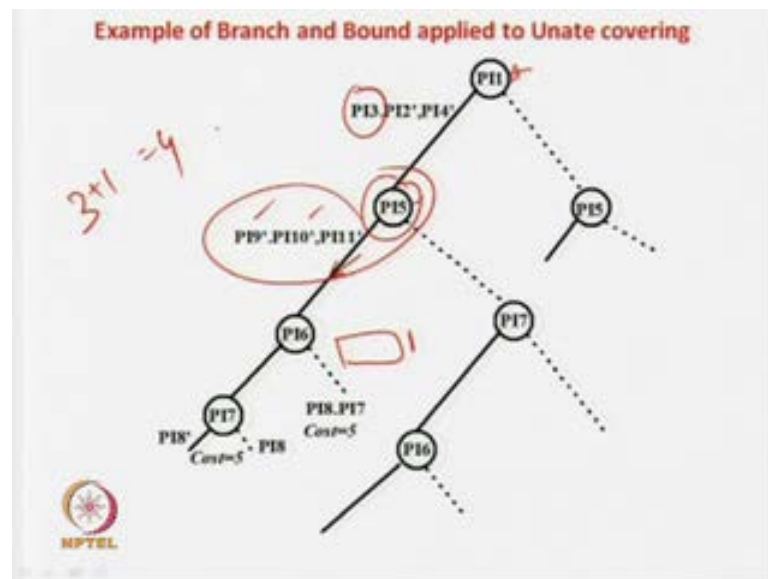
- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

The left edge from root node indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11" — this indicates that because of taking P15, P19, P110 and P111 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P110, P11) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1

(Refer Slide Time: 62:23)



Now you are here you generated a matrix at the cost of one now your present cost is P11 and P13 P12 plus P15. You have taken 1 2 and sorry P11 and P15 you have taken and also you have considered you have taken P16, P11, P15 and so on three values you have taken P11 P13 and P15 this minterms prime implicant you have taken the cost was three and here the intermediate matrix small cost of matrix is one. So you are actually adding a value of 1 over here that is equal to 4, that is equal to 1 bound and hope that you can explore in this path. Now what can be the exploring the path. So that I can take P16 as one that is the case so, you are taking P16 as one going by this path. If you take if you

see if you take PI 6 you are actually covering this guy actually covering this guy. So automatically sorry so this will be eliminated and this will be eliminated by default you are actually coming over here now PI 6 you have taken you have come over here.

(Refer Slide Time: 63:13)

Example of Branch and Bound applied to Unate covering

In this matrix Lower Bound = 2 comprising {5,7}. So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

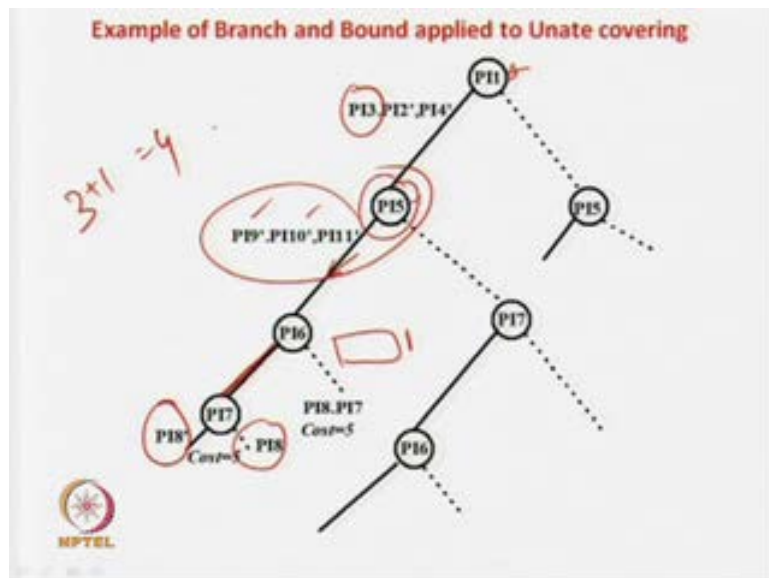
- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

The left edge from root node indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11" — this indicates that because of taking P15, P19, P110 and P111 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P110, P11) we get the following matrix. As of now the cost of the solution is 3 (P1, P13, P15).

	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1

(Refer Slide Time: 63:35)



Now what happen? If you consider this by default you have taken this by default. So in this case PI 6 is there, is there a choice you will take PI 7 or PI 8. Now the choice is there by taking PI 7 minterm 6 and minterm 8 have covered remaining is minterm 7 any one of them you have to do by considering PI 7 or PI 8. Now you do that this is your path. So

you may take PI 8 or you may not take PI 8. So if you take just you consider PI 6 you have taken and PI 7 you have taken this is the case. Now we go back and see you have taken PI 6 and you now also considered PI 7 that is the path you are averaging the path and you are done.

(Refer Slide Time: 64:07)

Example of Branch and Bound applied to Unate covering

In this matrix Lower Bound = 2 comprising {5,7}. So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- *rows minterm9, minterm10, and minterm11 are covered,
- *columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

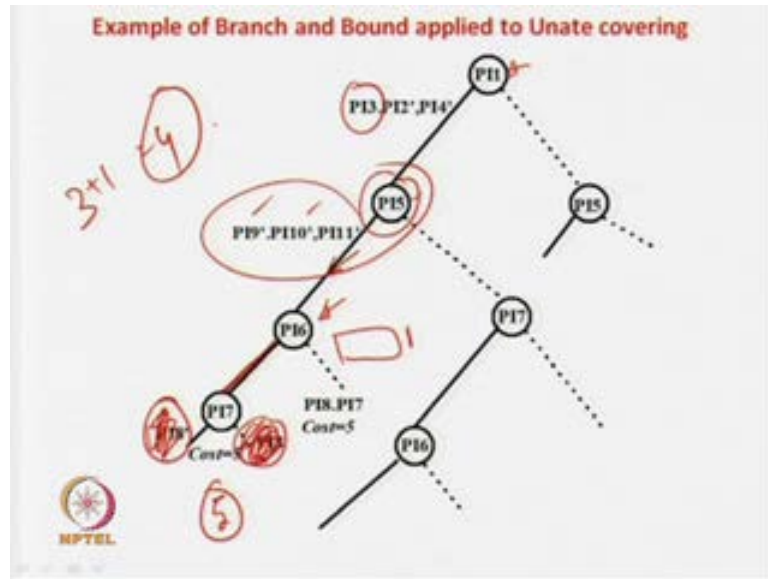
The left edge from root node indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11"—this indicates that because of taking P15, P19, P110 and P111 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P110, P11) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

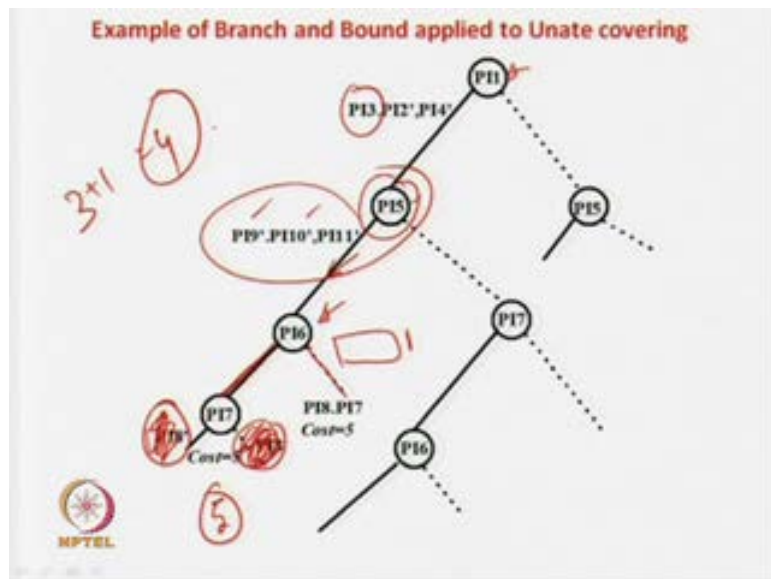
	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1

So PI 6 you have taken PI 7 you have taken both are covered. PI 7 you take so minterm is covered. So this one is the case this is the terminal case. So you may not even explore this path, you may not explore. So PI 6 you have taken and PI 7 you have taken this is the path. So the terminal case so i am sorry this is the terminal case over here.

(Refer Slide Time: 64:23)



(Refer Slide Time: 65:25)



(Refer Slide Time: 65:29)

Example of Branch and Bound applied to Unate covering

In this matrix Lower Bound = 2 comprising {5,7}. So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

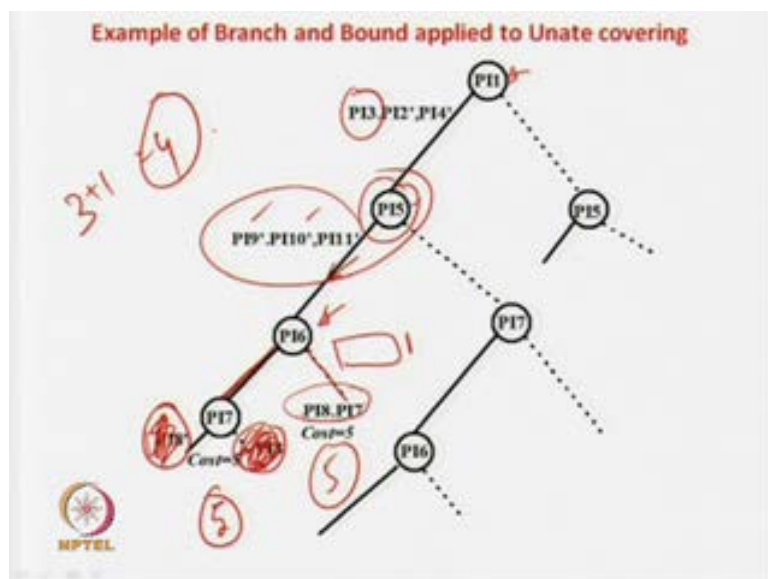
- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

The left edge from root note indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11" —this indicates that because of taking P15, P19, P110 and P111 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P110, P11) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	0	
minterm7	1	1	
minterm8	0	1	

(Refer Slide Time: 65:49)



So it says that PI 1 you have taken PI 3 becomes essential and come down at this stage you found out the matrix cost was 1 3 plus 4. So you explore it is the case you take PI 6 and PI 7. Now the cost is equal to the 5. 1, 2, 3, 4 and 5 that is more than equal to 4; that means, what the lower bound computation has misguided you just say that this matrix cost is 1.

So you have seen let me explore started exploring the path you have to take you have taken PI 6 and you have taken PI 7 then you have found that terminal case everything is

(Refer Slide Time: 66:09)

Example of Branch and Bound applied to Unate covering


In this matrix Lower Bound = 2 comprising {5,7}. So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P10 (P16) and P11 (by P16) are dominated

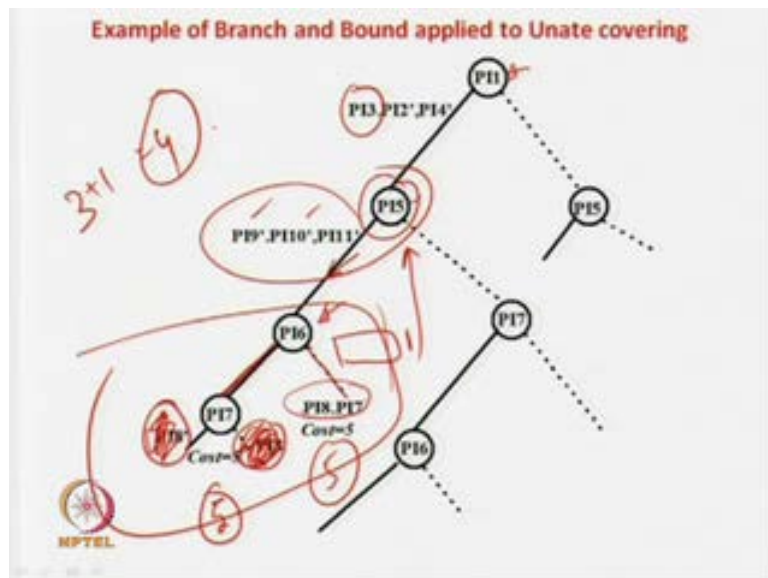
The left edge from root note indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11"—this indicates that because of taking P15, P19, P10 and P11 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P10, P11) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	0	
minterm7	1	1	
minterm8	0	1	



(Refer Slide Time: 66:20)




(Refer Slide Time: 66:43)

Example of Branch and Bound applied to Unate covering

The left edge from root node indicates PI1 being taken. Also in the edge it is marked that "PI3, PI2, PI4"—this indicates that because of taking PI1, PI3 has to be considered and PI2 and PI4 get eliminated.

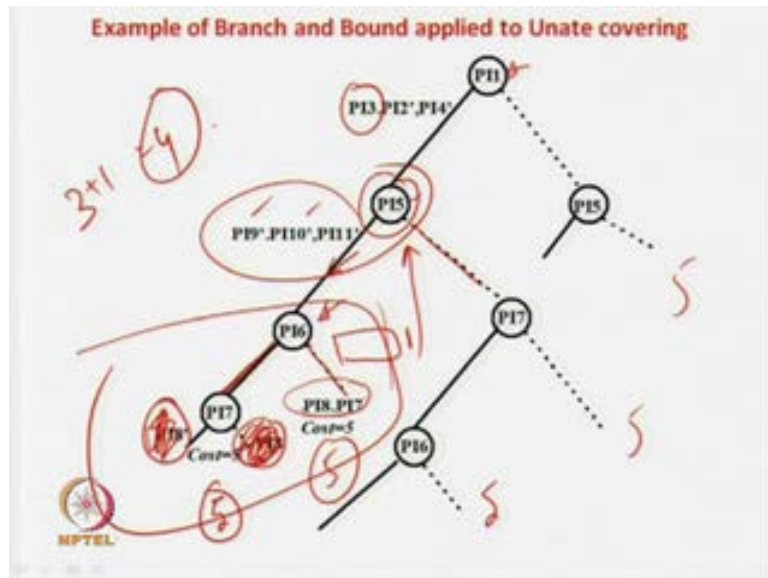
After reduction (i.e., taking PI1, PI3 and eliminating PI2, PI3) we get the following matrix. As of now, the cost of the solution is 2 (PI1, PI3).

	PI5	PI6	PI7	PI8	PI9	PI10	PI11	
minterm5	1	1	0	0	0	1	0	<div style="border: 1px solid red; border-radius: 50%; width: 40px; height: 40px; display: flex; align-items: center; justify-content: center; margin: 0 auto;">2</div> <div style="margin: 5px 0;">2</div> <div style="margin: 5px 0;">4</div>
minterm6	0	1	1	0	1	0	0	
minterm7	0	0	1	1	0	0	0	
minterm8	0	1	0	1	0	1	1	
minterm9	1	0	0	0	1	1	1	
minterm10	1	0	0	1	1	0	0	
minterm11	1	0	1	0	0	0	1	



This was saying that cost is equal to 4 sorry cost is equal to 2, is a false one. In fact the cost is 4, you can explore there and you find out there sorry the cost is 1 you explore there. So the minimum cost of this matrix is 1, so we explore this part of the sub tree this part of the sub tree we explore this is part of the tree. We explored and found out this is not the fact and at every cost is 2 over here. So the total cost is 5 we add the initial bound. So we missed the problem, so what you have to do, is we have to go back again to this node and try. Now we are actually telling lie and second matrix is saying sorry cost is equal to 2. In fact it is wrong, so there is some problem over this. So you have to try with some other node by not taking it with PI 5. Initially we started taking PI 5 and come around.

(Refer Slide Time: 66:58)



(Refer Slide Time: 67:26)

Example of Branch and Bound applied to Unate covering

- In this matrix Lower Bound = 1 comprising {6}. So if we explore on this matrix the solution cost lower bound is $3+1=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. If we consider P16, the following happens in the matrix
 - row minterm8 is covered,
 - Column P17 or P18 needs to be taken in the cover.
- This is the terminal case as all rows are covered. This is solution is $\{P11, P13, P15, P16, P17\}$ (left edge of P17) in the tree or $\{P11, P13, P15, P16, P18\}$ (right edge of P17).
- In both the solutions, the cost is 5, which is higher than the initial expected lower bound. Therefore, we need to retract. We may select an alternative path where, $P16=0$ (right edge of P16). It is easy to observe that in this case the solution is $\{P11, P13, P15, P17, P18\}$; as the cost is 5 we retract. We may select an alternative path where, $P15=0$ (right edge of P15).

NPTEL


So now there can be another path which is not taking 5. So you explore the whole tree and you have to find out the cost is equal to 5. So this is the case with an example. I illustrated the whole theory in an example, so this the big problem that has come. You have to explore the whole tree is the term for example, in this slide this is the terminal case. So whatever I told you, this is the terminal case so this is another terminal case. All those things are written over here, so this one terminal case is P1 1 P1 6 and P1 7 in left

and the other was in PI 1 PI 3 PI 6 and PI 8. That is in another terminal case, we are not considering the PI 7 so that is right edge of PI 7.

(Refer Slide Time: 68:06)

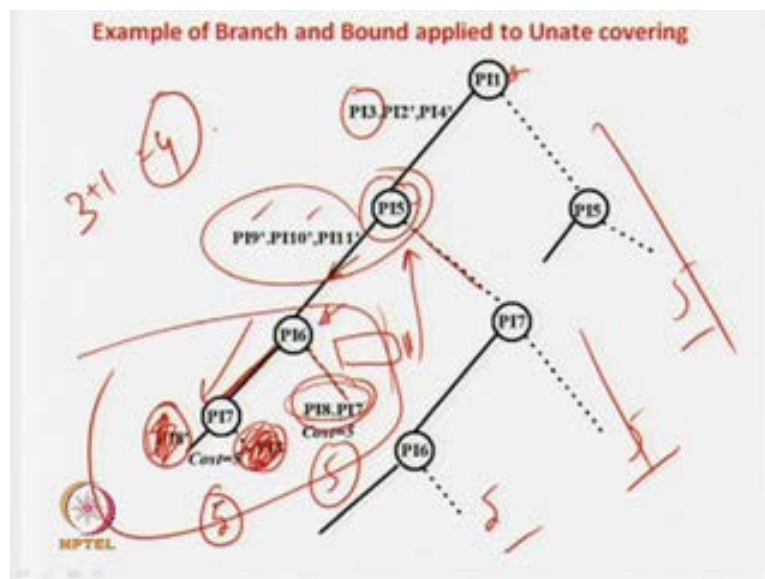
Example of Branch and Bound applied to Unate covering

- Similarly, the whole tree can be created. In this example, if the whole tree is created, it may be noted that the cost at all branches (solution space) is 5.
- Therefore, the whole tree will be explored and in the end, it will be concluded that the lower bound (=4) given by the estimate is not sharp and a solution (of cost 5) will be taken as cover.
- However, it is not always the case. In the Question and Answer part of the lecture we will provide a modified lower bound estimate algorithm and show how, in the same example (being considered in this sub-section) some paths need not be explored



Everywhere you try to explore you will find the cost is this one. This is the another solution you can see in both the exploration. The cost is five which we had in initial. So we have to select an alternative path where PI 6 is equal to 0.

(Refer Slide Time: 68:59)



(Refer Slide Time: 69:16)


Question and Answers

•**Question:** The lower bound estimate algorithm discussed in this lecture may not always give a sharp bound. Suggest suitable modifications and show that better bounds can be obtained. Also show using an example, how benefit is achieved in branch and bound algorithm using the modification .

Answer

The following 4 steps were present in the lower bound estimate algorithm discussed in the lecture. The modification is highlighted as bold.

1. Add a field w to each row, whose value is equal to the number of 1's in the row
2. Choose the row with minimum w . Let it be r_i . If there are multiple rows with same value, choose the one from the top.
3. Delete all rows r_j such that r_i and r_j have at least one column where both of them have a 1. Also, delete r_i . **If, after deletion no more rows remain, then check if there exists a column PI say, such that PI has all 1s. If there is no such column then 1 needs to be added the lower bound.**
4. Repeat step 2 and 3 until no more rows remain.




(Refer Slide Time: 68:06)

Example of Branch and Bound applied to Unate covering

•Similarly, the whole tree can be created. In this example, if the whole tree is created, it may be noted that the cost at all branches (solution space) is 5.

•Therefore, the whole tree will be explored and in the end, it will be concluded that the lower bound (=4) given by the estimate is not sharp and a solution (of cost 5) will be taken as cover.

• However, it is not always the case. In the Question and Answer part of the lecture we will provide a modified lower bound estimate algorithm and show how, in the same example (being considered in this sub-section) some paths need not be explored



You can easily see that, again the cost is equal to 5. So the solution will be PI 1, PI 3, PI 7 and PI 8. So this is this solution. So this is this solution PI 7 and PI 8, everywhere it will be 8. Now you explore the whole tree. Every where the solution will be 5. So in this case, the example is how the branch and bound algorithm works with heuristic. And also another example an another fact is that, we have landed into a problem here. In case every part of the tree, you have to explore. Because your heuristic was giving a wrong value 1

So it was a wrong value in your case or a very tight lower bound computation is very tight. In this case very much needed was there, so you have to explore the whole tree. Everywhere you have to get the value 5. So you have to explore most of the tree. So then you can say that why is it like that, how is that lower bound solution helping my problem and helping my solution? If you do not use the heuristics it will explore the large part of the tree. Here also you have to explore a lot of the tree. Because it everywhere it is giving the hope that the cost is 1.

So you add this 3 plus 1 and you go and write the solution. Here also other paths, if you see it will always tell you that which have a solution, which is 4. Every time you start exploring, you find the cost is 5. And hence get a problem. So how does it help you then why actually, what is the advantage of the heuristics? Now the question answers will follow and see that slightly modified with lower bound heuristics and really this lower bound of fruit of this coming into picture.

(Refer Slide Time: 69:44).

Question and Answers

•Question: The lower bound estimate algorithm discussed in this lecture may not always give a sharp bound. Suggest suitable modifications and show that better bounds can be obtained. Also show using an example, how benefit is achieved in branch and bound algorithm using the modification .

Answer

The following 4 steps were present in the lower bound estimate algorithm discussed in the lecture. The modification is highlighted as bold.

1. Add a field w to each row, whose value is equal to the number of 1's in the row
2. Choose the row with minimum w . Let it be r_i . If there are multiple rows with same value, choose the one from the top.
3. Delete all rows r_j such that r_i and r_j have at least one column where both of them have a 1. Also, delete r_i . **If, after deletion no more rows remain, then check if there exists a column PI say, such that PI has all 1s. If there is no such column then 1 needs to be added the lower bound.**
4. Repeat step 2 and 3 until no more rows remain.

Let us the question the question is the lower bound estimate algorithm. In this lecture may not give us the sharp bound. So suggest suitable modifications and show that how better bounds can be obtained. Also show that how benefit is achieved in branch and bound algorithm using the modification.

Because in the lecture flow we have seen that the bound was restrict, that is bound is actually was giving 1 and 3 plus 4. And explore the path and finally, finding it cost is 5.

And then it is actually here. Then the initial cost so explore the full space and then you find out that nobody can give the solution less than 5. So the your answer is 5 minterms and 5 prime implicants. And you take any one of them. So unnecessary you have to explore the whole tree. So we will see a very good what we can say a very good try to improve upon actually a very good improvement of the heuristics, there will be some improvement.

So because the more you make on the heuristics it will become more complex and it will take more complexity in time. So that idea is not there, so you have to make the heuristics as better as possible subject to the complexity. You should not get the huge amount of time and your aim should be it is not possible to explore the whole part of the tree. So let us see what small modification we have done? Add a w to each row to the number of 1s in the row. Same thing you do choose the minimum row if the multiple row with some value, choose from top. So you were adding in the ascending order. Delete all rows where the common 1s in the rows. So delete all rows in terms of all columns in 1, so then delete r_i .

So what we add both at very small computation, if after division no more rows remain then, check if there exist a PI such that PI has all 1s. If no such columns then 1 need to be added to lower bound. So if you see that if you go for a terminal case, in terminal case you are what we are doing is, find out where the common 1s. If they are finding out the common 1s, you replace eliminate those rows. So if case the rows all the rows are eliminated then you say that the bound is added to 1. Now you take one row, all other rows get eliminated, then you say that you adding 1 2 and this 1. So we are adding 1 weight to the lower bound, adding one to the lower bound and all other getting exhausted. Means you correspond the column and rows here. You put something just if you want to add the weight.

(Refer Slide Time: 72:23)

Example of Branch and Bound applied to Unate covering

In this matrix Lower Bound = 2 comprising (5,7). So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

The left edge from root note indicates P15 being taken (tree). Also in the edge it is marked that "P19, P10, P11"—this indicates that because of taking P15, P19, P110 and P111 get eliminated.

After reduction (i.e., taking P15 and eliminating P19, P110, P11) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15). $+1$

	P16	P17	P18		
minterm6	1	1	0	1	-
minterm7	0	1	1	1	=2
minterm8	1	0	1	1	

(Refer Slide Time: 69:44).

Question and Answers

•**Question:** The lower bound estimate algorithm discussed in this lecture may not always give a sharp bound. Suggest suitable modifications and show that better bounds can be obtained. Also show using an example, how benefit is achieved in branch and bound algorithm using the modification .

Answer

The following 4 steps were present in the lower bound estimate algorithm discussed in the lecture. The modification is highlighted as bold.

1. Add a field w to each row, whose value is equal to the number of 1's in the row
2. Choose the row with minimum w . Let it be r_i . If there are multiple rows with same value, choose the one from the top.
3. Delete all rows r_j such that r_i and r_j have at least one column where both of them have a 1. Also, delete r_i . **If, after deletion no more rows remain, then check if there exists a column P1 say, such that P1 has all 1s. If there is no such column then 1 needs to be added the lower bound.**
4. Repeat step 2 and 3 until no more rows remain.

(Refer Slide Time: 72:57)

Question and Answers

The motivation of the extension is explained by the following matrix.

	P11	P12	P13	
minterm1	1	1	0	1
minterm2	0	1	1	1
minterm3	1	0	1	1

① 1

In the above matrix, if we apply the lower bound estimate algorithm discussed in the lecture, then we get the answer as 1; the algorithm stops after 1 iteration because row1 is selected and row2 and row3 get eliminated. However, it may be noted one column cannot cover the three rows because no column P_i exists, such that P_i has all 1s. Therefore, in this case, we increment the cost by 1; two rows can cover. It may be noted that if the matrix has more rows and columns, then the addition required in the cost may be more than 2. However to keep the algorithm simple we compromise on accuracy and add just 1; calculating more accurate value to be added, requires more computation steps.

NPTEL

One just to check the column where everything is a 1, everything is not a 1 there is not meaning 1 you just put. Add up the value 1, that is very obvious if you say that by taking only one column everything gets eliminated. There is a column 1 or there is a odd number of 1's it will not be a case, it will be bit higher than that. In this case you actually you do not find out what is actually, what is higher, just add 1 to it like for example, if you see this in this example if you say that you take this 1, you get eliminated by this commonality. And this one gets eliminated by this commonality. And you set the bound is 1. So we check that is there any column, if not the case you add the value 1. So the lower bound is 2.

So why if you say that lower bound is 1, in that case so there exist a column, if you take that column all the 3 rows get eliminated. If rows not there of all 1s it is actually a false value. You have to add a 1 to it, this is a slight modification you have to add to it. Then you may added one to keep all the things simple. Just repeat the whole thing, so this is this case just I told you this one is common with this one eliminate this. This one is common with this one eliminate this.

(Refer Slide Time: 73:15)

Question and Answers

Let us again consider the same constraint matrix example. If we apply the modified lower bound estimate algorithm, we get Lower Bound = 4 comprising {1, 3, 5, 7} (same as the original algorithm of the lecture).

	PI1	PI2	PI3	PI4	PI5	PI6	PI7	PI8	PI9	PI10	PI11
minterm1	1	1	0	0	0	0	0	0	0	0	0
minterm2	0	1	1	0	0	0	0	0	0	0	0
minterm3	0	0	1	1	0	0	0	0	0	0	0
minterm4	1	0	0	1	0	0	0	0	0	0	0
minterm5	0	0	0	0	1	1	0	0	0	1	0
minterm6	0	0	0	0	0	1	1	0	1	0	0
minterm7	0	0	0	0	0	0	1	1	0	0	0
minterm8	0	0	0	0	0	1	0	1	0	1	1
minterm9	0	0	0	0	1	0	0	0	1	1	1
minterm10	0	0	0	0	1	0	0	1	1	0	0
minterm11	0	0	0	0	1	0	1	0	0	0	1
minterm12	1	0	0	0	0	0	0	0	0	0	1


(Refer Slide Time: 73:47)

Question and Answers

After third iteration of the modified lower bound estimate algorithm (i.e., deleting rows corresponding to 1, 3 and 5), we have the following matrix

	PI1	PI2	PI3	PI4	PI5	PI6	PI7	PI8	PI9	PI10	PI11
minterm7	0	0	0	0	0	0	1	1	0	0	0

After eliminating minterm7, we exhaust all rows, but need not add 1 to the cost because there are two rows PI7 and PI8, which have all 1s.



(Refer Slide Time: 73:15)

Question and Answers

Let us again consider the same constraint matrix example. If we apply the modified lower bound estimate algorithm, we get Lower Bound = 4 comprising {1, 3, 5, 7} (same as the original algorithm of the lecture).

	PI1	PI2	PI3	PI4	PI5	PI6	PI7	PI8	PI9	PI10	PI11
minterm1	1	1	0	0	0	0	0	0	0	0	0
minterm2	0	1	1	0	0	0	0	0	0	0	0
minterm3	0	0	1	1	0	0	0	0	0	0	0
minterm4	1	0	0	1	0	0	0	0	0	0	0
minterm5	0	0	0	0	1	1	0	0	0	1	0
minterm6	0	0	0	0	0	1	1	0	1	0	0
minterm7	0	0	0	0	0	0	1	1	0	0	0
minterm8	0	0	0	0	0	1	0	1	0	1	1
minterm9	0	0	0	0	1	0	0	0	1	1	1
minterm10	0	0	0	0	1	0	0	1	1	0	0
minterm11	0	0	0	0	1	0	1	0	0	0	1
minterm12	1	0	0	0	0	0	0	0	0	0	1

(Refer Slide Time: 74:48)

Question and Answers

After third iteration of the modified lower bound estimate algorithm (i.e., deleting rows corresponding to 1, 3 and 5), we have the following matrix

	PI1	PI2	PI3	PI4	PI5	PI6	PI7	PI8	PI9	PI10	PI11
minterm7	0	0	0	0	0	0	1	1	0	0	0

After eliminating minterm7, we exhaust all rows, but need not add 1 to the cost because there are two rows PI7 and PI8, which have all 1s.

4

Now actually the value is 1, but, there is no column with a 1. So you have add up to it. So what is the heuristics, for modification you replace the paragraph and left tree and again we take the whole example. We are not going to it. We slowly show the benefits. So this is our old example in this case the lower bound in this case is 4. You should go by the old technique, so if you find out the 1, 3, 7 if you are using old lecture and I mean i mean the old in this case the value is 2. In this case the value is two. And you try to reduce it, we will find the answer is 1, 3, 5 and 7. That is going to be your answer, you will get the

answer if you are using the heuristics. Now what we have done in this case the last element will be this one.

So now we are having 1, 3 and 5. And this 1 you are having now in this case you have to find out the, if you take few of the 7, this 1 will exhaust all the rows. Hardly 2, 1 in this case eliminate the case minterm 7 in which you keep upon you did. With this you will find out that we will reach the solution. Here in this case after eliminating the minterm, 7 we exhaust all the rows but, need not add 1 to the cost. Because there are no 2 rows PI 7 and PI 8 which all are having 1.

(Refer Slide Time: 75:46)

Question and Answers

After third iteration of the modified lower bound estimate algorithm (i.e., deleting rows corresponding to 1,3 and 5), we have the following matrix

	PI1	PI2	PI3	PI4	PI5	PI6	PI7	PI8	PI9	PI10	PI11
minterm7	0	0	0	0	0	0	1	1	0	0	0

After eliminating minterm7, we exhaust all rows, but need not add 1 to the cost because there are two rows PI7 and PI8, which have all 1s.

So let us see what is meant by that. So in this case if you start eliminating the row and all which you have seen at this case what you will land after the 3rd iteration, first iteration will land up in 1, 3 and 5. And we will have the following matrix after eliminating minterm 7, we will exhaust all the rows. We find out that all the rows and some 7 will be added. Now you say that lower bound is equal to the 4 1 1 sorry 2 3 and for this 7 you can take 7 minterm 7. So you have to include min term 7 also. So it will be 1 2, 3 and last 1, 4 will also be there. So answer is inclusion of the 4 is actually adding a 7.

So if you look at the our heuristics it will add one more. Why is that we say because after elimination 7, we exhaust all the rows. But, we need not add cost to this case. We will see because there is two rows PI 7 and PI 8. All 1s, so we are doing we are actually exploring this initial bound matrix by our method, not by the old method. But, if you

apply old method, you will get 1, 3, 5 and 7 and cost will be equal to 4. In our case in the terminal case what we will find out, in last case this matrix will be available to you and in this case we will find out the 1, 3 and 5 already taken. And so it will be adding 7, it will finish up the it s now for value 7. You will be adding the value of 1 or value of 2 to old heurists. We will add the value of 1 but, our heurists, what we will add but, our heurists.

(Refer Slide Time: 76:39)

Question and Answers

After third iteration of the modified lower bound estimate algorithm (i.e., deleting rows corresponding to 1,3 and 5), we have the following matrix

	P11	P12	P13	P14	P15	P16	P17	P18	P19	P110	P111
minterm7	0	0	0	0	0	0	1	1	0	0	0

After eliminating minterm7, we exhaust all rows, but need not add 1 to the cost because there are two rows P17 and P18, which have all 1s.

Example of Branch and Bound applied to Unate covering

MPTEL

Example of Branch and Bound applied to Unate covering

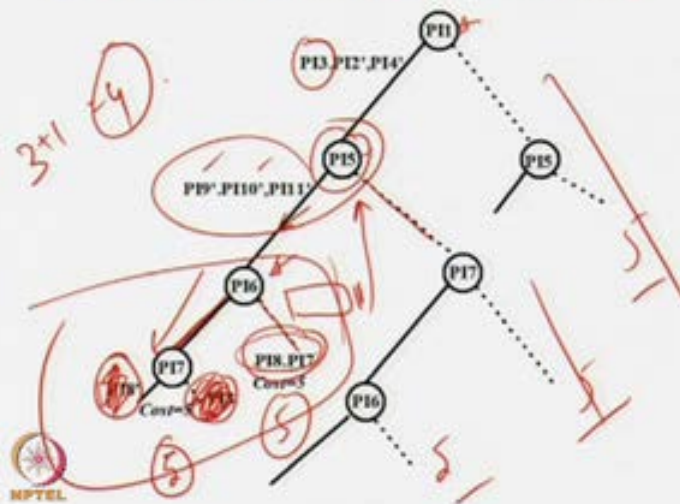
•Similarly, the whole tree can be created. In this example, if the whole tree is created, it may be noted that the cost at all branches (solution space) is 5.

•Therefore, the whole tree will be explored and in the end, it will be concluded that the lower bound (=4) given by the estimate is not sharp and a solution (of cost 5) will be taken as cover.

• However, it is not always the case. In the Question and Answer part of the lecture we will provide a modified lower bound estimate algorithm and show how, in the same example (being considered in this sub-section) some paths need not be explored



Example of Branch and Bound applied to Unate covering



Question and Answers

•**Question:** The lower bound estimate algorithm discussed in this lecture may not always give a sharp bound. Suggest suitable modifications and show that better bounds can be obtained. Also show using an example, how benefit is achieved in branch and bound algorithm using the modification .

Answer

The following 4 steps were present in the lower bound estimate algorithm discussed in the lecture. The modification is highlighted as bold.

1. Add a field w to each row, whose value is equal to the number of 1's in the row
2. Choose the row with minimum w . Let it be r_i . If there are multiple rows with same value, choose the one from the top.
3. Delete all rows r_j such that r_i and r_j have at least one column where both of them have a 1. Also, delete r_i . **If, after deletion no more rows remain, then check if there exists a column P_i say, such that P_i has all 1s. If there is no such column then 1 needs to be added to the lower bound.**
4. Repeat step 2 and 3 until no more rows remain.



Question and Answers

The motivation of the extension is explained by the following matrix.

	P1	P2	P3	
minterm1	0	0	0	1
minterm2	0	0	1	
minterm3	0	0	1	

In the above matrix, if we apply the lower bound estimate algorithm discussed in the lecture, then we get the answer as 1; the algorithm stops after 1 iteration because row1 is selected and row2 and row3 get eliminated. However, it may be noted one column cannot cover the three rows because no column P1 exists, such that P1 has all 1s. Therefore, in this case, we increment the cost by 1; two rows can cover. It may be noted that if the matrix has more rows and columns, then the addition required in the cost may be more than 2. However to keep the algorithm simple we compromise on accuracy and add just 1; calculating more accurate value to be added, requires more computation steps.

NPTEL

Question and Answers

Let us again consider the same constraint matrix example. If we apply the modified lower bound estimate algorithm, we get Lower Bound = 4 comprising {1, 3, 5, 7} (same as the original algorithm of the lecture).

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
minterm1	1	1	0	0	0	0	0	0	0	0	0
minterm2	0	1	1	0	0	0	0	0	0	0	0
minterm3	0	0	1	1	0	0	0	0	0	0	0
minterm4	1	0	0	1	0	0	0	0	0	0	0
minterm5	0	0	0	0	1	1	0	0	0	1	0
minterm6	0	0	0	0	0	1	1	0	1	0	0
minterm7	0	0	0	0	0	0	1	1	0	0	0
minterm8	0	0	0	0	0	1	0	1	0	1	1
minterm9	0	0	0	0	1	0	0	0	1	1	1
minterm10	0	0	0	0	1	0	0	1	1	0	0
minterm11	0	0	0	0	1	0	1	0	0	0	1
minterm12	1	0	0	0	0	0	0	0	0	0	1

Question and Answers

After third iteration of the modified lower bound estimate algorithm (i.e., deleting rows corresponding to 1, 3 and 5), we have the following matrix

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
minterm7	0	0	0	0	0	0	1	1	0	0	0

After eliminating minterm7, we exhaust all rows, but need not add 1 to the cost because there are two rows P7 and P8, which have all 1s.

NPTEL

(Refer Slide Time: 76:55)


Question and Answers

As discussed in the lecture, we start exploring the solution space by selecting P_{11} (i.e., $P_{11}=1$). With P_{11} being taken, the following happens in the matrix

- rows $minterm_1$, $minterm_4$, and $minterm_{11}$ are covered,
- columns P_{12} (by P_{13}) and P_{14} (by P_{13}) are dominated
- column P_{13} becomes essential.

After reduction (i.e., taking P_{11} , P_{13} and eliminating P_{12} , P_{13}) we get the following matrix. As of now, the cost of the solution is 2 (P_{11} , P_{13}).

	P_{15}	P_{16}	P_{17}	P_{18}	P_{19}	P_{110}	P_{111}
$minterm_5$	1	1	0	0	0	1	0
$minterm_6$	0	1	1	0	1	0	0
$minterm_7$	0	0	1	1	0	0	0
$minterm_8$	0	1	0	1	0	1	1
$minterm_9$	1	0	0	0	1	1	1
$minterm_{10}$	1	0	0	1	1	0	0
$minterm_{11}$	1	0	1	0	0	0	1



In this case you will add a value 1, because there is two columns where all the values are 1. This is very simple since the row is actually 1. This column has all 1s and this column has all 1s. So by both the technique, our heuristic and other heuristic lower bound will be equal to, but, now you see what will be the advantage? But, in the old heuristic and our heuristic will be same. But, we are not added that. Only thing we have to note that in the old example the old heuristic they will take $minterm_4$, we say that the value is equal to the 4. But, in our case the we also say that the value is 4. We have added 7 and we have added the value of 1. But, we have two columns, where we have two 1s. So for heuristic matrix both of them are giving the same algorithm.

(Refer Slide Time: 77:24)


Question and Answers

In this matrix Lower Bound = 2 comprising {5,7}; this is same, using the original lower bound computation algorithm as well as the modified one. So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

After reduction (i.e., taking P15 and eliminating P19, P110, P111) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1



(Refer Slide Time: 77:44)

Question and Answers


As discussed in the lecture, we start exploring the solution space by selecting P11 (i.e., P11=1). With P11 being taken, the following happens in the matrix

- rows minterm1, minterm4, and minterm112 are covered,
- columns P12 (by P13) and P14 (by P13) are dominated
- column P13 becomes essential.

After reduction (i.e., taking P11, P13 and eliminating P12, P13) we get the following matrix. As of now, the cost of the solution is 2 (P11, P13).

	P15	P16	P17	P18	P19	P110	P111
minterm5	1	1	0	0	0	1	0
minterm6	0	1	1	0	1	0	0
minterm7	0	0	1	1	0	0	0
minterm8	0	1	0	1	0	1	1
minterm9	1	0	0	0	1	1	1
minterm10	1	0	0	1	1	0	0
minterm11	1	0	1	0	0	0	1

S1?



(Refer Slide Time: 77:24)


Question and Answers

In this matrix Lower Bound = 2 comprising {5,7}; this is same, using the original lower bound computation algorithm as well as the modified one. So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- *rows minterm9, minterm10, and minterm11 are covered,
- *columns P19 (by P16), P10 (P16) and P11 (by P16) are dominated

After reduction (i.e., taking P15 and eliminating P19, P10, P11) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1



(Refer Slide Time: 78:01)


Question and Answers

As discussed in the lecture, we start exploring the solution space by selecting P11 (i.e., P11=1). With P11 being taken, the following happens in the matrix

- *rows minterm1, minterm4, and minterm12 are covered,
- *columns P12 (by P13) and P14 (by P13) are dominated
- *column P13 becomes essential.

After reduction (i.e., taking P11, P13 and eliminating P12, P13) we get the following matrix. As of now, the cost of the solution is 2 (P11, P13).

	P15	P16	P17	P18	P19	P110	P111
minterm5	1	1	0	0	0	1	0
minterm6	0	1	1	0	1	0	0
minterm7	0	0	1	1	0	0	0
minterm8	0	1	0	1	0	1	1
minterm9	1	0	0	0	1	1	1
minterm10	1	0	0	1	1	0	0
minterm11	1	0	1	0	0	0	1



(5,7)

So now we are again doing all this thing. If you remember taking I mean you are taking minterm 4, 7 and 12, so you take PI 4 and in mean case PI 4 will all be covered. This will be dominated and PI 4 will be essential reduced matrix after taking PI 1 and PI 3. And eliminating this we get the cost of the following matrix. The cost of the solution is this 1 whole thing we have done. If you remember this 1 in matrix, now if you see in this case the lower bound is 5 and 7. This is same as the original lower bound computation and the modified one.

So if we explore all the solution is 2 plus 2 as on the a values, on the lower bound we explore this say same thing with 5 been taken. So you take 5 old case and you start exploring the matrix. Obviously is the same thing you have to do if you remember. If PI had taken row nor 10, 11 gets eliminated. If you take this column it gets eliminated. Again some row dominance and column dominance. This 2 will get eliminated and finally, if we remember we are going to get this matrix. Now if we try to see the same old flow this 1 you start with this 1, this is the same flow. Previous example is this one and finally, we get this 1. Now we are going to find out the there will be difference in the heuristic, our heuristic and the old heuristic. sorry Old one we discussed, modified heuristic and old will find slight difference. So in this what you are going to find out, the what is the lower bound old heuristic will tell you that, this is common this common. So it is 1. .

(Refer Slide Time: 78:05)

Question and Answers


In this matrix Lower Bound = 2 comprising {5,7}; this is same, using the original lower bound computation algorithm as well as the modified one. So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- *rows minterm9, minterm10, and minterm11 are covered,
- *columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

After reduction (i.e., taking P15 and eliminating P19, P110, P11) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15) $+2$

	P16	P17	P18
minterm6	1	0	1
minterm7	0	1	1
minterm8	1	0	1

$\text{cost} = 5$
 $\text{cost} = 2$
 $\text{cost} = 1$



But, again you have to find out that modified, will check before reporting lower bound it is there, any column all 1s. In this case there is no such 1, you add the value 1 and the lower bound will be now 2. But, now our modified but, it will be the old one. Now you are reporting a value 2 and old 1 reporting the value 1. So it will see that what is the cost? The cost is the cost is P1 1 P1 3, P1 5 lat 2 equal to 5, new heuristic but, the old heuristic will be P1 1 plus P1 5 will be 4 by the modified paths 3 plus will be 5. So it is higher than the initial estimate of 4. So you do not explore on the path and then you are going to get a in this case the lower bound is 1, compromising to 6 and lower bound is

equal to 2. In our case it will be 5. Do not explore on that path because it is higher and we are safe from actually taking some of the paths. Just tell me what happened just happened.

(Refer Slide Time: 79:22)


Question and Answers

In this matrix Lower Bound = 2 comprising (5,7); this is same, using the original lower bound computation algorithm as well as the modified one. So if we explore on this matrix the solution cost lower bound is $2+2=4$. As this value is equal to that of the Lower Bound on the initial matrix, we explore on the search space. Let us now consider column P15. With P15 being taken, the following happens in the matrix

- rows minterm9, minterm10, and minterm11 are covered,
- columns P19 (by P16), P110 (P16) and P111 (by P16) are dominated

After reduction (i.e., taking P15 and eliminating P19, P110, P111) we get the following matrix. As of now the cost of the solution is 3 (P11, P13, P15).

	P16	P17	P18
minterm6	1	1	0
minterm7	0	1	1
minterm8	1	0	1



(Refer Slide Time: 79:26)


Question and Answers

In this matrix Lower Bound = 1 comprising (6); this is the value obtained using the original lower bound computation algorithm of discussed in the lecture.

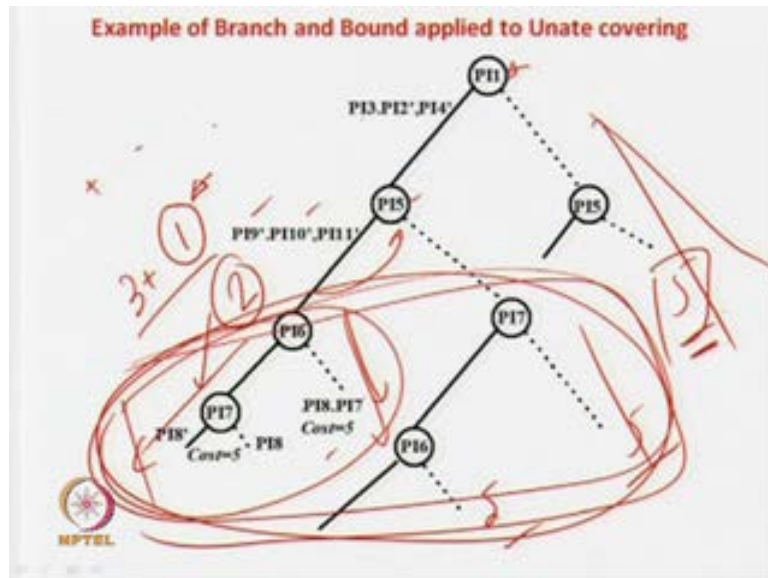
Using the modified algorithm the Lower Bound = 2 (in this case 1 is added to the cost because there is no column with all 1s.)

Therefore, if we explore on this matrix the solution cost lower bound is $3+2=5$. As this value is higher than the Lower Bound on the initial matrix, we do not explore on the search space.

As shown in the lecture, exploring in path P11=1, P15=1....., would have the cost of 5. As this value is available without actually exploring the path, when we use the modified bound computation algorithm, we save computation time in the branch and bound algorithm.



(Refer Slide Time: 79:57)



In this case the same old what will go to actually is going to the same path. So it actually 1 2 3 plus 4 3 plus. Our case old heuristic was reporting 1 and you keep on exploring on this problem and finally, you get a 5. That is the big problem in modified one, we give it equal to 2. So 3 plus 2 is 5. So if you say it do not explore on this path do not explore on this path the value is 5. Immediately you will back track to this and find out the less number of paths. Obviously the lower bound is 5, nobody can give you a solution less than that. The lower bound 4 which was instated by our heuristic or the old heuristic, that was the wrong estimate the initial values has to be 5. But, our heuristic everywhere you find 5 5 5.

So we will explore out this part of the tree. So finally, the leaf node there is no more solution is there. So you come to the conclusion, solution is 5, in what we have that we can find out that before I close. Just see what happened initially we have taken this matrix and old heuristic, and modified heuristic. Both of them are given a lower bound equal to 4. In fact we found out the that is not the correct path, minimum 5 has to be taken but, what is the gain, the gain was the started exploring and exploring which was and finally, we landed into this matrix. If you take the old heuristic lower bound is 1, but, if you take our heuristic the modified heuristic it will lower bound is equal to 2. Then you will say lower bound is 5.

So you will not explore many of the tree unless and until the last case of the that no more to be track. Because this is the only path to explore and terminate is exhausted and you are not explore this path properly. 3 plus 2, the cost is 5, 3 plus 2 every where it will stop and no more no more you have to find and the default cost is 5. And cost is 5. And this so on so what we have said, we have said that we have explored all the paths and just going to the terminal case directly where, if you take the old heuristic this one taking the lower bound 1 keep on exploring all the paths and at the end of the path you will realize the value is 5. Again retrace 5 and keep on going and I have reached the value of fix and that is the solution. But, unnecessarily you have explored more paths. That was not required.

So in modified heuristic, we have achieved this one. So with this this one is actually told, so with this we come to end of the lecture and this module. And what we have seen, that we have taken the unit covering problem. We have found out the algorithm incase a branch and bound algorithm, which can give you the subset of prime implicant, which can cover the function that is what we have done. Again you can see finding out the prime implicant through tabular method or by inter method or by you have to make a table matrix and all at some point of that time, you have to explore the minterms. Exploring the minterms means in terms of exponential case in number of inputs. Again if the algorithm is taking some complexity, the complexity is 2 to the power of n.

If n is the number of inputs the constant matrix will be in 2 to the power of n. Again exploring that the order will be something of the power of n. It is a very fearless algorithm. We tried to make it somewhat minimal somewhat reducing by the heuristic. The next lecture on the next module what you will we will not at all go in terms of minterms, representing a matrix in terms of minterms is only a exponential form. We will not go to that directly, we will put some heuristic and try to find a solution to that what will be looking in the next lecture.

Thank you.