

**VLSI Physical Design**  
**Prof. Indranil Sengupta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 47**  
**Layout Compaction (Part 2)**

So, we continue with our discussion on layout compaction. So, if you recall in our last lecture we had looked at broadly the different categories or techniques for compaction. And we looked at a couple of methods at least the basic principle without going into detail. We looked at the constraint graph method we saw how we can generate the layout related constraints by shining an imaginary light by generating shadows of particular blocks, and we looked at the vertical grid based compaction method also.

(Refer Slide Time: 01:03)



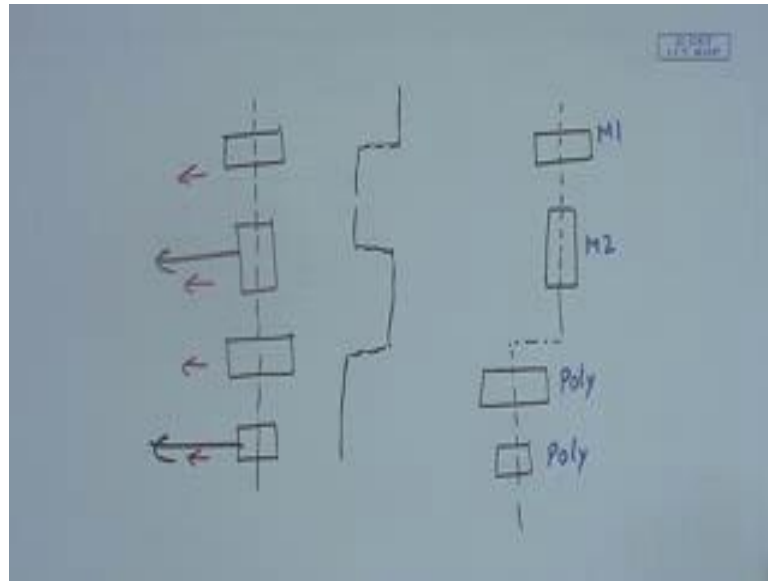
**Split Grid Compaction**

- Here we place distinct circuit features that fall on the same virtual grid separately, splitting the virtual grids where necessary.
- Initially, the compactor identifies groups of circuit features falling on the same virtual grid lines that need to be placed together.
  - Local connectivity is used to identify the groups.
- After the circuit features are grouped together, the compaction is done in two passes:
  - First x-compaction, and then y-compaction.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Today we start by looking at a variation of the virtual grid based compaction. This we call as this split grid compaction. So, here like in the virtual grid we also place distinct circuit features on the same virtual grids separately, but additionally we are allowed to split a virtual grid where necessary.

(Refer Slide Time: 01:40)



So, what it means is that suppose I had defined a grid with some objects already attached to it let say there are 4 objects. Now I am allowed to do something like this, like these 2 objects were there let say the virtual grid is coming. So, I do not disturb these 2, but the other 2 I can bring them a little to the left. So, I am splitting the grid like this.

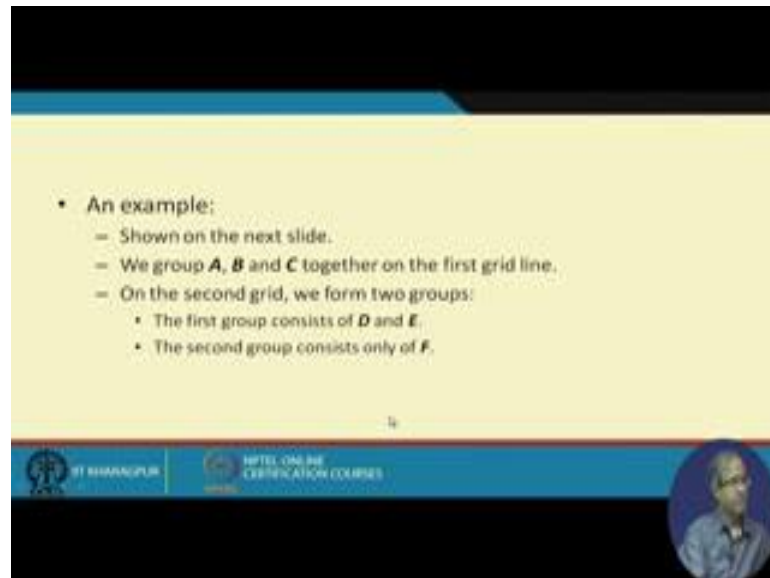
So, in the earlier case for virtual grid compaction, I was shifting these all these 4 blocks together as the grid was shifting, but now I am saying that maybe there are blocks to the left which is not allowing these 2 blocks to be shifted, but maybe these 2 features can be shifted.

Let say it is possible that these 2 are blocks in the metal layer, which has higher distance constraint, but these can be polysilicon layers' polysilicon has separation of only 2 lambda. So, maybe this can be shifted together, but not these right this is the basic idea.

So, initially it works like the virtual grid compaction composition. So, the compactor will identify groups of circuit features that will be falling on the same virtual grid lines and local connectivity issues are then identified like, here in this example there were for this virtual grid there were 4 components now locally you analyze the characteristic of these 4 out of these 4 how many you can shift and how many you cannot shift. So, accordingly you split this grid like the zigzag path. So, it is more generally you can possibly shift this one and then shift this one. So, your grid will look like this right. So, again just like virtual grid compaction here also we do the compaction typically in 2 passes one we do

in the x direction called x compaction then we do it in the y direction this called y compaction.

(Refer Slide Time: 04:17)



• An example:

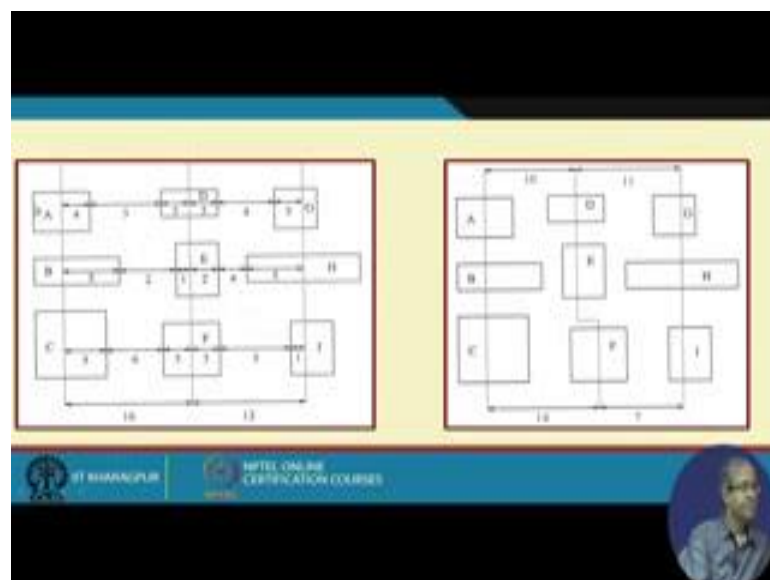
- Shown on the next slide.
- We group **A**, **B** and **C** together on the first grid line.
- On the second grid, we form two groups:
  - The first group consists of **D** and **E**.
  - The second group consists only of **F**.

4

ST BHANAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, we show an example in the next slide. So, let us go to that example we will come back.

(Refer Slide Time: 04:24)



So, you have an example like this the same example we took. So, these blocks are there you consider the middle grid D E F are the 3 ones which were connected an A B C are

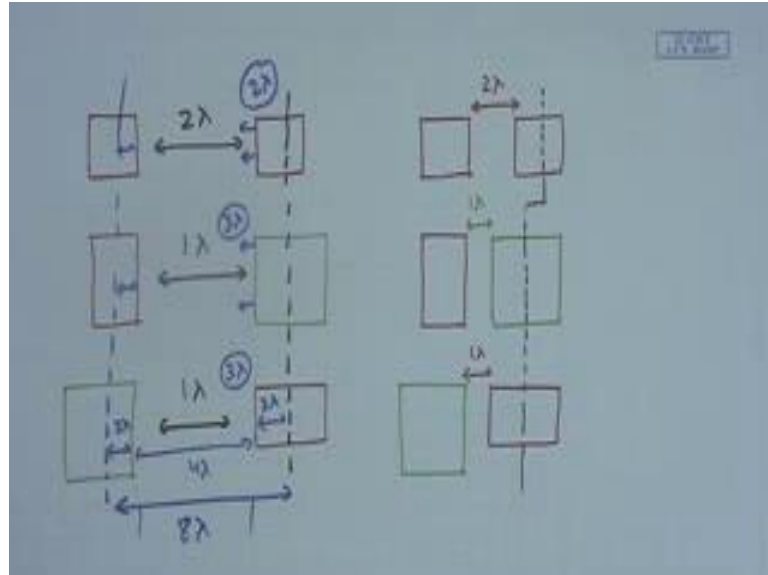
connected to the first grid. So, we are saying that we group A B C together on the first grid line there is no problem in that already A B C we are grouping together

But when you are grouping D E F we find because of some constraint D and E we can possibly bring closer together much more as compared to F. Let say here I am give an example, F I can bring together by 2 units of time, but D and E I can bring together further more than that. So, let say by maybe this B is also moving. So, this just an example illustrative example

So, this causes this grid line to be broken here. F is not moving, but D E have moved. So, what we were saying is that on the second grid D E F was there within that you have forming 2 groups. One containing D and E second group containing F. So, the basic idea is this. So, on each of the grid lines we carry out grouping then we move these blocks according to the scenario.

Let us take a very specific example with polysilicon and diffusion.

(Refer Slide Time: 06:05)



So, I am showing them using colors. Let say I have a polysilicon block here another polysilicon block here and I have a diffusion block here, so on the side on the right side let say I have a polysilicon here I have a diffusion here and I have let say another polysilicon here something like this.

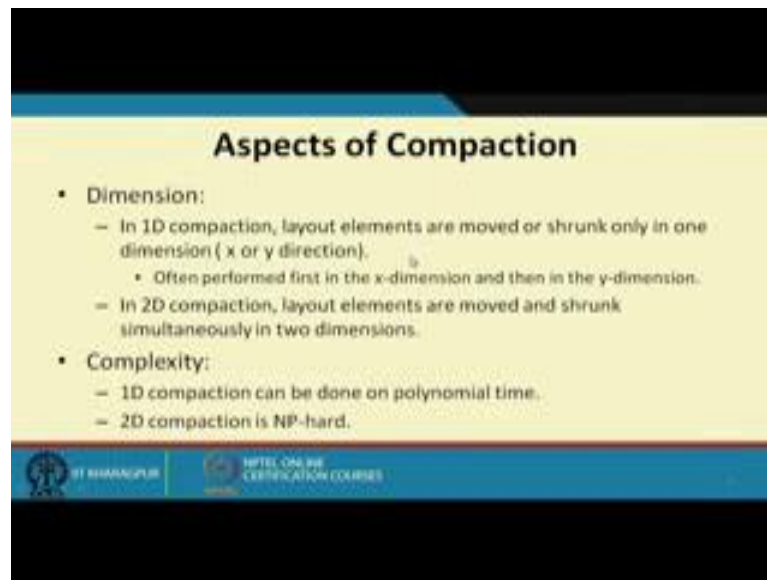
Now, if you recall this here we have only talking about the separation constraints right. Between 2 polysilicon the minimum separation has to be twice lambda, between a polysilicon and a diffusion separation can be 1 lambda between polysilicon and diffusion again it can be 1 lambda. Now when you are attaching a grid to these 3 let say, what you can do, but in the actual case let say this is the minimum constraint, but what you find in this layout that with respect to these 2 grid lines let say this is a grid and this is the other grid the separation between the 2 grid line is let say 8 lambda and this path is also something, let say these are all 2 lambda 2 lambda each 2 lambda each 2 lambda 2 lambda.

Similarly, here these parts are all let say 2 lambda each. So, taking out this 2 lambda we actually have an effective separation of 6 lambda right. Now see now 4 lambda, but we have these constraints. So, we have a separation between these 2 edges of 4 lambda, but here the requirement is 2 lambda. So, I can bring this block to the left by an amount 2 lambda. This 2 again is 4 lambda, but requirement is 1 lambda. So, I can bring this block to the left by 3 lambda.

This one again similar requirement is 1 lambda, I can bring it to the left by again 3 lambda. So, my new layout of the compaction will look something like this. These are the blocks connected to the first grid. Now this one you can move to the left by 2 lambda. So, now, the distance can be only 2 lambda. This gap will be 2 lambda. And this you can move by 3 lambda. So, the distance will be only 1 lambda.

The third one also here also you can bring it closer together to 1 lambda. Now with respect to the grid line, the grid line now becomes like this. These 2 are moving together because we have shifted both of them equally, but this was shifted means 1 lambda less. So, there will be a break in the grid line here. So, this is how it is done. Actually I am showing a specific example here the objects are all rectangles or some contacts the basic elements of the layout. And they are all specified by some design rules in typically in multiples of lambda. So, this method works like this.

(Refer Slide Time: 10:20)



**Aspects of Compaction**

- **Dimension:**
  - In 1D compaction, layout elements are moved or shrunk only in one dimension (x or y direction).
    - Often performed first in the x-dimension and then in the y-dimension.
  - In 2D compaction, layout elements are moved and shrunk simultaneously in two dimensions.
- **Complexity:**
  - 1D compaction can be done on polynomial time.
  - 2D compaction is NP-hard.

IT MANAGER | INTEL ONLINE CERTIFICATION COURSES

Now, let us look at the other approach where directly you are doing compaction in the different dimensions either one or 2 dimensions. Now the basic idea is that when we say aspects of computing of compaction dimension is one thing and of course, the time complexity of the algorithm is another thing. So, when you look at dimension the simplest can be possibly 1 dimensional compaction where we try to compact the layouts in only 1 dimension.

The idea is very simple. Suppose I have a layout. I have my left edge. I left to right I make a scan. Whatever objects are encountered I find or check whether they can be shifted left any further or not. If they can be shifted left, I shift it. And that way I scan the entire layout from left to right. So, so whatever objects or blocks are encountered I shift left by an amount which is possible without violating the design rule constraints and this is called 1 dimensional compaction, because I am compacting only in 1 direction, but in a general layout after you do it in the x direction you have to do another pass of 1 dimensional compaction in the y direction similarly. You may possibly be moving everything upwards right fine.

So, in 1d compaction typically you do the compaction first in the one dimension let say x, and then you do in the other dimension, but in doing that well you can always come up with examples where you could have done better, if you had done compaction in the both dimensions together. This is the so called 2d compaction. So, if you move the elements

simultaneously in some way in both the dimensions this is called 2d compaction which in general can result in better layouts better compaction.

But the advantage is that one dimensional compaction is easy it can be done in polynomial time, but 2 dimensional compactions you have to look at all the possibilities and find out the best move. So, the best 2d compaction is known to be computationally difficult. It is called np hard. So, you really cannot go for general 2d compaction algorithm which is the best possible. So, let us look at some examples here.

(Refer Slide Time: 13:12)

**1D Compaction: X followed by Y**

- Each square is  $2\lambda \times 2\lambda$ , minimum separation is  $1\lambda$ .
- Initially, layout area is  $11\lambda \times 11\lambda$ .
- After compacting along the x-direction, and then along the y-direction, layout area reduces to  $8\lambda \times 11\lambda$ .

So, here we illustrate 1d compaction method. Where a first we compact along x direction then we compact along y direction. So, here we have a simple example of a layout where all these blocks A D C D E F whatever is shown. Each of them are assumed to be 2 lambda by 2 lambda. And minimum separation let us assume to be 1 lambda. This is our example.

So, let us see C B this is 2 this is 2 separations is 1 lambda. Then you have I and H 2, 2 separations 1 lambda. And between b and I there will be separation of 1 lambda. So, total 5 plus 1 plus 5 11. Similarly, in the y direction 2 1 and 2 5 here also 2 1 and 2 5 and the 1 in between 11, initial layout area is 11 by 11.

Now, let us try and do an x compaction. So, we move from the left side to the right side progressively according to the x coordinates of all the blocks let see. So, C D E are

already to the left nothing to do. Then you encounter B and F they are already at a separation of 1. So, you cannot move them any further. A, A is quite further apart. So, the distance was 4 I think, 2 plus 1 plus 1 4. So, A you can shift to the left. So, that this separation becomes 1.

Similarly, I and H both you can shift to the left, till this separation becomes 1. G here the separation was 4 this also you move to the left. So, separation is 1. So, after 1d compaction your height is not reducing it is still 11, but width has reduced 2 plus 1 plus 2 plus 1 plus 2; that mean, 8. So, now, your area is 8 lambda, but you know; that means, 8 is in terms of the number of columns and 11 in this direction.

Now, you do y compaction; that means, you are pushing down let say. So, E F G are already down. So, I H are already having a gap of 1 you cannot do this. D is already having a gap of 1 you cannot push D further, but A you can move down. A, you can move down up to a point where there is a gap of 1. B also you can move down because B and A are together. So, because is connected to C and C is connected to A. So, that you have this connection also you cannot move them any further and C is having D below. So, you cannot move it further. So, you cannot reduce the height which is 8 lambda by 11 lambda. This is x followed by first to the x compaction. Then you do y compaction.

(Refer Slide Time: 16:40)

**1D Compaction: Y followed by X**

- Each square is  $2\lambda \times 2\lambda$ , minimum separation is  $1\lambda$ .
- Initially, layout area is  $11\lambda \times 11\lambda$ .
- After compacting along the y-direction, and then along the x-direction, layout area reduces to  $11\lambda \times 8\lambda$ .

The diagram shows a grid of blocks labeled A through H. Block A is at the top left, B is below it, C is to the right of B, D is below C, E is below D, F is to the right of E, G is below F, and H is to the right of G. The blocks are arranged in a way that shows the result of 1D compaction.

NPTEL ONLINE CERTIFICATION COURSES

Let us try the other way round y compaction followed by x compaction. This was our original problem 11 by 11. First let us do y compaction. You see D can be moved here



because this place is empty D is moved here gap of 1. C can also be moved down. B can also move down up to here because C B must be connected. So, B cannot be moved any further. Similarly, A, you can move up to here H this is possible.

So, now your height becomes 8, 2 1 2 1 and 2. Now you try to do x compaction where B or C D is already there. B F is already there. I you cannot move. H J also you cannot move, but A you can move, but it still remains 11 it does not reduce. So, your layout area still is 11 lambda by 8 lambda it is not reducing because in the earlier case it was reducing like this in the x direction here it was reducing the y direction.

(Refer Slide Time: 17:56)

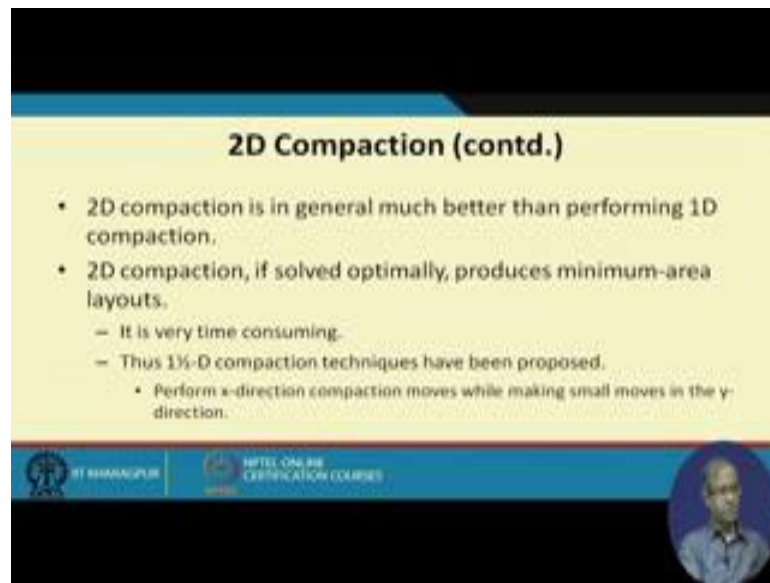
**2D Compaction**

- Each square is  $2\lambda \times 2\lambda$ , minimum separation is  $1\lambda$ .
- Initially, layout area is  $11\lambda \times 11\lambda$ .
- After 2D compaction, layout area reduces to  $8\lambda \times 8\lambda$ .

But now if you look at 2d compaction, see here we are taking an example which is quite similar right, quite similar, I H G now this was 11 by 11 again. Now if you do 2d compaction means I can move blocks simultaneously in x and y directions, like for example, this B block I am moving upward this I, I am bringing here this H I am bringing to the side and G is bringing here. So, now, nicely we are fitting it into a 3 by 3 kind of a layout. So, now, your total area will be 8 lambda this direction and 8 lambda this direction it is reduced.

But the problem is that given a problem, now you understand that the number of blocks are in the order of billions. So, it is not really feasible to find out the best way to move the blocks, so that you can get the best compaction like this. So, a number of heuristics has to be used. And that is what is done typically to achieve this.


(Refer Slide Time: 19:21)



**2D Compaction (contd.)**

- 2D compaction is in general much better than performing 1D compaction.
- 2D compaction, if solved optimally, produces minimum-area layouts.
  - It is very time consuming.
  - Thus 1½-D compaction techniques have been proposed.
    - Perform x-direction compaction moves while making small moves in the y-direction.

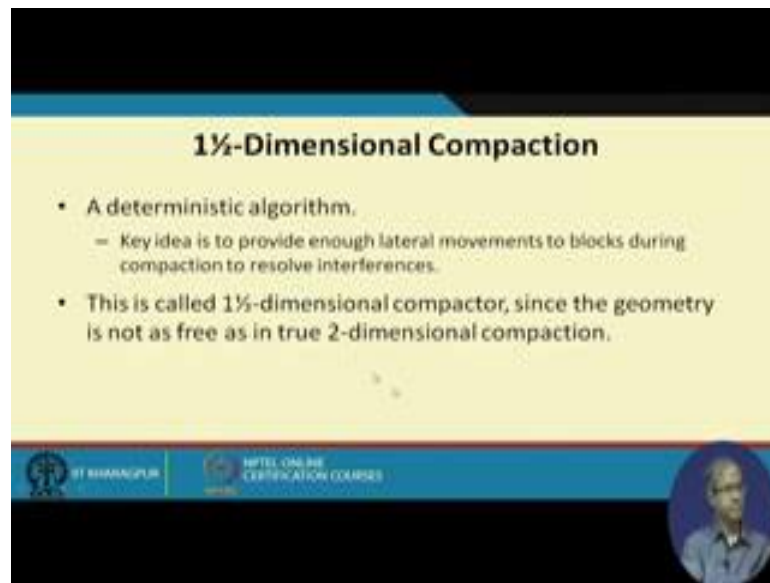
IT MANAGER | INTEL ONLINE CERTIFICATION COURSES



So, generally speaking as I had said 2d compaction gives better result as compared to 1d compaction. And if you can solve it optimally which unfortunately is not possible because it is NP-hard, it can produce minimum area layout, but as I had said because you have to make lot of checks and compares it is quite time consuming and 2d compaction per se is computationally not very feasible.

So, what people do? They do something in between. This is referred to as 1-and-a-half-dimension compaction. So, why it is called 1 and a half dimension. Because we are not giving equal importance to the x and y both the directions. We are primarily trying to compress in one direction, but in the other direction we are allowing only small moves this is the basic philosophy. That is it called 1 and a half d. So, let say we are performing x direction compaction, while making small moves in the y direction. This is the basic philosophy.


(Refer Slide Time: 20:45)



**1½-Dimensional Compaction**

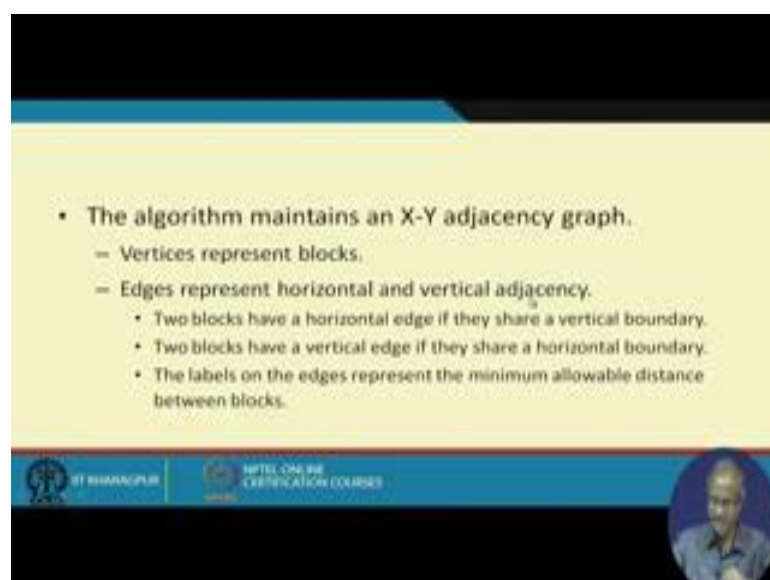
- A deterministic algorithm.
  - Key idea is to provide enough lateral movements to blocks during compaction to resolve interferences.
- This is called 1½-dimensional compactor, since the geometry is not as free as in true 2-dimensional compaction.

IT BHARANGPUR | NPTEL ONLINE CERTIFICATION COURSES




So, this is a deterministic algorithm not much checking is involved as we mentioned for the 2d algorithm. Now the idea is as follows that we are doing compaction, but during compaction we are also allowing lateral movement of the blocks. What does this mean? You see in a pure 1d compaction you had a block you just move it to the left, you have another block you move it to the left. You another block move it to the left, but now what we are saying when you select a block you are moving to the left all right, but in the process you are also allowing lateral movement you can also move a little up and then right if that gives you a better compaction, it is the idea.

(Refer Slide Time: 21:59)



- The algorithm maintains an X-Y adjacency graph.
  - Vertices represent blocks.
  - Edges represent horizontal and vertical adjacency.
    - Two blocks have a horizontal edge if they share a vertical boundary.
    - Two blocks have a vertical edge if they share a horizontal boundary.
    - The labels on the edges represent the minimum allowable distance between blocks.

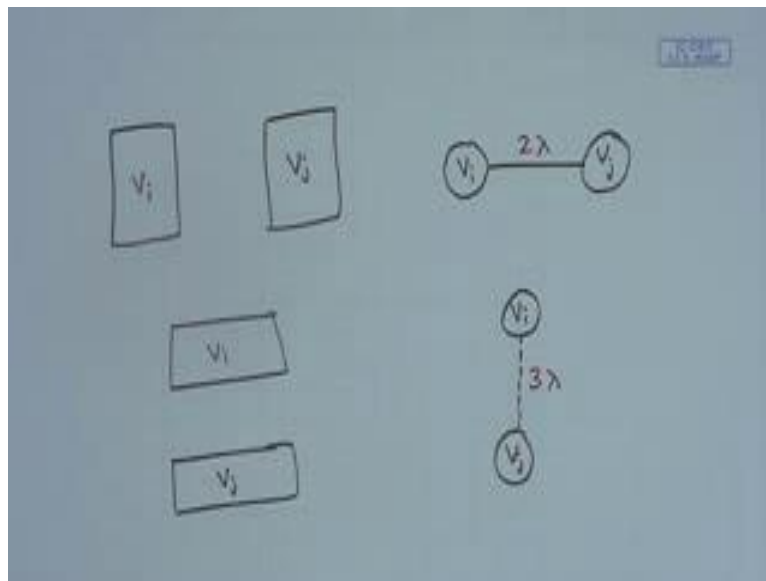
IT BHARANGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, this is called 1 and a half dimension as I had said because you are not so freely moving the blocks as in a true 2 dimensional compactions, but in a limited sense. You are allowing lateral movements in the other direction. Let us see how it works.

So, it uses something called a x y adjacency graph. Here the vertices represent the blocks in this graph, the edges represent horizontal and vertical adjacency, like let say if there are 2 blocks like this, which share a vertical boundary then in the graph. Let say this is  $v_i$  and this is  $v_j$ . Then in the graph there will be a horizontal edge between them.

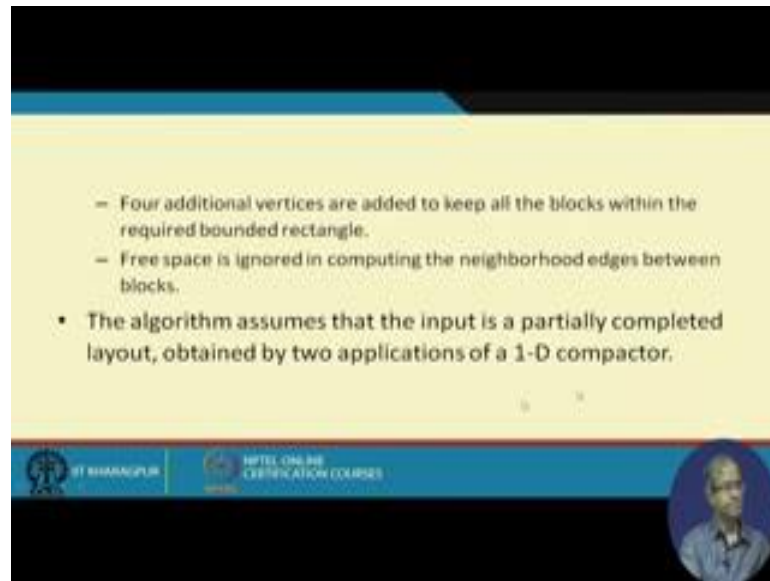
(Refer Slide Time: 22:24)



But suppose I have 2 vertices like this 2 blocks which correspond to vertices I mean let say  $v_i$   $v_j$  with a horizontal boundary between them. Then in the graph I connect them by a vertical edge which is showed differently let say draw it. So, there are 2 kinds of edges one is a so called horizontal edge other is a so called vertical edge. Depending on the direction where they are sharing their boundaries.

So, this is what I have just now mentioned, 2 blocks have a horizontal edge if they share a vertical boundary. They have a vertical edge if they share a horizontal boundary. And these edges are labeled representing the minimum allowable distance like for example, if the minimum separation between  $v_i$  and  $v_j$  is  $2\lambda$  this edge will be labeled as  $2\lambda$ . If this is  $3\lambda$  this edge will be labeled as  $3\lambda$ . So, we construct a graph and we label each of the edges by the minimum separation, minimum allowable distance which is given by the design rules.

(Refer Slide Time: 24:07)




– Four additional vertices are added to keep all the blocks within the required bounded rectangle.

– Free space is ignored in computing the neighborhood edges between blocks.

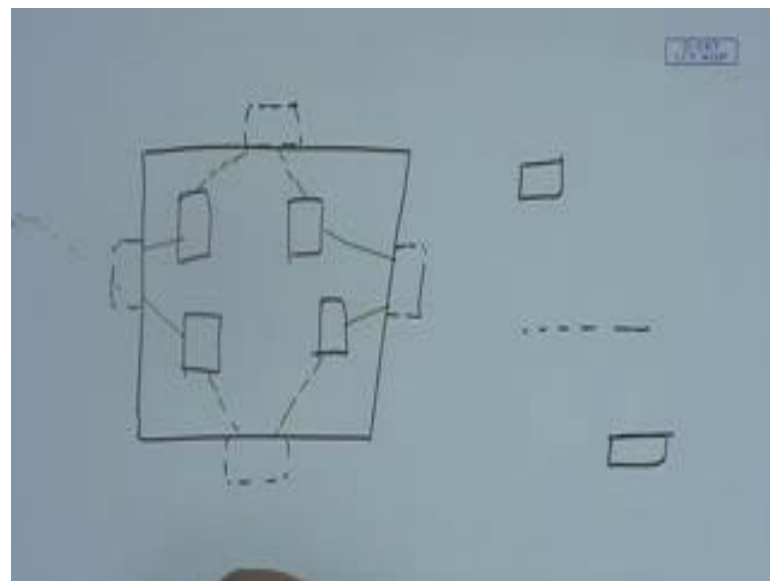
- The algorithm assumes that the input is a partially completed layout, obtained by two applications of a 1-D compactor.

IT BHARANGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, we are using 4 additional vertices. So, what kind of 4 additional vertices?

(Refer Slide Time: 24:17)

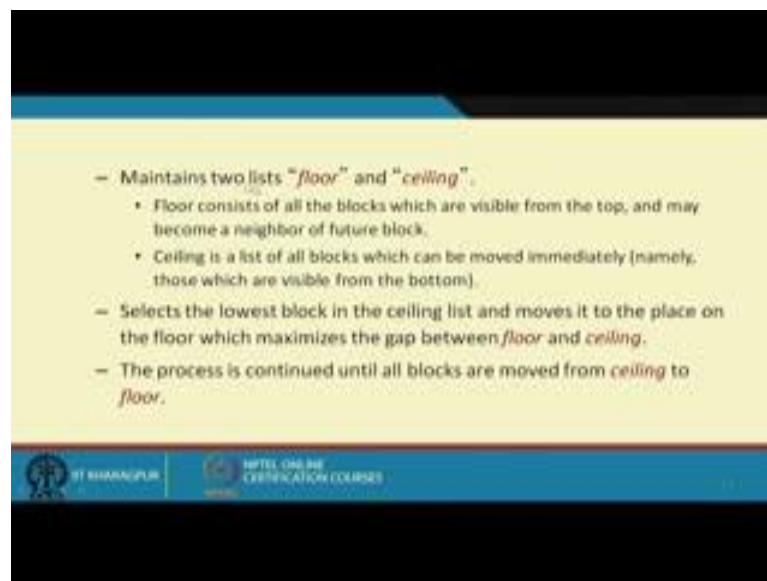


Suppose I have the layout area this is my total layout area, and all the blocks are in between. So, we have I shall show an example. So, I shall consider an imaginary block on the top an imaginary block on the bottom one on the right and one on the left. So, you see here I said blocks which share a vertical boundary they will be having a horizontal edge and reverse for this case. So, here which are the blocks which are having vertical

boundary? This imaginary left block and these, so there will be a there will be a horizontal edge marked by solid. Similarly, between these there will be vertical edge.

Similarly, if the blocks are sharing horizontal edge, like this and these 2 there will be vertical edge which will be marked by dotted edge something like this. Similarly, there will be edges in between the other also, but the 4 imaginary ones and the way they are connected will be something like this. So, while computing the edges or adding the edges we are ignoring the free space wherever there is, like what you mean by say is that suppose I have a block here, I have a block further down here. So, I assume that they are separated by a horizontal edge. So, there will be a vertical edge between them. So, we assume that the input is a partially completed layout. So, which is this may be obtained by some applications of 1d compactor then you are trying to do further optimization.

(Refer Slide Time: 26:17)



So, in this algorithm we are maintaining 2 lists floor and ceiling. Floor consists of all the blocks which are visible from the top, but you see you look at the 2 dimensional picture where blocks placed. Suppose you are viewing from the top, from the top some of the blocks will be hiding, some other blocks which are just below it, but some of the blocks you can see there are no obstacles. So, that ceiling that list will contain all those blocks which are visible.

So, ceiling is the blocks which can be moved immediately. They are visible means what there is a path in which you can move it towards the direction from where you are

viewing. Because there is a path you are able to see it. There is some space in between. Similarly, there is a list called floor it is from the other side, from the other side if you look, the blocks which you are able to see. So, floor consists of all the blocks which are visible from the top ceiling is a list of the blocks which are visible from the bottom, visible from the bottom means it can be moved immediately. Let see I will take an example how these are worked out.

So, the idea is as follows. You select the lowest block in the ceiling and move it to a place in the floor. So, from the ceiling you are trying to bring the blocks down towards the floor with some lateral movement where the distance or separation from the ceiling becomes maximum.

(Refer Slide Time: 28:33)

**Example**

- Since C is the lowest block in the ceiling list, it is selected for the move.

The slide contains a diagram of a room with blocks A, B, C, D, E, F, and G. Below the diagram, there are two lists: 'Ceiling' containing C, E, F and 'Floor' containing A, B, D, G. To the right of these lists is a graph with nodes representing blocks and edges representing possible moves between them.

So, you the select the lowest block in the ceiling move it to somewhere on the floor which maximizes the gap between floor and ceiling, and this process is continued because all the blocks are successively moved from the ceiling to the floor. So, I am showing one step of this algorithm. This is your initial problem. There are blocks A B C D E F G let us look at the ceiling and floor list first. Ceiling contains the blocks which can be seen from the top. You can see you can see G you can see D you can see C from this hole there is an empty space, we can see this block C E and F sorry I think means I come to ceiling means you are standing in a room and looking up the ceiling, you see which are the blocks you can see. You look up you can see G you can see D you can see

C you can see F, but you cannot see E ceiling will contain those. And floor means standing here you look down you can see A, you can see B middle there is nothing. So, this list is like this.

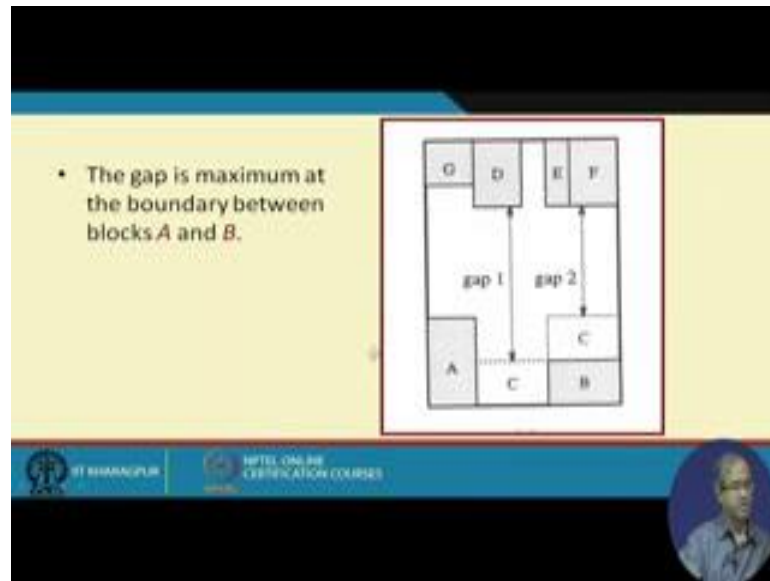
Now, let us look at the graph as I had said. So, there will be one vertex corresponding to A B C D E F G let us look at one by one. So, A and B are sharing a vertical edge they will be having a horizontal edge between them. So, as a convention I am using a dotted line there. You say you can use the other convention also no problem. Similarly, between G and D there is a vertical edge. D and E there is a vertical edge. E and F vertical boundary means horizontal edge. E and F there is one I think non-other, but you think of those imaginary vertices one is on the left  $x_1$  other is on the right  $x_2$ . So, from  $x_1$  there is a vertical edge with G there is a vertical edge with C and vertical edge with A. So, from  $x_1$  you add these edges.

Similarly, on the right with  $x_2$  you add F C B F C B similarly from top and the other way round sharing horizontal edges G A between G A there is an edge then C D C D C E D F and B C and similarly from the top there is an imaginary vertex  $y_2$  at the bottom there is  $y_1$  from bottom you can see the edges A and B from top you can see the ceiling G D C and F G D E and F I G D E and F these 4 you can see. So, this is your graph. So, each of the edges will be labeled with the minimum separation between the corresponding pair of blocks. So, so with respect to the imaginary one the edge weight can be 0.

Now, here C is the block on the ceiling which is lowest. So, you select C from the move. So, when you move it down there are.



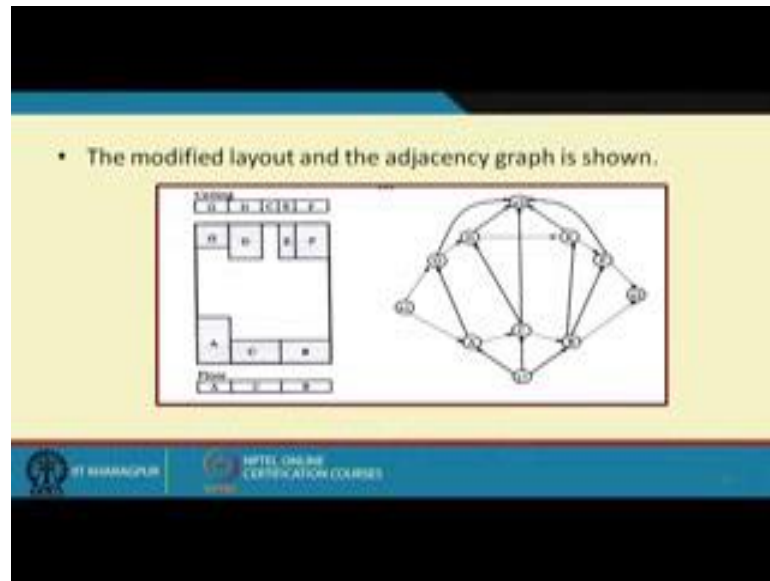
(Refer Slide Time: 32:05)



So, many places you can put it you can put it on top of A, you can put it on top of B you can put it in the middle. So, if you put it in top of B for example, the gap between the floor and ceiling will be this gap 2, but if you put it in the middle the gap here is increasing. So, you put it in a location which will have the maximum gap. So, this process will be repeated and this is how we get the so called one dimension or maybe one and a half dimensional compaction algorithm.

So, this works pretty. Well a pure 2d compaction becomes too complex to implement this computation is very difficult complex, but one and half dimension is not computationally that much intensive and with some lateral movement, when you are moving the block from ceiling to the floor you find what where to put it such that the separation between ceiling and floor the gap becomes maximum. That is the only heuristic you are using.

(Refer Slide Time: 33:01)



So, after you move it to C your graph immediately changes. So, you also keep changing the graph and also ceiling and floor list now floor also contains C because C can be seen in the floor also right this graph also gets modified. So, here I have shown for A 1. So, one by one these blocks will come down then the D possibly D will be moved down then E then F then G. So, one by one we you move it to the best location which maximizes the gap, and that will be your final compacted layout.

So, with that we come to the end of this lecture. And this finishes our discussion on layout compaction.

Thank you.