INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

NPTEL

NPTEL ONLINE CERTIFICATION COURSE
An Initiative of MHRD

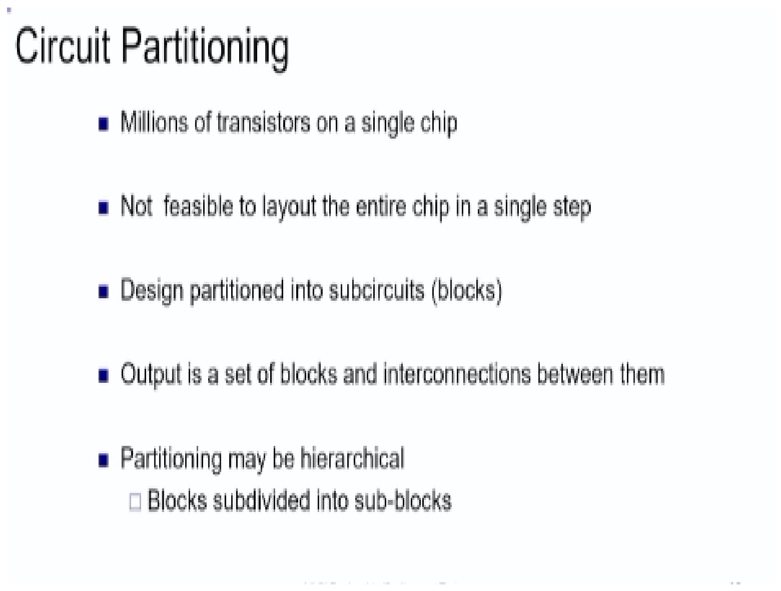
VLI Design, Verification & Test

Dr. Arnab Sarkar
Department of CSE
IIT Guwahati

Physical Design Automation

welcome the physical design process is divided into many sub steps the first sub step among them is circuit partition.

(Refer Slide Time: 00:36)



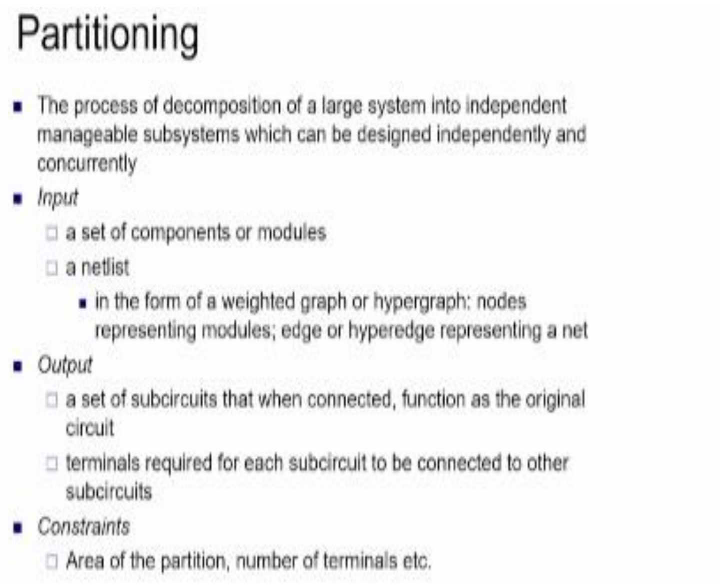
Circuit Partitioning

- Millions of transistors on a single chip
- Not feasible to layout the entire chip in a single step
- Design partitioned into subcircuits (blocks)
- Output is a set of blocks and interconnections between them
- Partitioning may be hierarchical
  - Blocks subdivided into sub-blocks

So we said that at the end of the logic design step we will have a circuit design consisting of millions of cells macros gates transistors and that interconnection now we have to place this entire circuit design into a single chip so we have millions of transistors on a single chip and it is not feasible to lay out the entire chip in a single step. Hence to control the complexity of the design process we partition the circuit into sub circuits are blocks the output is a set of blocks and interconnections among them.

So the output of the partitioning process is a set of blocks and interconnections among them partitioning can be hierarchical blocks can be divided into sub blocks okay.

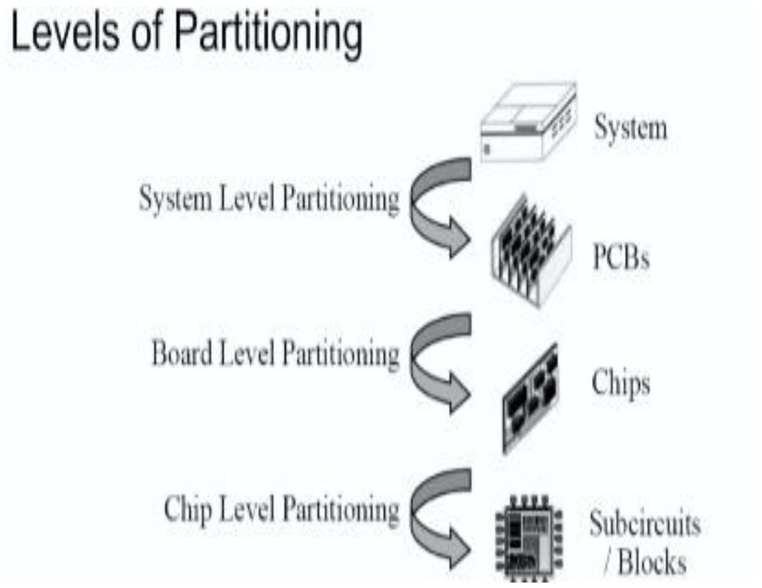
(Refer Slide Time: 01:30)



Partitioning

- The process of decomposition of a large system into independent manageable subsystems which can be designed independently and concurrently
- *Input*
  - a set of components or modules
  - a netlist
    - in the form of a weighted graph or hypergraph: nodes representing modules; edge or hyperedge representing a net
- *Output*
  - a set of subcircuits that when connected, function as the original circuit
  - terminals required for each subcircuit to be connected to other subcircuits
- *Constraints*
  - Area of the partition, number of terminals etc.

(Refer Slide Time: 01:32)



So partitioning is the process of decomposition of a large system into independent manageable subsystems which can be designed independently and concurrently. The input to the partitioning process is a set of components or modules these modules are the cells macros transistors gates that we have been telling and a net list in the form of a weighted graph or hyper graph nodes represent modules and edges or hyper edges represent a net or an interconnection and what is the output?

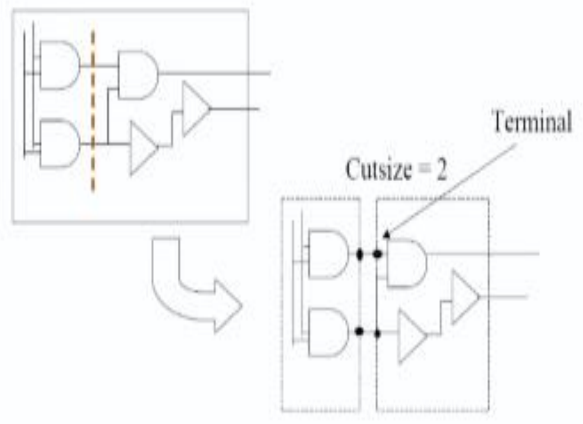
The output is a set of sub circuits that when connected function as the original circuit. So after partitioning the function must be preserved the function that was performed by the original circuit design should be preserved even after partitioning. The terminals required for each sub circuit to be connected to other sub circuits also needs to be defined so now when we have partitioned and defined the blocks we know how many wires will cross the boundary of the partition and we will go to blocks in other partitions.

Now therefore these wires that cross the partition and go to other partitions can only go through to the other partitions through terminals in this partition so therefore at the output of the partitioning process we also know the terminals of each partition. The partitioning process will be carried out under certain constraints for example there can be a minimum and maximum bound on the area of each partition on the number of terminals extra.

So now we come to levels of partition first we have the system the entire circuit diagram will consist of the entire system it will be divided into a set of PCBs in the system level partitioning where each PCB will consist of similar functions. These PCBs again will be divided into a set of chips at the board level partitioning so more similar functions will be partitioned into a single chip so therefore a set of chips will compose a PCB and then a chip is divided into sub circuits or blocks and that is called chip level partitioning.

(Refer Slide Time: 04:03)

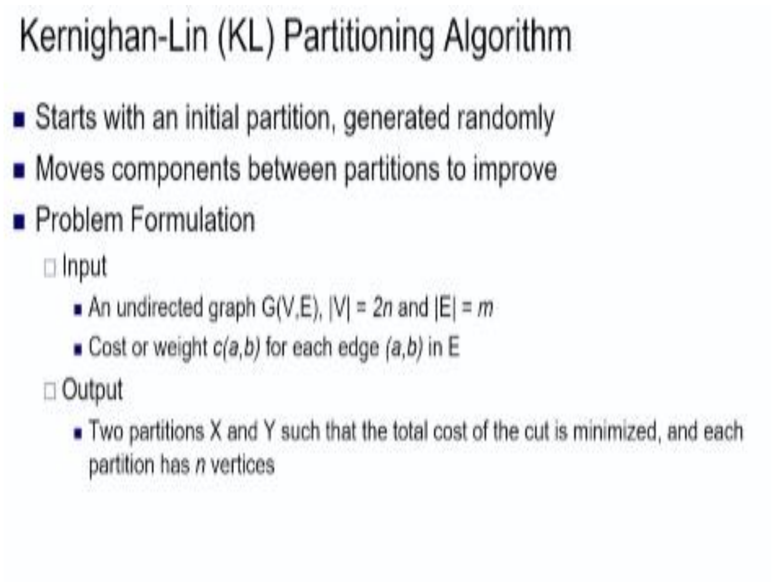
Partitioning Example



This is a representation of a hypothetical circuit design so here are the modules of the circuit here are the interconnections or wires and we said that we have cut the circuit into two partitions so this is partition one and this is partitioned two. Partition shown one consists of these two modules and partition to consist of these three modules and there are wires that are crossing the partition right and this line will be called a cut right and here we see that there are two lines that cut so these are the two partitions that we have after the partitioning process these will form two separate blocks.

Block one and block two and the size of the partition one of the important parameters by which a partition is characterized is the size of the cut so there are two wires that cut through the partition and therefore the cut size is to hear.

(Refer Slide Time: 05:13)



Kernighan-Lin (KL) Partitioning Algorithm

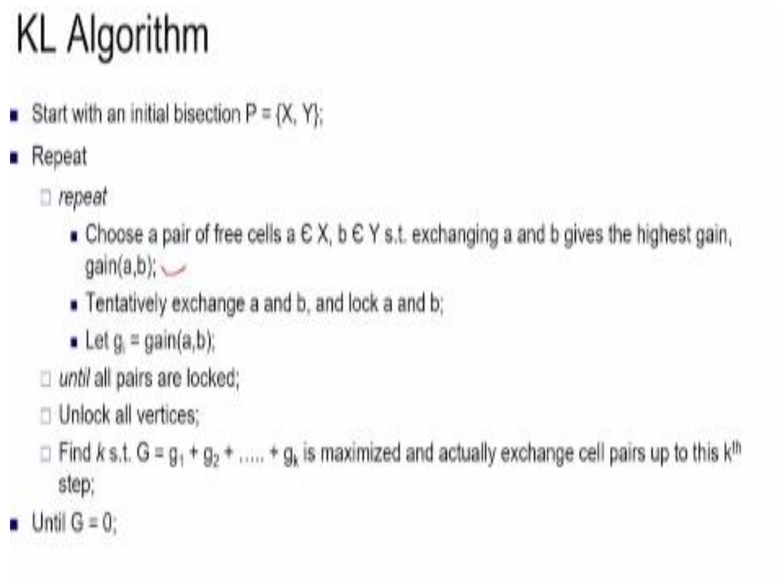
- Starts with an initial partition, generated randomly
- Moves components between partitions to improve
- Problem Formulation
  - Input
    - An undirected graph G(V,E), |V| = 2*n* and |E| = *m*
    - Cost or weight *c(a,b)* for each edge *(a,b)* in E
  - Output
    - Two partitions X and Y such that the total cost of the cut is minimized, and each partition has *n* vertices

So an important partitioning algorithm is the Carnahan and lynn algorithm so how does the Carnahan and lynn algorithm work it starts with an initial partition generated randomly and it moves components between partitions to improve so it is a group migration based algorithm and it is an iterative improvement based algorithm. Ititeratively improves over phases the Carnehan and lynn algorithm solves the following problem.

The input consists of an undirected graph the net list consisting of 2n vertices therefore we have two and circuit modules and M edges between them so there are M interconnections or wires connecting the circuit modules in this graph or net list and we also know the cost of each edge the cost C A B of each edge A B is known to me this is the input to the problem and the output what is what do we desire out of the cardigan and in algorithm we want to obtain two partitions x and y such that the total cost of the partition is minimized and each partition should have in vertices.

So we want to obtain two equal-sized partition of size n from the entire circuit consisting of 2n circuit modulest.

(Refer Slide Time: 06:47)



KL Algorithm

```
▪ Start with an initial bisection P = {X, Y};
▪ Repeat
    ▫ repeat
        ▪ Choose a pair of free cells a ∈ X, b ∈ Y s.t. exchanging a and b gives the highest gain,
          gain(a,b);
        ▪ Tentatively exchange a and b, and lock a and b;
        ▪ Let $g_i$ = gain(a,b);
    ▫ until all pairs are locked;
    ▫ Unlock all vertices;
    ▫ Find k s.t. $G = g_1 + g_2 + \dots + g_k$ is maximized and actually exchange cell pairs up to this kth
      step;
▪ Until G = 0;
```

The cardigan and lynn algorithm starts with an initial by section p equals to X , Y. So X is one partition and Y is the other initial partition so initially we will randomly choose n vertices and put in X and other n vertices and put in Y and get an initial partition and that initial partition will have a cost in terms of the number of wires that cross the partition that will be called the cut size. Then we proceed in steps so first we choose a fair a pair of free cells A belongs to X and B belongs to Y such that exchanging A and B gives the highest gain in terms of reduction in cut size.

So we choose two modules or vertices and one from X and the other from Y and exchange them how do we choose these two vertices we choose these two vertices such that these two vertices after exchange will give us the highest gain among all vertices in X and Y in terms of reduction in cuts eyes after their exchange. So then we actually exchanged A and B tentatively we actually exchanged A and B tentatively and then lock them from further consideration in this loop.

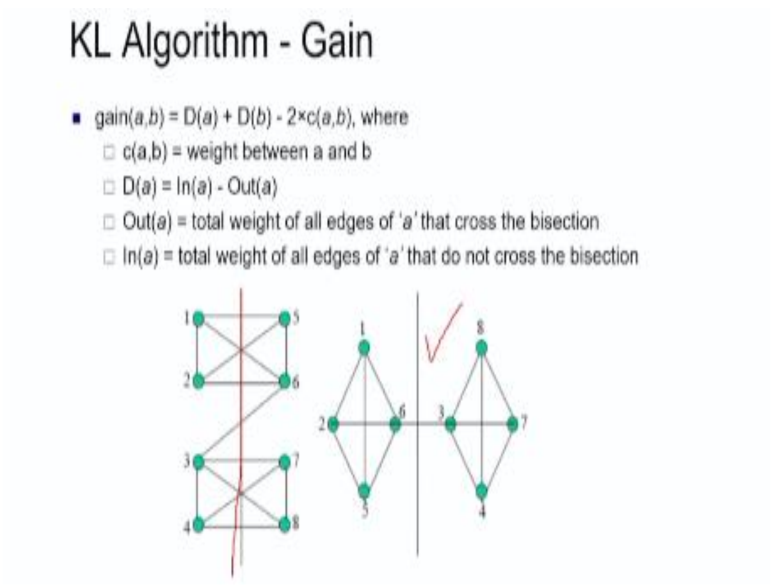
So after we have exchanged A and B and have locked them we cannot consider them for further exchanges in subsequent sub sequent iterations within this inner repeat loop we note the gain GI corresponding to this exchange of A and B and we do the same thing until all pairs are locked after a pass within the inner repeat loop we unlock all vertices and then we find K such that G equals to G 1 G 1plus G 2 + gk is maximized and actually exchange sell pairs up to this gate step.

so what do we do we find a cave we start from one the first exchange second exchange third exchange forth exchange we go on taking these exchanges until what until we get the highest gain. We have to remember that certain many of these gains will actually be losses that exchanging them did not provide me and decrease in cuts eyes it provided mean increase in cut size.

Therefore there will be a certain step k up to which we will have an effective and maximum increase in cuts size sorry effective and maximum decrease in cuts size and beyond that we will not do exchanges so in this pass we will do actual exchanges up to the KH step and then repeat this in the next loop.

In the next loop we will again find the a pair of vertices starting from those which provide me the highest gain and go on doing this until we have a pass when we do not have any gain at all.

(Refer Slide Time: 10:18)



KL Algorithm - Gain

- gain(*a*,*b*) = D(*a*) + D(*b*) - 2×c(*a*,*b*), where
  - c(a,b) = weight between a and b
  - D(*a*) = In(*a*) - Out(*a*)
  - Out(*a*) = total weight of all edges of '*a*' that cross the bisection
  - In(*a*) = total weight of all edges of '*a*' that do not cross the bisection

Firstly we need to understand how we choose vertex pairs and exchanges exchange them so we said that at any point in time we choose that vertex pair that gives me the highest gain so how do we define a gain? The gain is defined as DA+ DB - 2c AV where C AV is the weight between A and B. So it is the cost of the edge A, B so what can this cost of the edge A, B denote it can denote for example the number of wires that interconnect the two blocks A and B what is DA? DA is given by in A minus out A.

So what is in A? In A is the number of edges that are incident on A and do not cross the partition and what is out A? The number of edges that are incident on A and cross the partition ok so out there equals to total weight of all edges of a that cross the bisection in A total weight of all edges of it that do not cross the bisection.

So now let us see the values of DA for this note say here five so we see that it has an in A in a 5 is 1 and out of 5 is 2 so DA which is D of 5 will be 1 minus 2 which is minus 1 let us take another note 3 here for example 3 has an inage of 1 and there are 3 outage. So therefore D of 3 is 1 minus 3 which is minus 2 now let us say this pair gives me the highest gain among all pairs that we can consider right this pair gives me the highest gain and then we exchange these pairs.

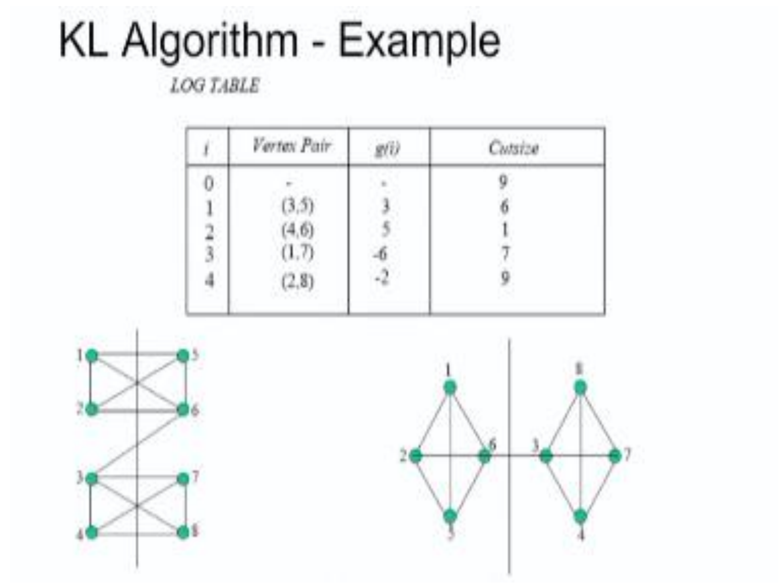
So what happens whatever was an inage becomes an outage and whatever was an outage becomes an inage now so 5 goes this side 5 comes this side now what happens it's outage will be 1 and its inage will be 2 so the changed DA value will now be +1 now when 3 comes to this side it's changed inage will be 3 and it's outage will be 1 so inage minus outage will be + 2 so the total gain will be 1 + 2 which is equal to 3.

If we consider another vertex pair say 6 and 3 before it is exchanged 3 had an inage minus as outage DA value of - 2 similarly 6 also had a DA value of - 2 it has 3 outages and 1 inages so inages- outages is -2. So now what happens when 6 comes this side and 3 comes this side on this side 6 DA value of changes to + 2 and 3 DA value also changes +2 so 2+2 the total gain is 4 . However this edge that connects 6 entry we need to decrease the cost of this of this edge.

So the gain has to be decreased by the cost of this edge and that is also twice one for 3 and once for 6. 6-4 6-3 was an outage for six when I took it to this in the DA value we considered that 6 3 has become an inage but because three has gone on the other side 6 3 is still an outage for 3 also 3 6 was an outage and after they exchanged instead of becoming an inage 3 6 still remains an outage because 6 has crossed over to the other side.

And then therefore we need to subtract the cost of CA be twice to get the actual gain so we have chosen two vertices A and B we want to find out what is the gain after exchanging them so how do we find that out we find the gain after exchanging a we find the game after exchanging be and if a and B these vertices vertex pairs are connected through an edge we need to deduct the cost of the edge from the total gain twice this is what this gain A, B says right

(Refer Slide Time: 15:25)



KL Algorithm - Example

LOG TABLE

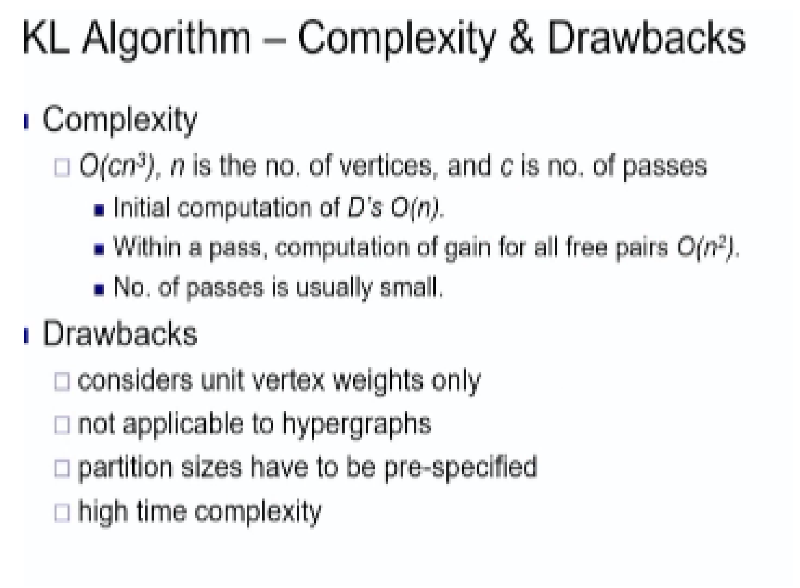
i	*Vertex Pair*	*g(i)*	*Cutsize*
0	-	-	9
1	(3,5)	3	6
2	(4,6)	5	1
3	(1,7)	-6	7
4	(2,8)	-2	9

So now we see the first step in this loop so what are we seeing we are seeing this one is set of steps this inner look we will see so first is to choose first we see that the cut size is 9 the initial cut size is 9 so first we choose a pair with the highest gain firstly we have to keep the vertex gains for each vertex pair in a separate table 3 6 gives the highest gain say so we saw that the gain by exchanging 3 6 is 3 and therefore the cut size reduces by 3 and the new cuts size become 6 after that the highest gain is provided by the vertex pair 4 6.

So we choose 4 6 exchange them and see that the gain produced now is fine so previously we see that the gain the maximum gain was three but after 3 and 6, 3 and 5 has been exchanged the new gain that is provided now is 6 4 and the gain is 5 which is higher than the first gain right and then the cut size reduces to 1 from 6. So 6 - 5 is 1. Now the next to vertex pair the gain is negative it is -6 and -2 however we have to continue doing this until all vertex pairs are exchanged in the inner loop.

And then we said we only take up to that gain up to that value of K which provides me the highest gain here we see that the value of K is 2 so after these two steps we do not get the highest game is provided only after these two steps and hence we only take up to these two steps right and in fact for this example we need only one pass in the second pass we will see that none of the exchanges gives me an again which is higher than 1 and the lowest cut size that is possible is 1 and in the second iteration G will remain equal to 0 and the algorithm will stop.

(Refer Slide Time: 17:57)



KL Algorithm – Complexity & Drawbacks

- Complexity
  - $O(cn^3)$, n is the no. of vertices, and c is no. of passes
    - Initial computation of D's $O(n)$.
    - Within a pass, computation of gain for all free pairs $O(n^2)$.
    - No. of passes is usually small.
- Drawbacks
  - considers unit vertex weights only
  - not applicable to hypergraphs
  - partition sizes have to be pre-specified
  - high time complexity

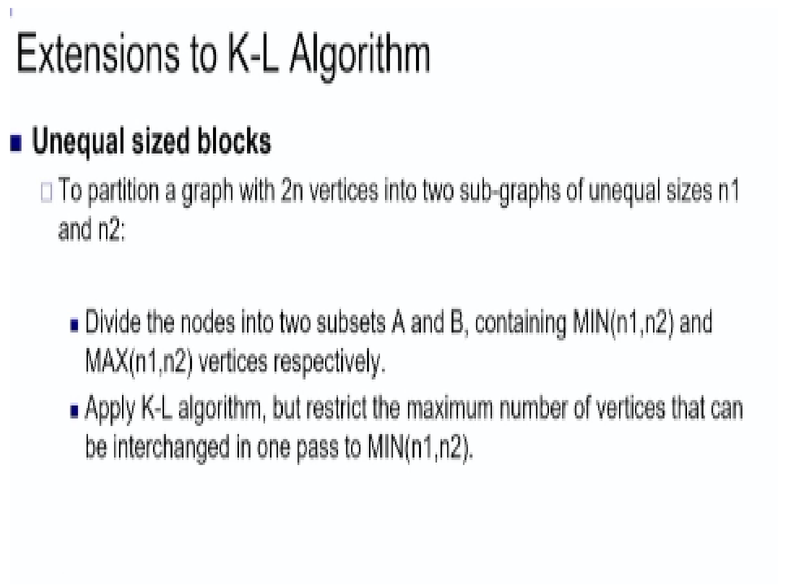
What is the complexity of this algorithm the complexity of this algorithm is big-oh of CN q where n is the number of vertices and c is the number of passes ok so why is it n cube because initially we need to separately calculate the D values of each node and that will take Big O of n type right so to calculate the gain value each calculation of each gain value for each node takes constant time for n nodes it takes Big O of n time now within each pass we need to compute the gain across all free pairs.

Now in each pass we have to once calculate the gain across all free pairs and then you have to go on updating the gains in each pair so what is the maximum that number of pairs that we have to choose its Big O of n square right we have n vertices on one side in one on one side of the partition n vertices on the other side of the partition and then we have to take one for text and find the gain with respect to all other vertices on the other side of the partition then we take another vertex on this side and find out the gain for all vertices on the other side likewise we will do for this step will be calculus ease over all vertices in a partition.

And hence the total step to the total computation that will be required is Big O of n square the number of passes is usually small what are the drawbacks of the corning unending algorithm it only considers unique vertex weights it is on it's not applicable to hyper graphs will not define hyper graphs here basically we can say that hyper graphs is delayed to the representation of nets that can be connected to multiple blocks more than two blocks.

So multi terminal Net scan be represented by hyper graphs but will not consider hyper graphs here partition sizes have to be pre specified we can only do a by partition of equal size and which is high time complexity big o of n cube.

(Refer Slide Time: 20:26)



Extensions to K-L Algorithm

- **Unequal sized blocks**
  - To partition a graph with 2n vertices into two sub-graphs of unequal sizes n1 and n2:
    - Divide the nodes into two subsets A and B, containing MIN(n1,n2) and MAX(n1,n2) vertices respectively.
    - Apply K-L algorithm, but restrict the maximum number of vertices that can be interchanged in one pass to MIN(n1,n2).

However of many of the algorithms in partitioning had extensions of the cunning are handling algorithm or use the kerning un ending algorithm in its various internal steps we will look at a few in extensions to the carving and in algorithm. First one is that how do we get unequal size partitions out of the Carnahan and Lynn algorithm to partition a graph with 2 n vertices into two sub graphs of unequal sizes in 1 and n 2.

So what do we do we divide the nodes into two sub sets A and B containing minimum of n1 and n2 and maximum of n 1 and n 2 vertices respectively in these two what is in these two partitions so initially during partitioning what we will do one of the partition say A will contain min of n1 and n2 and on the other partition we will keep the max of n1 and n2 nodes are not modules then we apply the kerning an Indian algorithm but restrict the maximum number of vertices that can be interchanged in one pass to a minimum ofn1 and n2 so now we need vertex pairs inn1 and n2 because the number of nodes in these two partitions are not same we can only exchange to a minimum of n1 and n2.

So what will happen to the other nodes they will be not considered in this path and they are remaining only because they're exchanged their exchange they form pairs with other vertices which produce very less gain only because of that they were not considered and they be considered in further subsequent iterations in this so at within each iteration we will only consider a minimum of n1, n2 vertex pairs.

And actually exchange them and consider them for exchanges the next extension to the cardigan and lynn algorithm is to generate to a partition of a graph whose vertices have unequal sizes so in this version of the turning under Lynn algorithm that we just studied we all of these vertices have the same size let us say unit size now to extend this to unequal sized vertices we assume well data let the smallest element has unit size and then we replace each element of size s with s vertices.

so s unit size vertices which are fully connected with edges of infinite wait so why do we do his so that we can feed the algorithm to the original cardigan and Lynn algorithm this will work but by representing it as s small unit size vertices what do we do we say we find we keep a track of what is the size of the partition on a given side right if you represented this a whole big vertex of size s as 1 then the partition size would not be correct the partition size on each sides on each side of the partition the area would not be correct.

Hence we represented it as an s sized click of where each edge of the click is of infinite weight why do we have infinite weight so that they will never be go into another partition and then we apply the kerning handling algorithm on the modified graph if you want to have cave a partitioning instead of two-way or by partitioning let us assume that the graph has came to n vertices k greater than 2 and it is required to generate a cave a partition each containing n elements.

So what do we do we begin with a random partition of K sets of n vertices each then apply the two-way partitioning procedure on each pair of partitions and on each pair of partition so what do we do we begin a random partition of case X of n vertices each and then apply to way partitioning procedure on each of these partitions if we have K partitions then there are K square pairs of these partitions k square distinct pairs of the separtition s so begin with a random partition of K sets of n vertices it and then apply to way partitioning procedure.

On each pair of partitions so how many distinct pairs of partitions do we have we have K square distinct pairs of partitions and therefore the time complexity of one pass over all these pairs is K square and cube and this will again proceed through a few number of passes so we understood three different extensions to the cardigan and in algorithm with this we come to the end of this module.

Dibyajyoti Lahkar

Kallal Barua

Kaushik Kr. Sarma

Queen Barman

Rekha Hazarika

CET Administrative Team

Susanta Sarma

Swapan Debnath