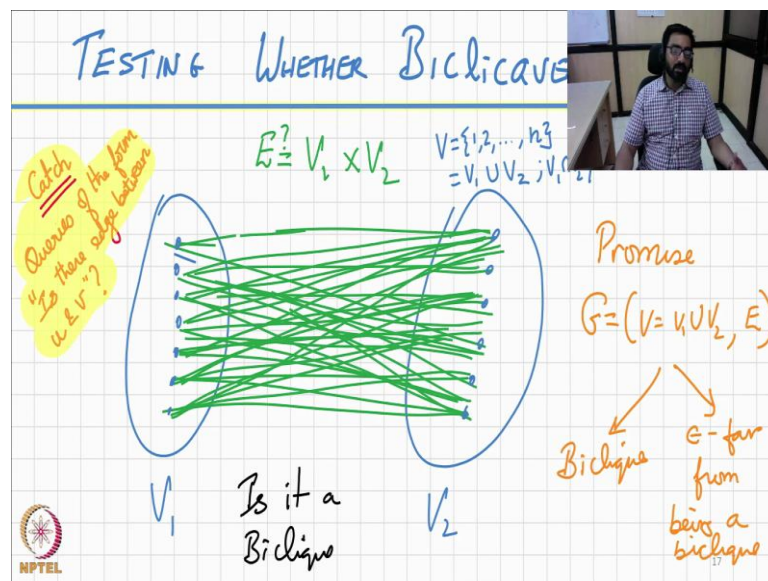


**Algorithms for Big Data**  
**Prof. John Ebenezer Augustine**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**Lecture – 40**  
**Testing if Graph is a Biclique**

In this segment, we are going to look at the problem of Testing if Graph is a Biclique and particular we going to assume a dense graph model in which the queries will of the form is there an edge between these 2 pair of a vertices and, in particular this is an excellent example, where the test and the enforce and test technique is exercised. So, without further adieu let us look at this problem.

(Refer Slide Time: 00:50)



Where we need to test whether the graph is a biclique and so, what is the biclique? Well, the vertex set  $v$  should be able to the partition it into subsist  $v_1$  and  $v_2$  such that of course, they do not intersect and of  $v_1$  and  $v_2$  is the entire vertex set and over its a biclique, If the edge set is exactly equal to  $v_1$  cross  $v_2$ . So, every such that one of them is in  $v_1$ , the other one is in  $v_2$  must have an edge between them and that is what a biclique is and of course, we are interested in testing whether the given graph is a biclique.

Of course, we have to assume that there is a promise and what is the promise here we are guaranteed that the graph is either a biclique, which means there exist such a partition  $v_1$  and  $v_2$ , but the edge is all possible edge is going across  $v_1$  and  $v_2$  or the graph is epsilon far from being a biclique, what is that even mean? Well, let us look at it shortly, but as we mentioned earlier we need to answer this question just by asking queries of a very specific nature this is the dense graph model and so we cannot assume the graph is stored in bit, the graph is stored in instead of list type of a format. It is more like a joint matrix and so we can ask the question is there an edge between vertex  $u$  and vertex  $v$  and using queries of that type, we should be able to answer this question whether the graph is biclique.

(Refer Slide Time: 02:59)

$\epsilon$ -FAR FROM BEING A BICLIQUE

- $\forall$  partitions  $V_1, V_2$

# of edges added / deleted to make  $V_1, V_2$  a biclique partition is  $> \epsilon n^2$

$\updownarrow$

Cardinality of symmetric diff:  $|(E \setminus (V_1 \times V_2)) \cup (V_1 \times V_2 \setminus E)| > \epsilon n^2$

Labels: # of vertices, edge set

So, what that mean for graph to be far from being a biclique? In particular, epsilon far from being a biclique, what is that mean for graph to not be a biclique. Well, any way in which you partition it the edge set is should not be equal to  $v_1$  cross  $v_2$  that is what, it means for graph to not be a biclique, but for it to be a far from being a biclique for all partitions  $v_1, v_2$  the number of edges that need to be added or deleted to make  $v_1$  and  $v_2$ , a biclique partition should be at least epsilon  $n$  square and if.

So, some of some time you may need to add a few adjust because these could be adjust in  $v_1$  cross  $v_2$ , there are missing or some furious adjust that are lying between 2 vertices in  $v_1$  or lying between 2 vertices on  $v_2$  need to be deleted and we need to make epsilon  $n$  square such modifications before this there we arrive at a partition  $v_1$  and  $v_2$ , such that this partition leads to a biclique is a biclique partition.

Another way of saying the same thing is that the cardinality of the following symmetric difference, in other word, in particular  $e$  minus  $v_1$  cross  $v_2$ , these are the edges that are in  $e$  that should not be in the biclique partition union  $v_1$  cross  $v_2$  minus  $e$ . These are the adjust that should be there, but that are missing, the union of these 2 differences is call the symmetric difference between the set  $e$  which actually exist in the graph and the set  $v_1$  cross  $v_2$ , but should be 4 of the biclique that is symmetric difference has to have a cardinality at least epsilon  $n$  square.

So, this is what they mean for a graph to be epsilon far from being a biclique. Llet us remind ourselves, we have a graph we know there is either a biclique or its epsilon far from being a biclique and we need to test and the find out whether which character we belongs to and we want to arrive at an algorithm than that exercising this enforcing test technique.

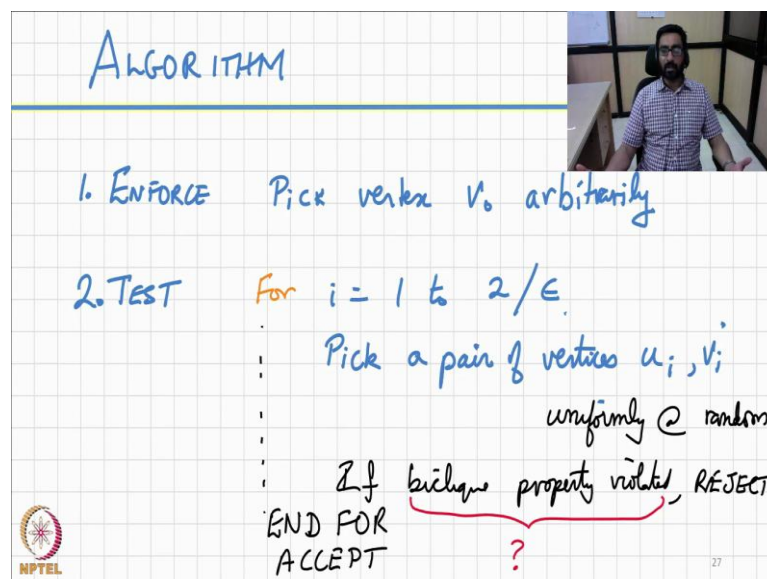
(Refer Slide Time: 05:55)

The slide is titled "ALGORITHM" in blue. It lists two steps: "1. ENFORCE" and "2. TEST". Under "1. ENFORCE", it says "Pick vertex  $v_0$  arbitrarily" with a green bracket. A green arrow points from this text to a diagram. The diagram shows a vertex  $v_0$  in a black oval labeled  $V_1$  (Non Neighbors). Red lines connect  $v_0$  to a red oval labeled  $V_2$  (Neighbors). A pink note says "Huh? That's it?". A yellow box contains the text "PARTITION  $V_1, V_2$  ENFORCED!". The NPTEL logo is in the bottom left corner, and the number 24 is in the bottom right corner.

So, here is how the algorithm works there is enforce part and in this going to be the test part and believe it or not the enforce part for this algorithm is simply just 1 step, pick a vertex  $v_0$ . Arbitrarily, it could be any vertex and you may say well is that it and as it turns out yes, but look at the power of this 1 single step. We as soon as pick this vertex  $v_0$  this immediately enforces a structure.

Now, if the graph is in fact, a biclique that is without a loss of generality assume  $v_0$  is in  $V_1$  all the neighbors of  $v_0$  are have to be in  $V_2$  and all the non neighbors of  $v_0$  must be in  $V_1$  and so, the entire structure of the graph is enforced and this makes it easy for us to test whether there is a violation of this enforced structures if the graphs. In fact, a biclique this is the way the structure has to exist and all neighbors of  $v_0$  must be in the set  $V_2$  and all non neighbors of  $v_0$  must be in the set  $V_1$ . So, let us go ahead and test if this is in fact, holding true.

(Refer Slide Time: 07:25)



**ALGORITHM**

---

1. ENFORCE Pick vertex  $v_0$  arbitrarily

2. TEST For  $i = 1$  to  $2/\epsilon$ .

Pick a pair of vertices  $u_i, v_i$   
uniformly @ random

If biclique property violated, REJECT

END FOR  
ACCEPT

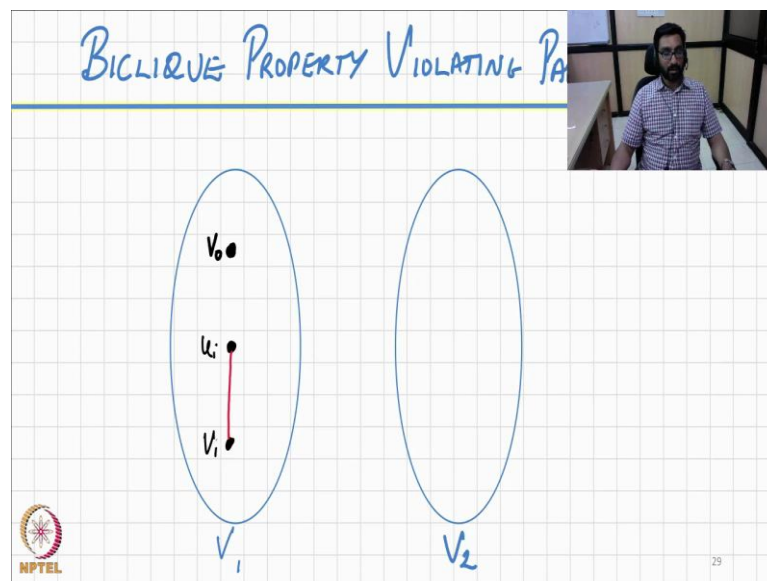
?

NPTEL 27

So, how do we do that? Here is how we do it we do this part again very simple we simply sample some  $\theta$  of  $1/\epsilon$  number of pairs of vertices. So, we simply want to  $i$  equal to 1 to  $2/\epsilon$  and each iteration of 4 loop is comprises picking of pair of vertex  $u_i, v_i$  and we are choosing uniformly at random and now we test if the biclique property is violated.

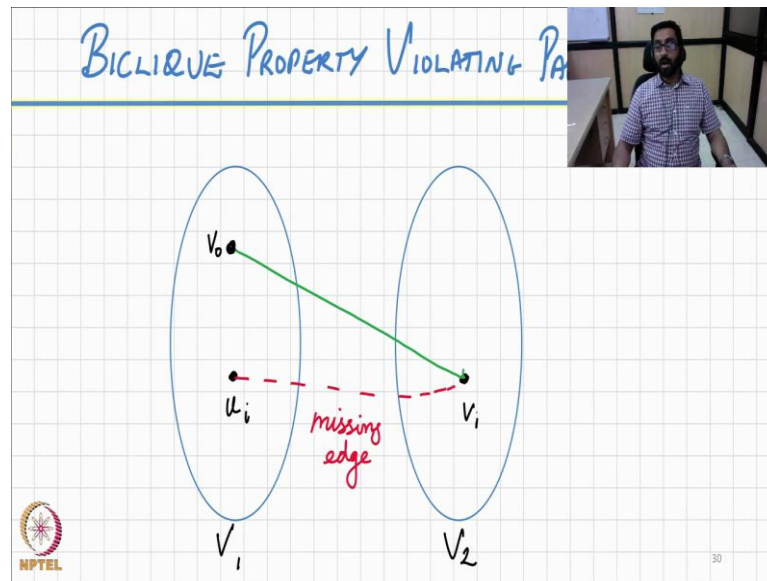
If it violated we immediately reject the instance, but if it is not violated we continue on and after testing to our epsilon of these pairs if we still have not found any such violating pair then we terminate the algorithm and accept the instance as 1 that is indeed a biclique and of course, now we need to ask us also what is it even mean to be able to detect a violation well let us look at this a bit carefully.

(Refer Slide Time: 08:43)



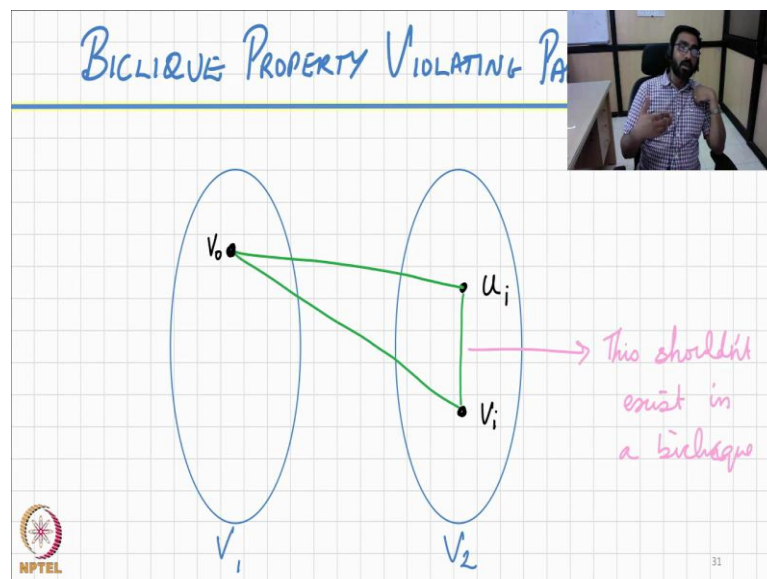
Recall that, we have pick vertex  $v_0$  and that meant that neighbors of  $v_0$ , went into  $v_2$  and non neighbors of  $v_0$  went into  $v_1$ . What are the ways in which we can encounter violating pairs of we can have edges of the form shown here where both  $u_i$  and  $v_i$  the sample pair both are non neighbors of  $v_0$ .

(Refer Slide Time: 09:16)



Or you can have  $u_i$  which is a non neighbor and  $v_i$  which is a neighbor, but this means that they must be an edge between the 2 of them, and if that edge is missing then that is another violation.

(Refer Slide Time: 09:30)



Here is the third type of violation,  $u_i$  and  $v_i$  are both neighbors of  $v_0$  and therefore, in  $v_2$ , but the end of having the edge an edge between them and that edge is also should not exist. So, there are three types of violation that you can encounter the first and the last type of violation we saw are extra edges that need to be deleted in order to make this this structure actually a biclique and the second one is missing.

Actually, this edge must be added in order to convert this into a biclique, remember that there are  $\epsilon n$  such modification if the graph is indeed not a biclique there is  $\epsilon n$  such modification are needed before the graph of it can come a biclique, so that that means, that our samples I mean this gives us the possibility of rather the high, if the good probability of capturing such violation.

(Refer Slide Time: 10:39)

ANALYSIS

If  $G$  is a biclique  
ALWAYS CORRECT !!

If  $G$  is  $\epsilon$ -far from being a biclique  
?

NPTEL

33

Let us analyze this bit carefully well if  $G$  is in fact, a biclique then we are always going to be correct because whenever we going to find the violation, the algorithm is going to output an accept verdict which is going to be always correct. So, it boils down to being able to reject a graph that is epsilon far from being a biclique.

(Refer Slide Time: 11:03)

ANALYSIS

ASSUME  $G$  is  $\epsilon$ -far from being a  $k$ -clique

# of violating pairs  $> \epsilon n^2$

$\Pr$  [pair chosen in iteration  $i$  is a violating pair]  
 $> \epsilon n^2 / n^2 = \epsilon$

$\Pr$  [No violating pair is chosen]  $\leq (1 - \epsilon)^{2/\epsilon} \leq e^{-2}$

$\Pr$  [also correctly rejects  $G$ ] =  $\Pr$  [a violating pair chosen]  $> 1 - e^{-2} > 2/3$

37

So, for rest of the analysis we are going to assume there are graph is epsilon far from being a clique which means member that the number of violating pairs that least is greater than epsilon n square and so now, let us see what happens is we need to ensure that the probability with which we reject such a graph is at least a half.

So, let us see how this can work out, what is the probability? Let us focus on 1 iteration, I what is the probability that the pair chosen in that iteration  $i$  is a violating pair well there are total of  $i$  mean ensure 2 pair that we can choose from let us see even approximate that n square pairs, out of which at least epsilon n square pairs are violating pairs.

So, if you choose pairs uniformly at random then there a probability epsilon that at least probability epsilon that we will be choosing a violating pair, but what could happen as we may end up never choosing a violating pair, but what is the probability of that happening well for 1, 1 iteration is going to be 1 minus epsilon at most and over 2 over epsilon iterations the probability is going to be 1 minus epsilon the whole raise to 2 over epsilon which is going to be at most epsilon to the minus 2 when you apply the formula that 1 that 1 minus x is at most e to the minus x.



So, we can ask what is the probability that we will be correct, I mean this step is talking about the probability that we would be incorrect that we will not be able to pick that violate pick any violating pair well that just the probability the algorithm actually reject graph is simply the probability that some violating pair will be chosen which is simply  $1 - \epsilon^2$  at least  $1 - \epsilon^2$  and. So, the  $\epsilon^2$  is a small. So, like a less than a third we ensure that that the probability of correctly rejecting the algorithm  $g$  is at least  $\frac{2}{3}$ .

So, you can see that this enforce and test technique makes the analysis very clean and simple it the first step enforces a certain structure and then it becomes easy to test against that enforced structure. So, that is the main idea behind this approach and then ones that when we are the trick is to design an enforcement that will help us to test and reject instances that are far from the having the property that is the key inside.

Now, this brings us to the end of this second segment in the next subsequent segment we will look at testing whether a given graph is bipartite or not again in the dense craft model and that is a bit more for realistic problem and one step more interesting and challenging in terms of the algorithm and the analysis. So, that is going to be the next segment.