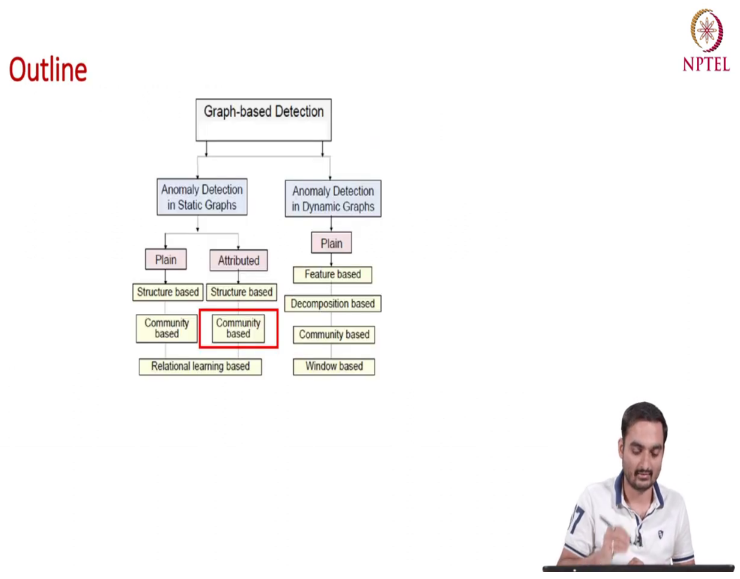
Social Network Analysis
Prof. Tanmoy Chakraborty
Department of Computer Science and Engineering
Indraprastha Institute of Information Technology, Delhi

Chapter - 08
Lecture - 04

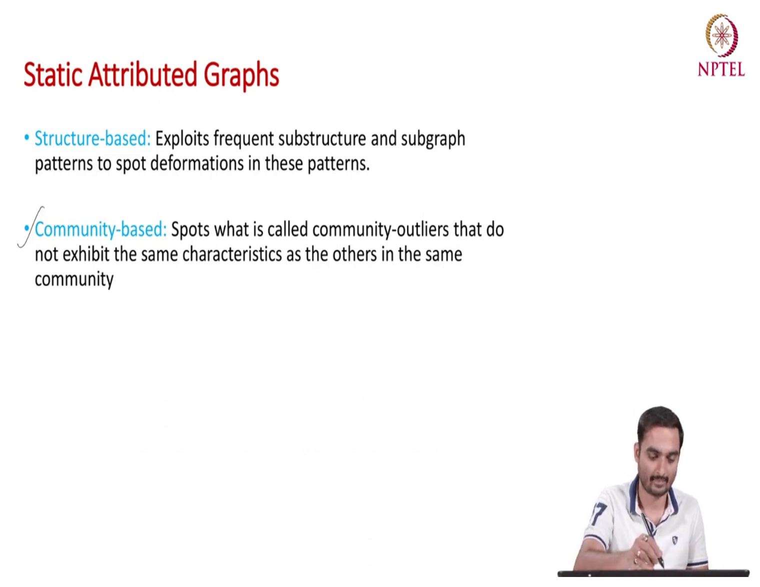
Alright, let us look at the attributed graph. So, so far we have only considered the plane graph right, plane static graph. Now, we will consider static attributed graphs ok.

(Refer Slide Time: 00:34)



And let us look at the structure based algorithms ok.

(Refer Slide Time: 00:37)



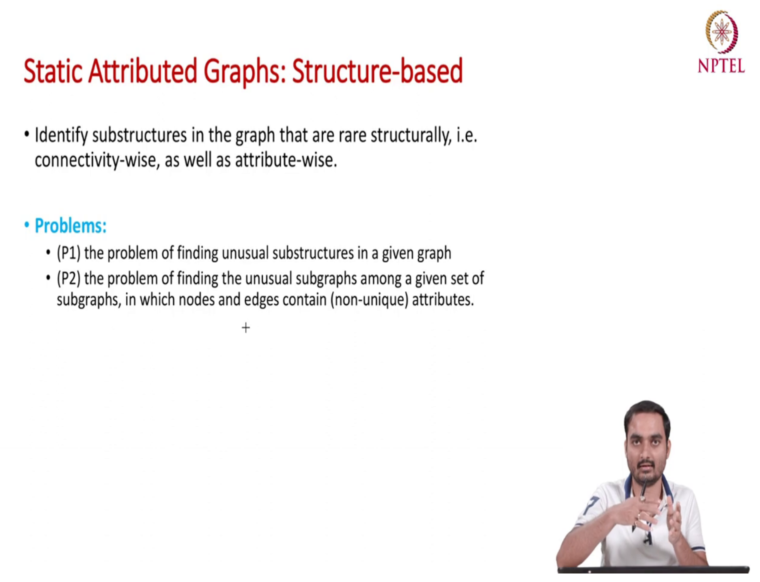
Static Attributed Graphs

- Structure-based: Exploits frequent substructure and subgraph patterns to spot deformations in these patterns.
- Community-based: Spots what is called community-outliers that do not exhibit the same characteristics as the others in the same community

So, here we look at two types of algorithms structure based algorithms and community based algorithms. So, in the structure based algorithms what it does they basically try to exploit you know frequent sub structures can be motif, can be any you know sub graph mining techniques. And sub graphs patterns to spot you know deformations in this pattern.

Whereas, in the community based approach we look at a community and we look at entities within a community and try to identify those entities which should not fit within the community ok. We will discuss one such algorithm in the community based you know outlet detection techniques.

(Refer Slide Time: 01:21)



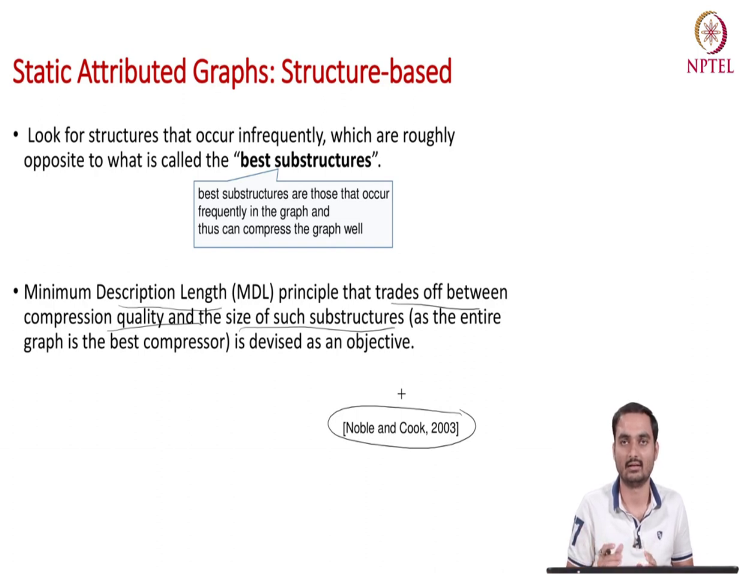
Static Attributed Graphs: Structure-based

- Identify substructures in the graph that are rare structurally, i.e. connectivity-wise, as well as attribute-wise.
- **Problems:**
  - (P1) the problem of finding unusual substructures in a given graph
  - (P2) the problem of finding the unusual subgraphs among a given set of subgraphs, in which nodes and edges contain (non-unique) attributes.

So, in the structure based algorithm for the attributed graph. So, the task is to identify sub structures in a graph. That are rare structurally that is connectivity wise as well as attribute wise ok. So, what are the problems that we are solving here? The first problem is the problem of finding unusual substructures in a given graph. And once we identify unusual substructure the next task is to identify how unique ok that substructure is right.

Node and nodes and edges contains I mean that part would be unique and that would be non unique attributes in the sense like you know if you look at if you look at the you know unusual substructures possibly attributes are very unique, but there are some attributes for some nodes whose attributes are non unique right. And we identify those as outliers ok.

(Refer Slide Time: 02:29)



Static Attributed Graphs: Structure-based

- Look for structures that occur infrequently, which are roughly opposite to what is called the "**best substructures**".

best substructures are those that occur frequently in the graph and thus can compress the graph well

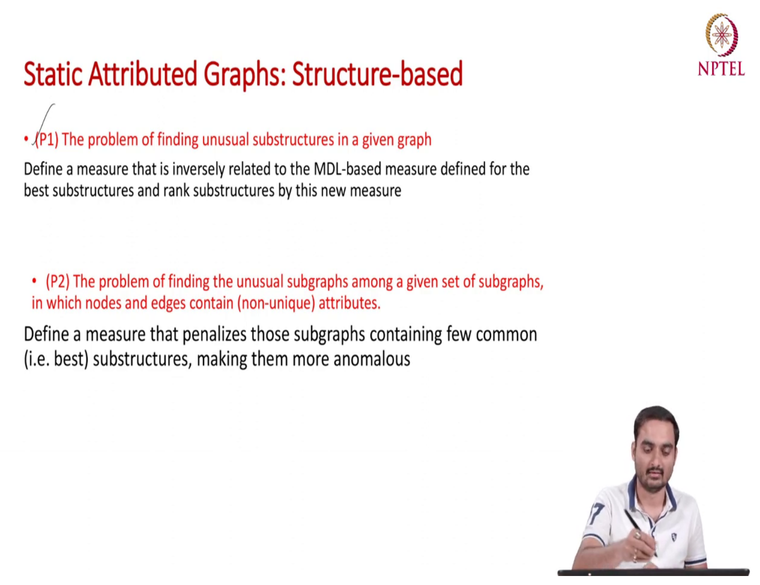
- Minimum Description Length (MDL) principle that trades off between compression quality and the size of such substructures (as the entire graph is the best compressor) is devised as an objective.

[Noble and Cook, 2003]

So, what people do? People try to identify roughly that; what is best substructure ok? So, the best substructures are those that occur frequently in the graph and thus are and thus can compress the graph well. Again we are looking at it from a minimum description lens point of view, information theoretic point of view and we will use minimum description length principle that basically trades off between the compression quality and the size of the substructure right.

This compression quality and the size of the substructure are devised as the objective function right. So, this is this famous paper Noble and Cook. If you want you can look at this paper I am not again going into the details, because information theory itself you know you need to understand this concept separately. That itself requires quite significant time. So, I will not go into that part. Rather I will go into those algorithms which are based on the foundations that we have discussed so far ok.

(Refer Slide Time: 03:42)



Static Attributed Graphs: Structure-based

- (P1) The problem of finding unusual substructures in a given graph

Define a measure that is inversely related to the MDL-based measure defined for the best substructures and rank substructures by this new measure

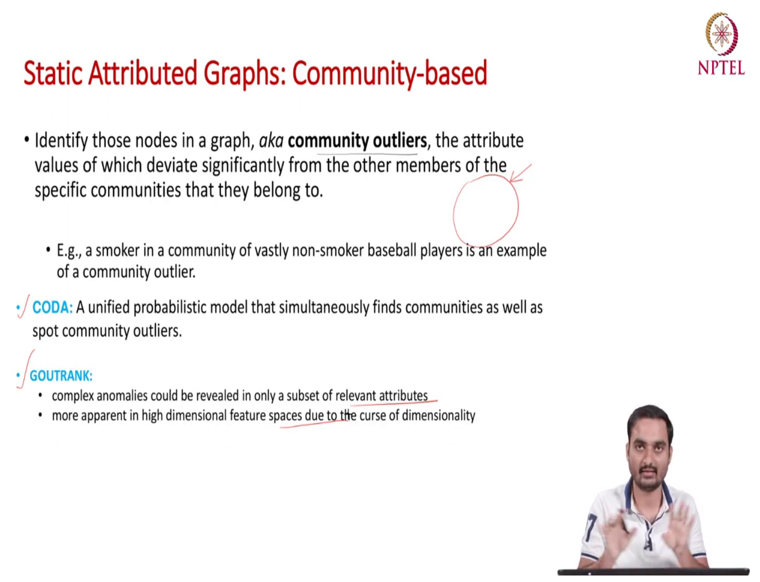
- (P2) The problem of finding the unusual subgraphs among a given set of subgraphs, in which nodes and edges contain (non-unique) attributes.

Define a measure that penalizes those subgraphs containing few common (i.e. best) substructures, making them more anomalous

And as I mentioned in problem 1; how to define, how to find unusual substructure? So, we define a measure that is inversely related to the minimum description length based measure, defined for the best substructures and then we rank substructure accordingly. And then how do we identify the uniqueness or non uniqueness. So, define a measure that penalizes right. Those sub graphs containing few or the best right few common or few best substructures making them more anomalous ok.

I understand this is little bit vague. I am not again going into the details of this part. Rather I will look at the second part and go and try to go deeper into this ok. So, this is for attributed static graph, but we look at community structure ok.

(Refer Slide Time: 04:43)



Static Attributed Graphs: Community-based

- Identify those nodes in a graph, *aka* **community outliers**, the attribute values of which deviate significantly from the other members of the specific communities that they belong to.
  - E.g., a smoker in a community of vastly non-smoker baseball players is an example of a community outlier.
- **CODA:** A unified probabilistic model that simultaneously finds communities as well as spot community outliers.
- **GOUTRANK:**
  - complex anomalies could be revealed in only a subset of relevant attributes
  - more apparent in high dimensional feature spaces due to the curse of dimensionality

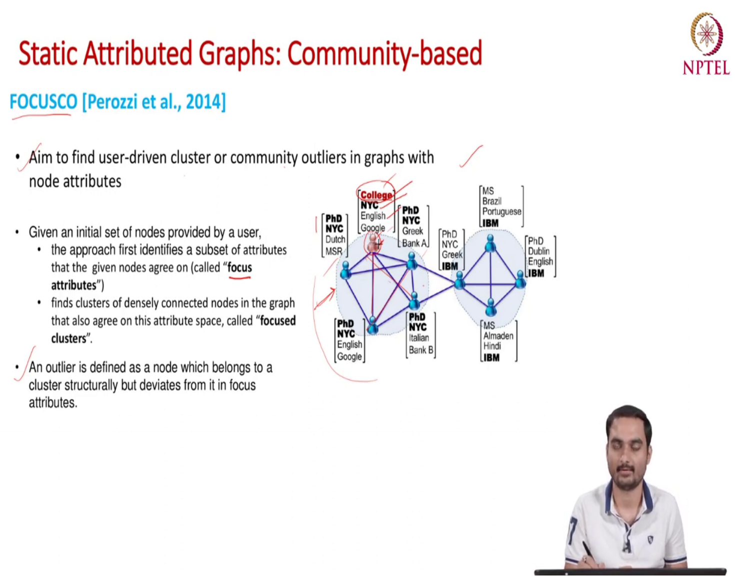
So, what is the intuition behind this? The intuition behind these types of algorithms is basically that we try to identify those nodes in a graph right. Also known as community outliers ok. The attribute values of which deviate significantly from the other members of the specific communities that they belong to ok.

What does it mean? Say think of a; think of a smoker right, in a community of vastly non smoker basketball players ok. There is a community of basketball players and most of them are non smokers, but there is one such or one or two such players who are smokers. So, with respect to this community, these two smoker nodes are outlier nodes, but if you look at the entire graph, there are many such smoker nodes right.

And smokers in general are not outliers right. So, but if you look at a particular community for that specific community those nodes are possibly outliers ok. So, these two algorithms the first one is coda. What it does ah? It is a unified probabilistic model that simultaneously finds communities as well as outliers right, from a graph.

In fact, all the algorithms that we consider all those algorithms identify communities and outliers side by side. The second one is called you know g o u t rank goutrank. So, this is again identify this identify complex anomalies right. That could be revealed in only a subset of relevant attributes. More apparent in high dimensional feature space due to the curves of dimensionality ok. So, these are more of a of an algorithm which do not use graph structure right. Generally in this feature space ok.

(Refer Slide Time: 06:52)



Static Attributed Graphs: Community-based

FOCUSCO [Perozzi et al., 2014]

- Aim to find user-driven cluster or community outliers in graphs with node attributes
- Given an initial set of nodes provided by a user,
    - the approach first identifies a subset of attributes that the given nodes agree on (called "**focus attributes**")
    - finds clusters of densely connected nodes in the graph that also agree on this attribute space, called "**focused clusters**".
- An outlier is defined as a node which belongs to a cluster structurally but deviates from it in focus attributes.

So, but we will discuss one such algorithm, which considers the graph structure. The attributes of nodes and edges and detects outliers. And this algorithm is called focusco. Focusco is an algorithm which identifies community centric outliers from a static attributed graph ok. So, let us take an example ok. Let us say this is the graph and there are two communities.

So, this is one community and in this community there are 5 nodes and each node is associated with 4 attributes ok. The degree educational degree it can be college, PhD etcetera. The current city or state or country whatever New York etcetera. Then language right mother tongue, spoken language Dutch, English, Greek etcetera. And then the current job right Google, Microsoft, bank etcetera right.

So, in this particular community if you look at it carefully right. And if you also say that hey in this community I will only look at two attributes ok. I will not look at other attributes, because this community is defined by these two attributes ok. So, and what are these attributes we are considering?

We are considering the first two attributes ok. So, the educational qualification and the current place ok. So, with respect to these two attributes. If you see all the nodes belong to New York City right. Most of the nodes have PhD degree, except this node this guy has only college degree right.

So, with respect to this community and, how can I say that this is a community? Look at the structural property right. They are highly connected right. They are highly connected. So, this is a community topological community and they are also attribute wise similar right, but this node is outlier with respect to this attribute ok. So, what is the aim of this algorithm?

This algorithm what it does it first identifies important attributes right, measures the weights of attributes. Then you know try to then tries to group nodes into communities depending upon the structural properties right. And then tries to identify nodes within that community, which are very different attribute wise ok. Structurally it may not be different, but attribute wise it may be different ok.

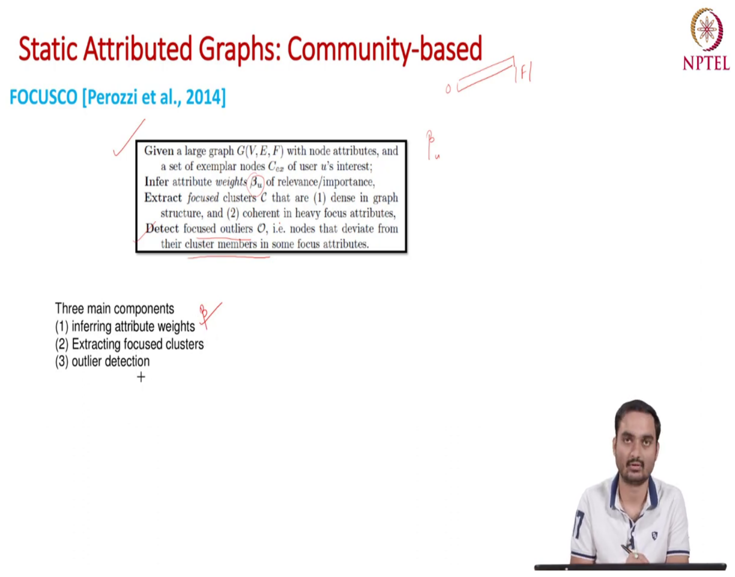
So, let us look at this algorithm carefully. So, the aim is to find user driven cluster or communities right. And therefore, we aim to identify user driven community outliers right, in a graph with node attributes. Nodes agree on right as I mentioned. We try to identify focus attributes or important attributes right.

We call them focus attributes and then it finds clusters of densely connected nodes in the graph that also agree on this attribute space called focus clusters ok. So, what is the idea? So, idea is that we let us assume that user has already given you that a set of nodes which are similar ok. So, what would be your task?

Your task would be to identify a cluster, where nodes of similar types of the exemplar nodes right will be clubbed together right and form a cluster. Now, within a cluster you identify focus attributes ok. The focus attributes which define this cluster; meaning the focus attributes which are unique to this cluster ok. And then we call this cluster as a focused cluster right. And an out layer is defined as a node which belongs to the cluster structurally, but deviates from it with respect to focused attributes ok.

So, this node belongs to this cluster structurally, because this guy is connected to all the other guys right, but attribute wise it is not that you know suitable to be placed within this cluster ok alright.

(Refer Slide Time: 12:17)



Static Attributed Graphs: Community-based

FOCUSCO [Perozzi et al., 2014]

Given a large graph $G(V, E, F)$ with node attributes, and a set of exemplar nodes C_{ex} of user u's interest;
Infer attribute *weights* $\boxed{\beta_u}$ of relevance/importance,
Extract *focused* clusters $\mathcal{C}$ that are (1) dense in graph structure, and (2) coherent in heavy focus attributes,
Detect focused outliers $\mathcal{O}$, i.e. nodes that deviate from their cluster members in some focus attributes.

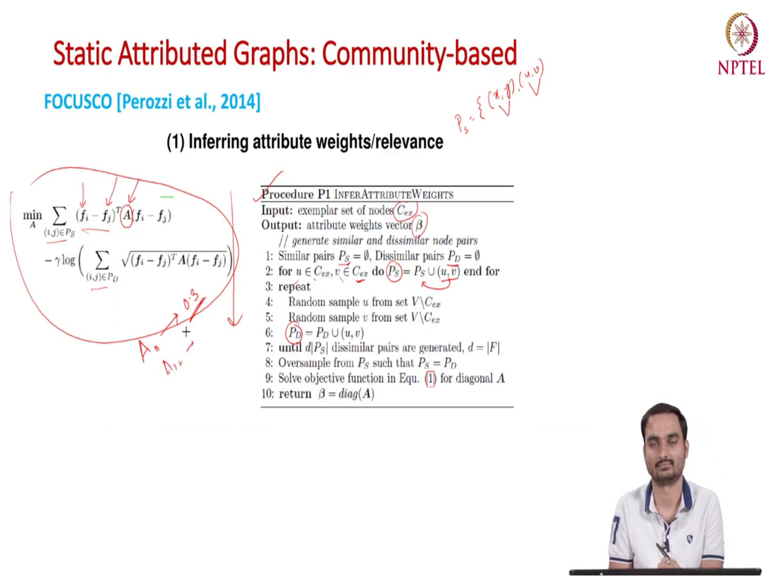
Three main components
(1) inferring attribute weights
(2) Extracting focused clusters
(3) outlier detection

So, this is the problem statement given a large graph G V comma E comma F. V is the set of nodes, E is the set of edges and F is a set of attributes. Remember each node has a vector of size mod F of attributes ok. And we are also given a set of exemplar nodes Cex right.

So, the Cex set of nodes are similar ok both structurally and attribute wise. And what is the aim? The aim is to infer weights of attributes ok beta. So, this beta u is the weight of the attribute u right and the higher the weight for a given cluster more likely that this at the corresponding attribute is a focused attribute ok.

Then extract focused cluster C, that are dense in the graph structure and coherent in heavy focus attributes ok. And then we detect focused outliers O, that is nodes that deviate from their cluster members with respect to some focused attributes ok. So, what are the main components inferring attribute weights beta, extracting focused clusters then outlet detection ok alright.

(Refer Slide Time: 14:06)



Static Attributed Graphs: Community-based

FOCUSCO [Perozzi et al., 2014]

(1) Inferring attribute weights/relevance

$$\min_{A} \sum_{(i,j)\in P_S} (f_i - f_j)^T A (f_i - f_j) - \gamma \log \left(\sum_{(i,j)\in P_D} \sqrt{(f_i - f_j)^T A (f_i - f_j)} \right)$$

```
Procedure P1 InferAttributeWeights
Input: exemplar set of nodes $C_{ex}$
Output: attribute weights vector $\beta$
    // generate similar and dissimilar node pairs
 1: Similar pairs $P_S = \emptyset$, Dissimilar pairs $P_D = \emptyset$
 2: for $u \in C_{ex}, v \in C_{ex}$ do $P_S = P_S \cup (u, v)$ end for
 3: repeat
 4:     Random sample $u$ from set $V \backslash C_{ex}$
 5:     Random sample $v$ from set $V \backslash C_{ex}$
 6:     $P_D = P_D \cup (u, v)$
 7: until $d|P_S|$ dissimilar pairs are generated, $d = |F|$
 8: Oversample from $P_S$ such that $P_S = P_D$
 9: Solve objective function in Equ. (1) for diagonal $A$
10: return $\beta = diag(A)$
```

So, the first task is to infer attribute weights ok. And how do we do that? So, this is the procedure algorithm that they proposed ok. So, what we have? We have as an input the set of exampler nodes Cex ok. And what we want to derive? We want to derive beta ok. So, now, what we do?

We define two sets right P S is the set of node pairs, which are similar. Remember P S consists of node pairs like x y right u v and so on and so forth, which are similar. So, x and y are similar u and v are similar and so on and so forth. P S, this S corresponds to similar nodes.

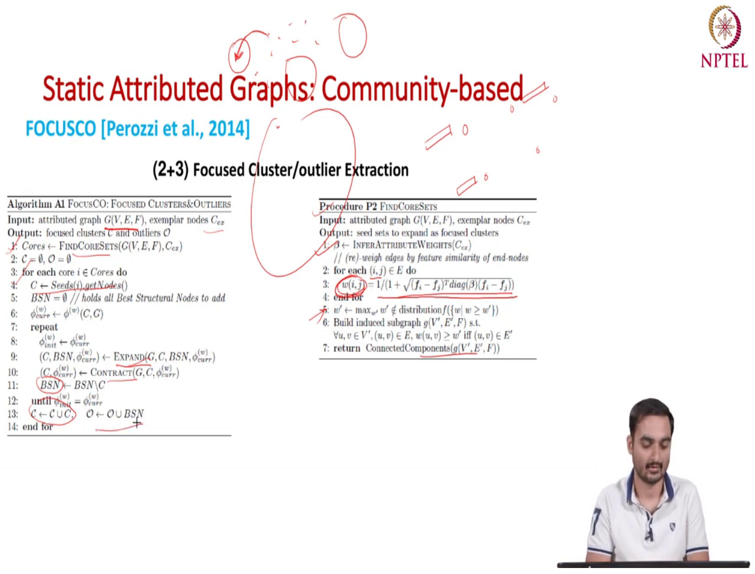
We also define another set called P D, dissimilar nodes set of dissimilar nodes ok. So, how do we populate this P S set of similar nodes? We take a node from we take a pair of nodes from Cex right, u and v right. And we create a pair u v and then we add this pair to P S ok, because we know that within Cex all the nodes are similar ok. So, we take pairs and we basically populate P S. And how do we populate P D? We take all other nodes which are not part of Cx right. And similarly we take pairs right.

And then we populate P D ok. So, but what we do? We try to we try to create a P D, whose size would be much much higher than P S ok. And then what is our objective function? Our objective function is to minimize this quantity ok and what is this? So, if i is a feature vector of node i and if j is the feature vector of j. For each pair in P S ok. We try to minimize this distance, because these two nodes are similar.

And for each pair of nodes in P D, we try to maximize the distance, because they are dissimilar ok. And we try to learn a matrix A, this A is a learnable matrix right, which basically looks at P S and P D and try to minimize this. This is a very standard mole based distance measure ok between a pair of vectors. So, at the end of the day what we will have? We have a matrix A right and this matrix A would be a diagonal matrix right.

So, A 11 would be say let us say A 11 is 0.3. So, this is the weight of the first attribute. Similarly A 22 is the weight of the second attribute.

(Refer Slide Time: 18:11)



Static Attributed Graphs: Community-based

FOCUSCO [Perozzi et al., 2014]

(2+3) Focused Cluster/outlier Extraction

```
Algorithm A1 FocusCO: Focused Clusters&Outliers
Input: attributed graph $G(V,E,F)$, exemplar nodes $C_{ex}$
Output: focused clusters $\mathcal{C}$ and outliers $\mathcal{O}$
1: $Cores \leftarrow$ FindCoreSets($G(V,E,F), C_{ex}$)
2: $\mathcal{C} = \emptyset, \mathcal{O} = \emptyset$
3: for each core $i \in Cores$ do
4:     $C \leftarrow Seeds(i).getNodes()$
5:     $BSN = \emptyset$ // holds all Best Structural Nodes to add
6:     $\phi^{(w)}_{curr} \leftarrow \phi^{(w)}(C,G)$
7:     repeat
8:         $\phi^{(w)}_{init} \leftarrow \phi^{(w)}_{curr}$
9:         $(C, BSN, \phi^{(w)}_{curr}) \leftarrow$ Expand($G, C, BSN, \phi^{(w)}_{curr}$)
10:        $(C, \phi^{(w)}_{curr}) \leftarrow$ Contract($G, C, \phi^{(w)}_{curr}$)
11:        $BSN \leftarrow BSN \setminus C$
12:    until $\phi^{(w)}_{init} = \phi^{(w)}_{curr}$
13:    $\mathcal{C} \leftarrow \mathcal{C} \cup C$,    $\mathcal{O} \leftarrow \mathcal{O} \cup BSN$
14: end for
```

```
Procedure P2 FindCoreSets
Input: attributed graph $G(V,E,F)$, exemplar nodes $C_{ex}$
Output: seed sets to expand as focused clusters
1: $\beta \leftarrow$ InferAttributeWeights($C_{ex}$)
   // (re)-weigh edges by feature similarity of end-nodes
2: for each $(i,j) \in E$ do
3:     $w(i,j) = 1/(1 + \sqrt{(f_i - f_j)^T diag(\beta)(f_i - f_j)})$
4: end for
5: $w' \leftarrow \max_{w'}, w' \notin$ distribution $f(\{w \mid w \geq w'\})$
6: Build induced subgraph $g(V', E', F)$ s.t.
   $\forall u,v \in V', (u,v) \in E, w(u,v) \geq w'$ iff $(u,v) \in E'$
7: return ConnectedComponents($g(V', E', F)$)
```

You know weights of the attributes right. Let us look at the algorithm now focusco algorithm. So, the input is a graph and the set of exemplar nodes. This set of exemplar nodes are used to measure the importance of attributes. So, the first task is to identify the cores right the core components using this function find core set ok. So, in the find code set right.

First we identify this beta, beta is the weights of the attributes and then what we do? So, we have this graph right. We also know the attributes of nodes right. We create another graph on top of this and this graph will be created based on the similarity of attributes ok. So, for every node pair i j right. We measure the weight; this weight is measured by the inverse of the distance ok.

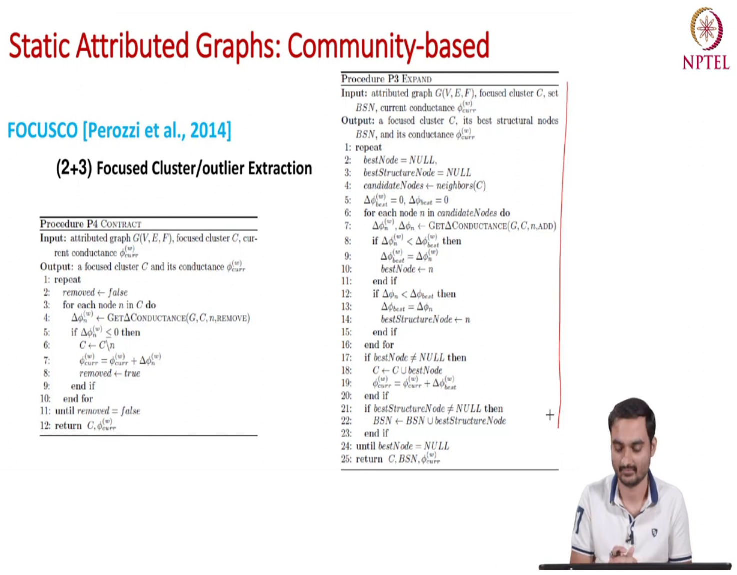
So, this weight is essentially a distance. So, higher the similarity lower the distance right. So, this w i j will give a weight of the age between i and j right. Then what we do? We identify a threshold right. We only keep those edges whose weights are above the threshold ok. So, from the original graph now we get you know nodes which are connected right, but can have components right.

We have multiple components, because we have removed some edges whose weights are less right. So, we call these components as the core set. So, these components are similar in terms of both the attributes as well as topology ok. So, we find such cores right and then for every core i. So, let us say this is one core another core other core and there are nodes which are not parts of this core ok. So, for every core we take all its corresponding nodes.

And what we do? We measure the conductance right, conductance cart we discussed in the communication chapter. So, we measure the conductance you know by including a node into the core set, which is not a part of the core set now, but after inclusion the conductance value may decrease. Remember lower the conductance higher the clustering the you know the cluster likelihood ok.

So, we add this node and check the conductance ok. We keep on adding such nodes into the core sets and see whether the conductance value decreases or not ok. And we keep on adding until unless we see the conduct value conductance value starts increasing. We can also use modularity for example, right. So, this expand procedure right

(Refer Slide Time: 22:14)



Static Attributed Graphs: Community-based

FOCUSCO [Perozzi et al., 2014]

(2+3) Focused Cluster/outlier Extraction

```
Procedure P4 CONTRACT
Input: attributed graph $G(V,E,F)$, focused cluster $C$, current conductance $\phi^{(w)}_{curr}$
Output: a focused cluster $C$ and its conductance $\phi^{(w)}_{curr}$
 1: repeat
 2:   $removed \leftarrow false$
 3:   for each node $n$ in $C$ do
 4:     $\Delta\phi^{(w)}_n \leftarrow$ GETΔCONDUCTANCE($G, C, n$, REMOVE)
 5:     if $\Delta\phi^{(w)}_n \leq 0$ then
 6:       $C \leftarrow C \setminus n$
 7:       $\phi^{(w)}_{curr} = \phi^{(w)}_{curr} + \Delta\phi^{(w)}_n$
 8:       $removed \leftarrow true$
 9:     end if
10:   end for
11: until $removed = false$
12: return $C, \phi^{(w)}_{curr}$
```

```
Procedure P3 EXPAND
Input: attributed graph $G(V,E,F)$, focused cluster $C$, set $BSN$, current conductance $\phi^{(w)}_{curr}$
Output: a focused cluster $C$, its best structural nodes $BSN$, and its conductance $\phi^{(w)}_{curr}$
 1: repeat
 2:   $bestNode = NULL$,
 3:   $bestStructureNode = NULL$
 4:   $candidateNodes \leftarrow neighbors(C)$
 5:   $\Delta\phi^{(w)}_{best} = 0$, $\Delta\phi_{best} = 0$
 6:   for each node $n$ in $candidateNodes$ do
 7:     $\Delta\phi^{(w)}_n, \Delta\phi_n \leftarrow$ GETΔCONDUCTANCE($G, C, n$, ADD)
 8:     if $\Delta\phi^{(w)}_n < \Delta\phi^{(w)}_{best}$ then
 9:       $\Delta\phi^{(w)}_{best} = \Delta\phi^{(w)}_n$
10:       $bestNode \leftarrow n$
11:     end if
12:     if $\Delta\phi_n < \Delta\phi_{best}$ then
13:       $\Delta\phi_{best} = \Delta\phi_n$
14:       $bestStructureNode \leftarrow n$
15:     end if
16:   end for
17:   if $bestNode \neq NULL$ then
18:     $C \leftarrow C \cup bestNode$
19:     $\phi^{(w)}_{curr} = \phi^{(w)}_{curr} + \Delta\phi^{(w)}_{best}$
20:   end if
21:   if $bestStructureNode \neq NULL$ then
22:     $BSN \leftarrow BSN \cup bestStructureNode$
23:   end if
24: until $bestNode = NULL$
25: return $C, BSN, \phi^{(w)}_{curr}$
```

I am not going into the details, but the expand procedure basically expands different components right, by adding potential nodes which are not part of the core components. But after addition the quality of the this community gets increased right. Similarly once we expand individual components, the next part would be it may happen since this is a greedy algorithm. It may happen that after certain times you realize that you have added some nodes we should not have been added right.

So, what we do? We similarly perform contract this contraction operation keeps on removing nodes from the course until unless we see that the conduction value conductance value increases. So, it is a greedy approach. So, we need to you know repeat this process multiple times to make the communities refined right.

So, through this process what would happen is that there would have some you know you will find some nodes right, which would be removed from the from the code set right. Based on, because code set is this defined based on both the topology and attributes, but with respect to only the topology those nodes will be there.

But with respect to both topology and attributes those nodes will be removed ok. And we call this nodes as BSN. So, BSN is a superset from this BSN right, best structural nodes; BSN best structural nodes. Among the BSN set there are some nodes which are also similar attribute wise. So, they would be included in the core set right, but those which would remain after the iteration in the BSN set those would be the outliers ok.

So, this is all about focusco algorithm. So, we start off with a set of exampler nodes right. We identify the importance of attributes. We then create another graph; meaning weights of edges. Then we identify components right. Those components are cores we keep on expanding the cores through expansion and contraction operations and through the process we will we are left with some nodes, which are structurally similar, but attribute wise difference different ok. And those nodes are called as outliers ok

So, this is focusco. We stop here and this is basically about a community centric outlier detection. So, we stop here. In the next lecture we will discuss you know the relation learning algorithms and we will also look at dynamic algorithms dynamic graphs and outlet detection from dynamic graphs ok.

Thank you.