Local community detection in a graph generated by Preferntial deletion

Prithika Vijayakumar

I. INTRODUCTION

The proposed project will be able to identify community (locally dense and globally sparse structure) within a networks that are strongly connected within themselves than to other networks. This will help in identification of various categories of a very large web like network from which various insights can be obtained. .A dynamic random graph models that combine node and edge addition with a uniform and a preferential deletion of nodes will be created according to the method quoted in the paper "Preferential deletion in dynamic models of web-like networks" by Narsingh Deo , Aurel Cami.Later at a last timestep communitiy for a given set of seed nodes is detected and outputted as result.This is done by using Bagrow and Bollt's community detection algorithm as quoted in the paper "Local method for detecting communities" by James P. Bagrow and Erik M. Bollt,that will try to detect a local community to which a given seed belongs.the algorithm is more clearly stated in the paper "A Greedy Community-Mining Algorithm Based on Clustering Coefficient " which is the basis with which the implementation is done.

There are several subgraph detection procedures such as Louvain algorithm which is a hierarchical clustering algorithm and Newman and Girvan that are widely used today but these do not performs well within a variety of networks

II. APPROACH

the idea is to make use of the dynamic graph, take snapshot of the generated graph for a particular time step and find the community to which a given node of the graph belongs to.To perform this a clustering coefficient will be used to find how clustered the nodes are together compared to nodes futher appart.This method would try to determine the density of nodes around a given node (initial community), increasing the community at every step or layer by finding its density relation with nodes adjacent to it.

III. METHOD DESCRIPTION

A dynamic graph as stated in the paper 1 creates a discrete time dynamic random Graph implementation that interleaves the birth of nodes and edges with the death of nodes.The birth of the node follows a preferntial linear attachment process and he death process is done by preferntial deletion.It is also found that the birth-death models follows a power law distribution.

To create a random graph,With a small fixed graph at subsequent time step a new node is added to the graph or an existing node is deleted from the graph.The dynamic random graph is created by 2 process birth process and the death process;

Birth Process

A new node with an incident edge is added to the graph with a probability p.The node to which this new node is attached is obtained by a linear preferential model:

$$P_{t+1}[u] = \frac{d_t(u)}{2m_t}$$

The node with maximum number of connectivity is the target node for the new node

Death process

With probability $q = p - 1$ An existing node of the graph is deleted from graph which has the least number of connectivity.This is called preferential deletion where a node with least degree is chosen for deletion. To get the node of smallest degree, the following formula is calculated for every node and the max of the these distribution values will give the node with least degree

$$P_{t+1}[u] = \frac{n_t - d_t(u)}{n_t^2 - 2m_t}$$

Community detection

Using the algorithm stated in paper 2 ,community is detected for a set of seed node.Here in our project the seed nodes are taken to be the 2 nodes that was created the oldest.The proposed algorithm consists of an l shell spreading outward from a starting vertex or a seed. As the starting vertex's nearest neighbors and next-nearest neighbors, etc., are visited by the l shell, two quantities are computed: the emerging degree and total emerging degree. The emerging degree of a vertex is defined as the number of edges that connect that vertex to vertices the l shell has not already visited as it expanded from the previous l1,l2,... shell.Emerging shell does not consider the vertices within the same shell. The total emerging degree of an l shell is then the sum of the emerging degrees of all vertices on the leading edge of the l shell. The algorithm works by expanding an l shell outward from some starting vertex j and comparing the change in total emerging degree to some threshold α .The algorithm in the paper uses α value to be 0.5 whereas here in my project the value of α is set to 0 so that the change in total emerging degree is positive (i.e the algorithm will add into community all nodes whose number of connection with nodes inside existing community is equal or more to the ones outside. This will allow the algorithm to add all nodes whose degree is equally and more clustered to the community than others.

Another method of detecting community using

modularity is also implemented using python built-in function.greedy_modularity_communities() that finds communities in graph using Clauset-Newman-Moore greedy modularity maximization.Greedy modularity maximization begins with each node in its own community and joins the pair of communities that most increases modularity until no such pair exists.

IV. Implementation

Implementation of dynamic graph is done first with randomly generating probability p and q that decides whether an birth or death process would take place.Initially a node is created with an edge it itself.A nodelist of nodes of the graph is used along with a list blist that stores birth probabilities of each node in nodelist(graph) and a list dlist that stores death probabilities of each node of the graph.

Birth process implementation

Whenever the a random variable is chosen whose probability is p a birth process takes place.A node with with max birth probability (calculated using birth process formula) is selected by calling function Cummulativeprob() as a node to which a new node is to be attached.Then birthnode() function is called to attach the new node to the node selected by CumulativeProb().This function would attach the node and update all the parameters of both the new node and selected node(updates edges and adjacent node).It also updates the new birth and death probabilities of the blist and dlist by calling the bdcal() function.

death process implementation

Whenever the a random variable is chosen whose probability is q a death process takes place.A node with with max death probability (calculated using death process formula) is selected by chosing the node with max dlist value .Now this node is deleted using deathnode() function.This function delete the node from the nodelist and update all the parameters of adjacent node of the deleted function (updates edges and adjacent node).It also updates the new birth and death probabilities of the blist and dlist by calling the bdcal() function.If all the node of the graph are deleted then the begin process is initiated where a node with a self loop is created as a seed node.

After each time step the number of nodes and the number of edges of the graph are updated in an array Nodes and Edges respectively and a network is plotted with t=100

community detection

For simplicity the project takes the graph at t=100(after complete cycle of creation of dynamic graph) however a community for a seed node at any timestep can be created.Here in this project the seed nodes are taken to be the 2 nodes that were created the oldest.$1^{st} and\ 2^{nd}$ nodes of the nodes list.These 2 nodes form the community comm,then nodes adjacent to these two node that are not in the community are taken to be layer 1 nodes and nodes that are adjacent to layer 1 and not in layer 1 and comm are taken to be layer 2 nodes.

Now for every node in layer 1, the clustering coefficient is calculated with respect to nodes inside the community (shell l)(c0v),with respect to nodes within layer 1(c1v) and with respect to nodes outside layer 1(c2v).The clustering coefficient values are simply the count of edges between nodes.Now these values are compared such that for every node in layer 1 ,if they have more edges connected to the community or within the layer than edges connected in layer 2 then these nodes are added to the community.(i.e if c0v-c2v¿=0 or c1v=c2v¿=0 then it is added to the community).The value of α in my project is chosen to be 0 so as to count the nodes that are equally likely to be connected with outside layer.This way it allows to include nodes that have partial link to the communtity.This process in done until layer1 and layer 2 values become small.

The result obtained is plotted as a graph to represent the community of the first and second created nodes.Another approach to community detection is also implemented in the paper that uses modularity to find all the communities in the network.For this community.greedy_modularity_communities(G) built in function is used.A list of all communities in the network is displayed in the console.

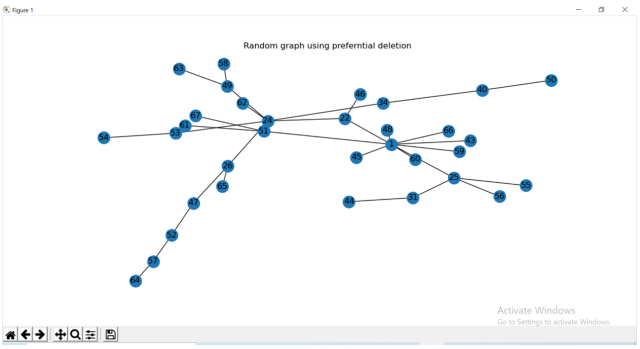

Fig. 1: Random graph generated by preferential deletion and linear attachment of nodes.

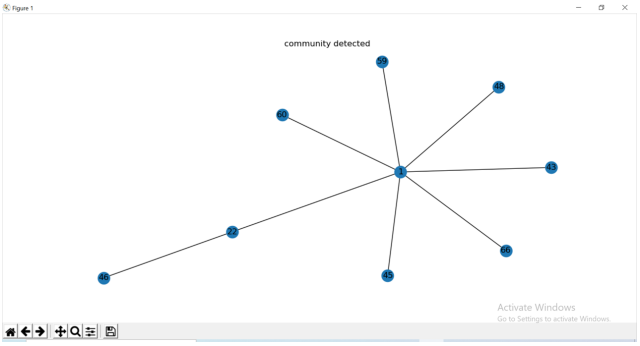

Fig. 2: community detected for seed node 1 and 22.

V. Results

The community detected by clustering coefficient is more efficient to the community detected by modularity process.This is because the community by clustering coefficient is also able to detect nodes that are equally partially connected and is also able to detect community

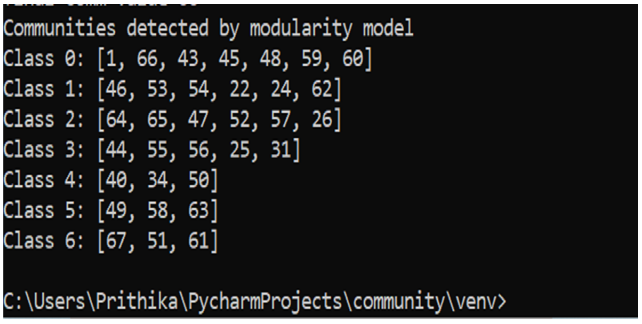

Fig. 3: List of communities detected by the greedy modularity built-in funtion.

for 1 or more nodes(a seed set) whereas the model using modularity detects all possible communities but another algorithm has to be run to find the community to which a node belongs.

VI. Conclusion

The methods for generating a dynamic random graph was implements that includes adding a node with probability p to a node with most edges and removing a node from network with probability q that has the least number of connections (edges).This is done by linear attachment and preferential deletion respectively.It is observed that the graph generated by preferential deletion follows a power law distribution.Further using a greedy, best-first algorithm the community to which a given set of nodes belongs was discovered.For simplicity the community algorithm runs for the graph generated at the last timestep.This process can also be randomised by choosing a timestep randomly for which the algorithm runs.

VII. Reference

Reference was taken from previous implementation of the paper to understand the logical construction of the graph.

1.https://github.com/HelenaJulie/-Preferential-deletion-in-dynamic-models-of-web-like-networks-implementation/blob/master/AlgoAssignment2.pyL208
2.https://github.com/sdevkota007/dynamic_graph/blob/master/simulate.py
3.https://programminghistorian.org/en/lessons/exploring-and-analyzing-network-data-with-pythonbasics-of-networkx-creating-the-graph

Refernce papers:

1) ”Preferntial deletion in dynamic models of web-like network” by Aurel Cami and Narsingh Deo
2) ”A Greedy Community-Mining Algorithm Based on Clustering Coefficient ”Aurel Cami and Narsingh Deo
3) ”Local method for detecting communities” by Aurel Cami and Narsingh Deo.