CSC 591 GRAPH DATA MINING
COMMUNITY DETECTION
PROJECT 2 REPORT

tkini@ncsu.edu

The paper implemented is Efficient Identification of Overlapping Communities? Link.
The project is hosted on Github. Link

Algorithm Description:
The algorithm presented in this paper aims at effectively identifying communities that are overlapping and not disjoint in nature. This is important as in the real world there are usually communities that are overlapping and the intersection can be given multiple truth values. The algorithm takes into consideration only the edges between the nodes and no other information like the weight, labels etc. The algorithm is configured for an undirected graph and only the metric needs to be adapted if a directed graph is used. The algorithm relies on density as a metric for assessing the cluster quality. We try to find the local maximum density of the cluster with respect to its neighbors that are close to it.
There are 2 main algorithms implemented in the code which are as follows-

- LA- this function initializes the seed clusters that serve as a starting point. This improves on the RaRe algorithm(Rank Removal) and uses page rank of the nodes to order them at the start of the algorithm. We maintain the clusters in the graph like an adjacency list. We iterate through the list of vertices in non increasing order of their page ranks and check for all the clusters if adding this node in it would increase its density metric. If so then we include it in that cluster or else we leave it out. If no existing cluster accepts this node then a new cluster containing only this node is added to the list of clusters.
- IS2- this function iteratively scans through the vertices to improve the cluster according to the metric and stops when it finds a locally optimal collection of clusters. In this algorithm we take a seed cluster and calculate its density. Then we find all the adjacent nodes of the nodes in the cluster and then do one of the following. If the adjacent node is already in the cluster then we remove it from the cluster and if it is originally not we add it to the cluster. If the density is improved then we update our cluster.

Parameter effect on Algorithm:
The algorithm does not take any parameter for both the algorithms that were described above. The only thing the algorithm uses is the edges and the cluster density as a metric(which is derived from the edges) to run itself.

Theoretical Performance evaluation:
To asses the run time of the algorithms we consider the following parameters-

- |C| is the tight bond on the number of clusters at any time in the algorithm.
- |E| is the number of edges in the graph
- |N| is the number of nodes in the graph

The LA algorithm would in each iteration go through the list of vertices and the clusters. The density calculation is assumed to be completed in constant time for the graphs passed. Adding the vertex in question doesn't take time as it is an adjacency list representation. Thus we can say that the Time complexity for the algorithm would be O(|C|*|V|). For the space complexity part we have to maintain a list of vertices in O(|V|) and an adjacency list of clusters in O(|C|+k|V|)~ O(|C|+|V|). In totality the space compel;xity is O(|C|+|V|).

For the IS2 algorithm we see that the clusters from LA are given as input to IS2. We need to run IS2 for all the clusters bound by O(|C|) and have to go through the edge list of each node in it bound by O(2|E|). Thus we can say that the runtime of the algorithm is bound by O(|C|*|E|).

Goodness metric and Performance metric:
The metrics generated for understanding of results in the *.pmetrics.cvs files are-

For amazon.medium.graph

```
TP,FP,FN,TN,Recall,Precision,F-Measure,Specificity,RAND,NMI
28427,33,35176,12408879,0.44694433,0.99884048,0.61755537,0.99999734,0.9971
7707,0.65379195
```

For amazon.small.graph

```
TP,FP,FN,TN,Recall,Precision,F-Measure,Specificity,RAND,NMI
13032,32,6677,3104009,0.66122076,0.99755052,0.7952888,0.99998969,0.9978522
6,0.75676894
```

For dblp.medium.graph

```
TP,FP,FN,TN,Recall,Precision,F-Measure,Specificity,RAND,NMI
14103,572,8085,12459746,0.63561385,0.96102215,0.76515747,0.99995409,0.9993
0647,0.75467286
```

For dblp.small.graph

```
TP,FP,FN,TN,Recall,Precision,F-Measure,Specificity,RAND,NMI
7004,90,3906,3112750,0.64197984,0.98731322,0.77804932,0.99997109,0.9987207
7,0.76684147
```

For youtube.medium.graph

```
TP,FP,FN,TN,Recall,Precision,F-Measure,Specificity,RAND,NMI
5583,1009,11041,12479867,0.33583975,0.84693568,0.48096141,0.99991916,0.999
03581,0.77809085
```

For youtube.small.graph

```
TP,FP,FN,TN,Recall,Precision,F-Measure,Specificity,RAND,NMI
2573,279,1797,3121601,0.58878719,0.90217391,0.712545,0.99991063,0.99933595
,0.82215244
```

Goodness metric and Performance metric understandings:
Ideally we would like to have high Recall and Precision values. We can see that the precision values are high(>0.95) for all the graphs above. That means that the positives reported by our algorithm were right. The recall measures the proportion of the actual ground truth reported by our algorithm which is not great. It might be due to the fact that the ground truth given was not taking into the fact that overlapping communities do exist and the community to which a node belongs and found last would be present. The RAND and NMI values are also high for all graphs that is a good indication.Overall we can say that our algorithm implementation did pretty well.