CSC 591 (605) P2: Community Detection

Vivek Reddy Karri[1] **and Yashwanth Reddy Soodini**[2]

[1]**ysoodin@ncsu.edu**
[2]**vkarri@ncsu.edu**

ABSTRACT

This report describes the algorithm used for the identification of communities, the dataset used, and the results obtained.

Keywords: Graph Data Mining, Community Detection, neo4j

INTRODUCTION

In their paper, Baumes et al. (2005b) have presented an algorithm for identifying overlapping communities that only requires data on who communicated with whom. While the authors presented the paper with social communities in mind, the algorithm can be applied to any graph by generalizing communication between members of a social community as edges between the nodes of a graph.

The algorithm consists of two parts: **initialization**,LA, which creates *seed clusters*; and **improvement**, IS^2, which iteratively improves any given set of clusters. Baumes et al. (2005b) state that the algorithm can be applied to networks with $\sim 10^6$ nodes.

COMMUNITY SCORING FUNCTION

The community scoring function considered in the paper is *the density of a group of actors in a social network*, defined as the average density of communication exchanges between the actors of the group. Since we were going to experiment with general graphs, we used Conductance as our scoring function.

Conductance

Given a set of nodes S, conductance reveals the extent to which S qualifies as a community by considering both the internal and the external connectivity of S. There exist more than one notions of conductance. The notion which we considered and experimented with is the following:

The conductance of a set of nodes S in an undirected graph G(V,E) is defined as

$$cond(S) = \frac{p_S}{2q_S + p_S}$$

where q_S is the number of edges in S and p_S is the number of edges on the boundary of S.

ALGORITHMS

The Link Aggregate Algorithm (LA)

The Link Aggregate Algorithm is an initialization algorithm i.e., it generates clusters which are given as initial guesses for IS^2. IS^2 then improves each of these initial clusters to produce the final communities.

The set of clusters is initially empty. The nodes of the graph in question are first ordered according to their PageRank (other criterion can also be used to order the nodes). These ordered nodes are then processed sequentially. At every iteration, the current node is added to all the clusters whose conductance improves upon adding the current node to them. If the current node is not added to any of the clusters, then a new cluster is created with the current node. This ensures that every node is in at least one cluster.

The pseudocode for LA is given below. The procedure takes a graph and the Conductance function as input and returns a set of clusters.

procedure LA($G = (V,E), Cond$)
$C \leftarrow \theta$;
Order the vertices $v_1, v_2, ..., v_{|V|}$ according to their PageRank
for $i = 1$ to $|V|$ **do**
    $added \leftarrow$ false
    for all $c \in C$ **do**
        if $Cond(c \cup v_i) > Cond(c)$ **then**
            $c \leftarrow c \cup v_i$;
            $added \leftarrow$ true;
        if $added =$ false **then**
            $C \leftarrow C \cup \{\{v_i\}\}$;
return C;

The Improved Iterative Scan Algorithm (IS^2)

The Improved Iterative Scan Algorithm is an improved version of the original Iterative Scan algorithm proposed by Baumes et al. (2005a) in their previous paper. IS^2 takes a seed cluster as its input and constructs a cluster that is a local optimum w.r.t the conductance score.

A set consisting of all the nodes already in the cluster and the nodes that are in the immediate neighbourhood of the cluster is formed. The nodes in the set are then processed one by one. If the current node is already in the cluster, it is removed from the cluster if removing it decreases the conductance score of the cluster. If the current node is not in the cluster, then it is added to the cluster if adding it decreases the conductance score of the cluster. After all nodes in the previously constructed set have been processed, it is seen whether the conductance score of the cluster has changed. A change in the conductance score indicates that the member nodes of the cluster are altered. So the procedure is repeated again. If there is no change in the conductance score, it implies that no further improvement in the conductance score can be obtained with a single addition or deletion of a node(i.e., the local optimum is achieved). The algorithm is therefore stopped in this case. The pseudocode for IS^2 is given below.

procedure IS^2(seed,$G, Cond$)
$C \leftarrow seed$;
$w \leftarrow Cond(C)$;
increased $\leftarrow$ true;
while increased **do**
    $N \leftarrow C$;
    for all $v \in C$ **do**
        $N \leftarrow N \cup adj(v)$;
    for all $v \in N$ **do**
        if $v \in C$ **then**
            $C' \leftarrow C \backslash \{v\}$;
        else
            $C' \leftarrow C \cup \{v\}$;
        if $Cond(C') > Cond(C)$ **then**
            $C \leftarrow C'$;
    if $Cond(C) = w$ **then**
        increased $\leftarrow$ false;
    else
        $w \leftarrow Cond(C)$;
return C;

COMPLEXITY ANALYSIS

Runtime Analysis of LA

In the paper, it is proved that the runtime of LA is $O(|C||E|+|V|)$ i.e., it is bounded it terms of the number of output clusters. A slightly more intelligible version of the same proof is provided below.

Theorem *The runtime of* LA *is* $O(|C||E|+|V|)$.

Proof. Let C_i be the set of clusters before the ith iteration of the outer **for** loop and let v_i be the next node that will be processed. The total number of edges adjacent to v_i is $deg(v_i)$. Let the set of edges adjacent to v_i be $M(v_i)$. The edges in $M(v_i)$ fall into two classes: (1) the edges that are on the boundary of the cluster and (2) the edges that are not on the boundary of the cluster. Let x_i be the number of edges in $M(v_i)$ that are on the boundary of the cluster and $y_i = deg(v_i) - x_i$. Since we have to go to every edge in $M(v_i)$ to see if it is on the boundary of the cluster, the time it takes to calculate x_i and y_i is $deg(v_i)$. Once x_i and y_i are known, the conductance of the cluster with v_i added can be computed quickly (in $O(1)$) in the following way:

$$
\begin{aligned}
p_{c_j}{}' &= p_{c_j} - x_i + y_i \\
q_{c_j}{}' &= q_{c_j} + x_i \\
Cond(c_j \cup v_i) &= \frac{p_{c_j}{}'}{2q_{c_j}{}' + p_{c_j}{}'}
\end{aligned}
$$

where q_{c_j} is the total number of edges in c_j and p_{c_j} is the total number of edges on the boundary of c_j

Since there are $|C_i|$ number of clusters before the ith iteration, the runtime of the ith iteration is $O(|C||E|+|V|)$. The total runtime is therefore asymptotically of the order

$$
\begin{aligned}
&\sum_{deg(v_i)>0} |C_i| deg(v_i) + \sum_{deg(v_i)=0} 1 \\
&\leq \sum_{i=1}^{|V|} |C_i| deg(v_i) + \sum_{i=1}^{|V|} 1 \\
&\leq \sum_{i=1}^{|V|} |C| deg(v_i) + |V| \\
&= O(|C||E|+|V|).
\end{aligned}
$$

We have observed that instead of iterating over the entire set of clusters for every node, iterating over only those clusters to which the node is connected improves the overall runtime. Finding the clusters with which a node has connections is easy if the connected components of the graph are known beforehand. Neo4j (Kemper, 2015), the graph database used for this project, takes care of finding the connected components (neo4j uses the label propagation algorithm to find the connected components of a graph). The aysptotic time complexity of the algorithm however remains the same. Also note that the runtime reported in the results is the runtime for this slightly altered version of the algorithm.

Runtime Analysis of IS^2

In the original Iteractive Scan algorithm IS, the entire list of nodes is iterated over and over until the value of the scoring function cannot be improved. As the algorithm progresses, the clusters formed C_1, C_2, C_3.. are such that $Cond(C_1) > Cond(C_2) > Cond(C_3)$... Since the inequality is strict and the number of possible clusters is finite, the algorithm must terminate when started on any seed cluster.

The improved algorithm IS^2 does not iterate over the entire list of nodes following the observation that the nodes that are not directly connected to member nodes of the cluster never improve the conductance score of the cluster and can hence be skipped.

This change in the algorithm may decrease or increase the runtime depending on the graph. If the graph is sparse, then it is likely that cluster and it's neighbourhood are small and therefore, finding the neighbourhood will take relatively less time. If the graph is dense however and the cluster and its neighbourhood are large, then finding the neighbourhood will be expensive and the expense will likely be greater than the time saved as only a few nodes are skipped.

Space Complexity

In our implementation of the algorithms, the graph is stored in the neo4j graph database and the changes to the graph are done through Cypher queries. Since no space is allocated for the input graph in memory, the space complexity is essentially constant i.e., $O(k)$.

RESULTS

The algorithms were run on three datasets, each of different size : (1) The amazon large dataset (16,716 nodes), (2) The youtube medium dataset (5000 nodes), and (3) The DBLP small dataset (2500 nodes). The graphs were first given as input to LA and the resultant clusters were then given as input to IS^2 one at a time. The performance and goodness metrics obtained eventually are reported below.

Performance Metrics

Dataset	Precision	Recall	F-measure	NMI	Rand Index
amazon large	0.99956712	0.05453777	0.10343214	0.11099451	0.99742109
dblp small	0.96601833	0.79211732	0.87046737	0.85123805	0.99917663
youtube medium	0.88359732	0.5	0.63862318	0.79120808	0.99924729

Goodness Metrics

Dataset	Separability	Density	Cohesiveness	Clustering Coefficient
amazon large	3.88074105	0.77126923	0.61342734	0.61826524
dblp small	15.92314025	0.87566335	0.75937054	0.82425883
youtube medium	2.46755933	0.83259624	0.76103014	0.12185631

Time performance

Dataset	Runtime in seconds
amazon large	5440
dblp small	605
youtube medium	1021

The github repository for the project also contains detailed results and results for graphs of other sizes. It can be seen from the results that the algorithm takes seconds to execute even for large graphs. The performance of the algortihm however decreases with the size of the graph. This could possibly be due to the culling of very small clusters after the LA phase since a lot of clusters of that kind were produced for large graphs.

REFERENCES

Baumes, J., Goldberg, M., Krishnamoorthy, M., Magdon-Ismail, M., and Preston, N. (2005a). Finding communities by clustering a graph into overlapping subgraphs. pages 97–104.

Baumes, J., Goldberg, M., and Magdon-Ismail, M. (2005b). Efficient identification of overlapping communities. In Kantor, P., Muresan, G., Roberts, F., Zeng, D. D., Wang, F.-Y., Chen, H., and Merkle, R. C., editors, *Intelligence and Security Informatics*, pages 27–36, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kemper, C. (2015). *Beginning Neo4J*. Apress, Berkely, CA, USA, 1st edition.