CHAPTER 4

4. MULTI-LABEL PROPAGATION FOR OVERLAPPING COMMUNITY DETECTION

4.1 INTRODUCTION

Label Propagation algorithms that use single labels for propagation face the problem of random tie breaking when choosing a label among a set of labels for propagation. Due to the random tie breaking nature, techniques that follow single label for propagation produces non-deterministic outputs. This research work presents a multi-label propagation approach for community detection in complex networks, particularly, in the perspective of social networks and the work simulates the human pair wise communication behavior. The proposed work uses multi-label propagation technique to detect overlapping communities in social network. Designing a multi-label propagation algorithm to detect overlapping communities involves propagation of more than one label during the community detection process. The focus of this research work is based on the decision of choosing multiple labels for propagation and storing multiple labels received from the propagation process.

A multi-label propagation approach which is presented in this chapter, is a modification of Speaker-listener Label Propagation Algorithm (SLPA). The three important phases of the approach are: sending labels to other nodes, listening labels from other nodes for deciding which label to choose from the set of labels received from other nodes and updating the nodes. All the three phases relies on the propagation process in the network. Hence, first the propagation functionalities should be proposed to ensure that multiple labels are propagated between nodes in the network. Therefore, as a first step of the design, label propagation constraints are formulated. The nodes are categorized as speaker nodes and listener nodes and the constraints are imposed on the two categories. The research work treats each node in the network as both speaker and listener. When a specific node starts gathering the labels from the adjacent nodes, it acts
as an information consumer. Whereas, when it sends labels to the adjacent nodes it acts as an information provider. Until the listener asks for the labels from speaker, no labels are propagated from node to node. In this research work, each node broadcasts multiple labels to the neighbour nodes and at the same time receives multiple labels from its neighbours.

To start with, each node is assigned a unique label. During the iteration, every speaker sends multiple labels to the listener based on certain rules. Listener accepts a label or set of labels send by the speaker based on specified rules. Different algorithms use different strategies to propagate and update a label. In SLPA, at each iteration, a label that has a maximum probability is send by the speaker. The listener listens to all the labels send by the neighbour nodes, but, accepts only one label that has maximum number of occurrence. This research work aims to modify this particular process of speaking rule and listening rule by using multiple labels for speaking and listening. The approach of passing multiple labels between speaker and listener is the main difference between SLPA and the proposed algorithm. This system has been designed as a collection of agents that determine the following process:

1. Which labels should be sent to the neighbour nodes during the propagation?
2. Which labels are to be accepted from the received labels?
3. How to assign probabilities?
4. How to update the labels and stabilize it?
5. When to stop the propagation?

The critical issue related to the first three questions is how information should be maintained and it solely relies on the criteria formulated in the system. Depending upon the criterion chosen by the agents, labels propagate to form communities. This chapter discusses about agents designed for the system and elaborates the criteria used for identifying overlapping communities in social network. An important problem in label
propagation algorithms are its randomness in the output. Due to the random tie breaking strategies followed in various label propagation algorithms, every time when an algorithm is executed different outputs are produced. Though the number of communities may be approximately same in all the runs, the label of communities might change due to the dynamics in the criteria used by the algorithms. The objective of this research work is to design an algorithm that detects community overlap by avoiding the random tie breaking strategy followed in SLPA and produce deterministic communities. This work provides further understanding on dynamics of label propagation, in particular, how different propagation strategies can alter the dynamics of the process.

4.2 PROBLEM FORMULATION

4.2.1 Basic notations

An undirected, unweighted graph $G = (V, E)$ consisting of a vertex (node) set $V$ and edge set $E$ is represented as a social network. Let $n$ and $m$ be the number of nodes and edges in the network. $V$ is the set of $n$ nodes and $E$ is the set of $m$ connections. The notation $(u, v) \in E$ represents there is a link between two nodes $u$ and $v$. The graph is represented mathematically by an adjacency matrix $A$. In terms of social network with $n$ individuals, the social relationships between different individual is represented by the adjacency matrix. The degree of a node $i$ is defined as the number of adjacent nodes of $i$. For a node $i \in V$, let $|K|$ denote its degree.

A partition $P = \{P_1, P_2, P_3, ..., P_r\}$ is a division of network into disjoint communities, where $P$ denotes the partition composed of $n$ non-overlapping communities. In a pair of communities $P_i$ and $P_j$, if $P_i \cap P_j = 0$ $\forall i \neq j$ then it is said as disjoint communities. A cover or network community structure is represented as $C = \{C_1, C_2, ..., C_n\}$, i.e. a collection of subset of $V$ where each $C_i \in C$ and its subgraph form a community of $G$. If node belongs to more than one community such as $C_i$ and $C_j$, then there exists a pair of community $C_i$ and $C_j$ such that $C_i \cap C_j \neq 0$ $\forall i \neq j$. 

96
4.2.2 Problem definition

Given an undirected, unweighted network $N(n,m)$, where $n$ is the number of nodes and $m$ is the number of connections in the network, the objective of this work is to detect communities $C_1, C_2, C_3, ..., C_p$, where a node in the network $N$ belongs to more than one community. Initially, before the algorithm starts, the network $N$ is assumed to contain $n$ communities represented as $C = \{C_1, C_2, C_3, ..., C_n\}$. The objective of the label propagation algorithm is to reduce the number of communities to $p$ such that $p < n$ and form communities $C_1, C_2, C_3, ..., C_p$. The number of communities to be detected is not known or neither received as a parameter form the user which is the main characteristic of all label propagation algorithms.

Complex networks are normally huge in size and explicit information of their size and subgroups are not known in advance. Given a complex network the objective of a community detection problem is to find whether inherent similarity or an externally specified similarity among the nodes of the network exists or not. Hence, the goal of a community detection algorithm is to identify the number of communities in spite of lack of knowledge about the size of network. Based on the above assumption, the aim of this research work is to design and develop a label propagation algorithm that uses multiple labels for propagation as well as for updation to detect community overlaps in the network using only the network structure as its guide. An algorithm with near-linear time complexity and which produces deterministic communities that avoids the dynamics in communities is essential to detect overlapping communities.

The problem is based on the idea of label flooding algorithms that uses node identifiers of the network as labels and propagates labels step by step through the neighbours until it reaches the end of community. To aid the propagation process, two node agents namely listener and speaker are developed to propagate and receive labels based on the designed rules to form communities. Identifying overlapping communities through the process of spreading labels between nodes, without knowing the number of communities to be formed is the main objective of this research work.
4.3 MULTI-LABEL PROPAGATION ARCHITECTURE

The main contribution of this work is based on the process of propagating multiple labels at a time from node to node for identifying sub groups of the given network. The process involves interaction among agents based on certain rules for proper propagation of labels. The propagation is carried out to solve a problem, in other words to attain a specific goal. The goal of this research work is to detect overlapping communities. The work concentrates only on the concept of overlap and not on hierarchical communities in the detected community. To design the system, agents are defined with the roles to be played to carry out the process with the aim of detecting communities. Once agents are designed, various processes are formulated to detect communities from the network. The process involved in the designed system is pre-processing, propagation, updating and post-processing (Prabavathi and Thiagarasu 2014).

A node that listens to the labels of other adjacent nodes (i.e. neighbours of a node) in the network is treated as a listener. The set of neighbour nodes that propagates labels to the listening node is treated as speaker node. An agent that performs tasks such as treating a node as listener and invoking the communication process is called listening agent. The listening agent initiates the communication with the speaker node and invokes the propagation. An agent that resides in the speaker side, deciding the labels to be spread and start spreading the labels is called as speaker agent. In the proposed algorithm, during each iteration, a node is taken as a listener. The neighbours of listening node send one or more labels depending upon the probability of occurrence of that label. Hence, it differs from SLPA which sends only one label at each iteration. The listening node receives one or multiple labels from each speaker. But, in SLPA, a listener receives only one label from each speaker. Determining the order of the nodes for each iteration, is done randomly in the SLPA. In the proposed system, the order of nodes for propagation has been determined either through node identifiers or sorted nodes based on degree. There are two types of execution for label updating process: synchronous and asynchronous. In this research work, asynchronous updating has been used. The multi-label propagation architecture is represented as shown in Figure 4.1.
Figure 4.1 Multi-label Propagation Architecture
4.3.1 Pre-processing and Initialization

Pre-processing

First the nodes of a network are loaded into a data structure. Every node in the network is initially assigned with only one label name. If the loaded network does not contain self-loops, then self-loop is added to change a node as neighbour to itself. Making a node as neighbour to itself helps to send its own label during propagation process trying to form communities under its name. Taking into account one’s own label, makes the propagation process smooth as in SLPA and LabelRank algorithms (Xie et al. 2011b; Xie et al. 2013a). If the self-loops are added the node degrees of each node in the network should be calculated.

Normally, labels are not passed from a node to all the nodes in the network. It is passed only to the neighbouring nodes based on specified rules. The nodes to which the labels should be passed at the starting of the iteration are determined by node degree. A node degree indicates the number of nodes adjacent to it. Storing the neighbour nodes in a memory for each node of the network is important. Without knowing the neighbour nodes, the algorithm cannot propagate labels in the network. Hence, to reduce the time complexity of calculating neighbours every time when it is placed in the propagation process, the neighbour list is calculated before the propagation process.

Initialization

The process of label propagation starts from any one of the node in the network. The node which requests for labels from its neighbouring nodes is called listener node. All the neighbours of a listening node act as speaker nodes. The first phase of propagation starts from deciding which node is going to act as listener? When one of the nodes among $n$ nodes is selected as listener by the system, then all the neighbouring nodes start propagating labels to that listener. Updation of node labels takes place after the listener accepting the propagated labels. The important task is to decide which node is going to be the first listener among $n$ nodes, the second listener and so on.
In social networks, if a user has maximum friend list in a group and if that user posts a message, it will be viewed by maximum friends. If one or all listeners start to propagate the posted message, then, that particular message spreads more in the network. The label of a node which has maximum node degree, spreads its labels faster than the label of a node which has less node degree. In reality, a person adopting a new idea tries to follow a neighbour who has more connections to other neighbours because the neighbour who have more number of connections has higher number of potential sources of information (Xie and Szymaski 2011a). Based on the above idea, the order of propagation has been designed based on the node degrees. Various choices for determining the order of choosing the listener node are:

1. Randomly select a node as a listener.
2. Nodes are sorted in ascending order according to node labels (in case the labels are numbers). From the sorted list of labels, the order is determined.
3. The nodes are sorted according to its degree from high to low. A node that has maximum degree acts as the first listener, then the next node and so on.

In the designed system both second and third choice were followed while testing the algorithm. Every node stores the probabilities of labels what it observes as the propagation progresses. The nodes will not or cannot observe labels from all the nodes in the network. Initially, it listens to the labels sent by the neighbours only. The neighbour list of each node is designed either as a matrix representation or as a vector space. If the probabilities are initialized as matrix representation, space complexity is high, whereas vector space occupies less space. In this work, both designs were used for storing the probabilities. The following steps are designed for initializing label distribution memory:

1. Select a node in the network.
2. Calculate and assign initial probabilities for all the neighbours in the memory of the selected node (label distribution matrix or vector).
3. Repeat steps 1 to 3 for all the nodes in the network.
If the label distribution is treated as matrix representation, then in step 2, for all the neighbours of a node, its initial probabilities are calculated and initialized. For the nodes that are not neighbours, the probabilities are initialized as zero. Initializing zeros makes the probability distribution to be sparse as propagation progresses. This can be avoided through vector representation as iteration progress or from the beginning of the process itself. If label distribution memory is represented as vectors, then each element of the vector contains label names and their corresponding probabilities.

### 4.3.2 Rules for multi-label propagation

The main objective of a label propagation algorithm is to pass the labels from a node to its neighbouring nodes based on designed rules with the intention of finding communities. Initially, if there are $n$ number of nodes in the network, then, there are $n$ number of communities. A multi-label propagation algorithm aims to propagate multiple labels between nodes in the network and maintains more than one label for a node indicating its strength of membership to multiple communities. The propagation of multiple labels and retaining multiple labels for updating process is the major difference between SLPA and the proposed algorithm. The algorithm is comprised of three phases:
1. deciding which node is going to send the label
2. deciding which node is going to get the label and
3. how to update the label probabilities. The order of propagation is decided as designed in the initialization phase. Two agents, namely speaker and listener are designed for propagating labels. The received labels are not just updated immediately. An update condition is used in this system for deciding whether to accept the labels or retain the old labels. The functionalities of the communicating nodes are managed by following agents:

**Speaker**

The propagation starts with a speaker and listener node. A node acts both as speaker and listener depending on whether it is sending labels or receiving labels. In LPA, a node has only one label. Hence, it sends only that label name to its neighbour. In COPRA and SLPA, a node contains multiple labels as the intention of these algorithms is
to detect overlapping communities. Even though a node has multiple labels in its memory, SLPA sends only one label to its neighbour using random selection. Due to this random selection, the algorithm is non-deterministic in nature. In order to overcome this limitation, in this research work, two rules (Rule1 and Rule2) are designed for speaker agent. The main function of speaker agent is to decide which labels should be passed to the listening node, when listener agent waits for the labels from the speaker node. For deciding the labels speaker agent follows the rules:

**Rule 1:** If there are $l$ labels stored in the probability distribution memory of node $i$ and all labels have equal number of occurrence, i.e. no label has maximum frequency, then all the labels are sent to the listener.

**Rule 2:** If there are $l$ labels stored in the memory of $i$ and if $p, q$ are labels that occur equally maximum than other labels, then both the labels $p$ and $q$ are sent to the listener.

### Listener

The listener node is used to receive all the labels send by one or more speaker nodes and decide which labels to pass for updating process. The listener node holds as many labels as it likes depending on the underlying network structure and the knowledge accumulated repeatedly by observing the labels. It uses a memory to store all the received labels instead of erasing the previously stored labels. Each node has a memory and takes into account information that has been observed in the past to make current decisions. The more a node listens to a label, the more it spreads this label to other nodes. This nature is similar to that of people’s behavior of spreading most frequently discussed opinions in social network. Hence, this type of label spreading algorithms is most suitable for complex networks, especially for social network.

A label propagation process always starts with a listener. The core task performed by the listener agent is the decision making process of which labels are to be accepted and which labels are to be omitted? If a node $i$ contains $j$ neighbours and if the number
of labels send by $j$ nodes to node $i$ is stored in label list $l$, then, to determine what labels should be considered for updation, the following rules are designed:

**Rule 1:** If the label names in the list $l$ are unique, i.e. all labels occurs only once, then, all the labels in the list $l$ should be sent for updating process.

**Rule 2:** If labels in the list $l$ are not unique, i.e. some labels occurs less number of times and the remaining labels occur more number of times, then, the following rules are applied:

(i) if one specific label has maximum number of occurrence ($n$) times among $l$ labels and all other labels occur less than $n$ times, then only one label that have maximum occurrence should be considered for updating process and all others should be ignored.

(ii) Assuming $p$ and $q$ are two label names in the label set $l$, and if both $p$ and $q$ occur maximum number of times and the number of occurrence of other labels except $p$ and $q$ are less than this count, both labels $p$ and $q$ are passed for updating process.

### 4.3.3 Label updation and termination

Algorithms such as COPRA and LPA do not consider the labels that have been observed in the past to take current decisions. SLPA accumulates knowledge of repeatedly observed labels instead of erasing all but one of them using memory for storing the labels. When a listener accepts multiple labels from the neighbouring nodes, the next critical question is how to maintain the accepted labels. The issues are:

1. Whether the previously stored labels should be retained?
2. Whether totally to forget the previous knowledge and store only the new labels?
In the proposed design, the approach followed by SLPA is adapted. When new labels are propagated to the listener, the listener maintains all the new labels with the previously stored labels. This is determined using a probabilistic approach. The probability of all selected labels is updated in the corresponding memory of the node only after performing a conditional check. For performing the condition check, similarities between the node labels are used after every propagation. If the conditional check is satisfied, the node is updated. Due to this updating process, the probabilities of labels change during each iteration. Depending upon these updating process, at a certain stage the communities are formed. In order to contract the propagation process, inflation operator used in LabelRank algorithm is used in this work. Inflation is used to detect the communities quickly. The contraction is performed by increasing the probabilities of labels which have maximum probabilities and decreasing the probabilities of labels which do not have maximum probability. When the inflation operator is applied to the probability distribution matrix or vector, the labels which have higher probabilities get more preference than the labels with lower probability.

In LPA, each node holds only a single label that is updated adopting the majority labels in the neighbourhood at the end of the iteration. At the end of the propagation process, disjoint communities are formed. The stop criterion used in LPA is every node is assigned to the most popular label in its neighbourhood. In COPRA, each node is allowed to possess multiple labels. SLPA also follows the same procedure by retaining multiple labels at the end of iteration. In SLPA, during each iteration, a speaker sends only one label and listener accepts only one label from the collection of labels received from multiple speakers. SLPA continues till it collects sufficient information for post processing or $T$ iterations are reached.

Since the aim of this work is to detect overlapping communities, listener agents designed in this system accepts multiple labels from speaker and at the end of the propagation process; a node can hold multiple labels like COPRA and SLPA. The stop criterion of original LPA, does not suit for algorithms that follows multi-label propagation approach. A stop condition which is similar to that of LabelRank algorithm
using a similarity measure has been designed in this work. After an iteration is completed, a stop condition is checked. If the stop condition is true, then the iteration stops and to find the communities post-processing phase is to be performed. If the stop condition is false, the iteration continues until the specified number of times. If the predefined number of iteration is completed, then, post processing is performed or else, the propagation process continues until either the stop condition is satisfied or the maximum number of iterations specified is reached.

4.3.4 Post-processing

At the end of the iteration, only the label information that reflects the underlying network structure is present. The node labels are stored as probability representation of labels. The probability distribution denotes how strong a node belongs to a community. A threshold value is applied to the resultant probability of labels. The label of a node which is less than the threshold value is deleted from memory of that node. After the threshold procedure is over, the system groups the nodes having a particular label and forms a community. This is the only step performed in post-processing phase. Normally, after applying threshold value, a node may contain one label. If so, the system is said to detect disjoint communities. If a node contains more than one label after applying threshold value, then the node belongs to more than one community. Such nodes are called overlapping nodes and the resultant communities are said to be overlapping.

If the threshold value is too low, then the algorithm produces multiple communities. Conversely, if the threshold value is too high, communities may not be identified correctly. Post-processing refines the output in an acceptable manner. The work is designed such that when the label information is post-processed, community structure is detected. In algorithms such as COPRA, a node belonging to number of maximal memberships is controlled by the parameters. This system is designed such that the maximum number of membership of a node belonging to multiple communities is determined by the network structure and the rules designed. It does not depend on any parameter for restricting a node from belonging to multiple communities.
4.4 OVERALL SYSTEM DESIGN

**Input:** Network $N = (V, E)$

**Output:** Community structure $C = \{C_1, C_2, \ldots C_P\}$

- $T$: user defined maximum iteration
- $r$: threshold for post processing
- $q$: threshold for conditional update
- $i$: inflation operator

1. Give initial unique label names for each node (same as node $id$)
2. Add self loop to all the nodes in the network
   - Sort nodes in the order of the degree and store it in ordered vector
   - Calculate initial probabilities such that equal probabilities are assigned for each neighbour and store it in label distribution matrix
3. Repeat $T$ times or until stop criterion satisfied
   - All nodes in ordered vector are marked as unvisited
   - repeat
     - Select a unvisited node
     - Each neighbour of the selected node sends a label or multiple labels based on the probability of occurrence (frequency) of labels in the memory based on speaking rule
     - Listener accepts one or more labels based on listening rule for updating process
     - If the update condition is satisfied, listener updates the probability of one or more labels and the listener is marked as visited. Inflation operator is applied to increase the probabilities of labels having maximum occurrence
   - until all nodes in ordered vector are marked as visited
4. Post-process the labels in the memory based on the threshold $r$ and form the communities. If a node contains multiple labels, overlapping communities occur.
4.5 DISCUSSION

Given a network, the aim of overlapping community detection algorithm is to find sub groups of nodes, such that a node exists in more than one community. In social network, a community is a group of people who are more similar to each other within the group than people outside the group. Assuming this basic idea that communities are essentially local structures, an algorithm that uses the network structure alone to guide its process and requires neither parameters nor objective functions to detect communities is designed.

The single label propagation strategy used in SLPA has been modified to propagate multiple labels. COPRA uses multiple labels, but forces a belonging coefficient to limit the number of labels. This has been avoided in the designed system and it uses the features of SLPA and LabelRank algorithms. In the design, agents are utilized to propagate and accept multiple labels and determine the labels for updating process which is controlled by the formulated rules. Two agents are designed with rules that restrict the unwanted labels for propagation and updation. The designed system involves the coordination between speaker and listener agents for sharing the labels to form communities. The formulated rules help to avoid the non-deterministic communities and avoid the randomness in propagation process. The novelty of this approach is to consider not a single label for updating process but multiple labels based on update conditions.

The designed system does not ask for any parameters regarding the number of communities to be detected beforehand. The listener and speaker agents detect the number of communities in the network using the process of label propagation alone. The inflation operator is used to contract the propagation and reduce the size of label distribution vector and the updating condition is used to detect termination based on scarcity of changes in the network. The threshold for post-processing is used to remove the labels with low probabilities. The inflation, updating condition and threshold parameters help the label propagation algorithm to detect communities quickly.
Proper pre-processing helps to improve the detected quality of communities and initialization helps to reduce the time complexity of the algorithm. The conditional updating using similarity measures and inflation operators stabilizes the detected communities and avoid random outputs. Post-processing guides to group nodes with highest probabilities. The multi-label propagation algorithm designed in this research work, retains multiple labels in the memory which is the feature found in SLPA and uses the inflation, conditional update operators used in LABELRANK algorithm to detect overlapping communities. The designed system helps to identify overlapping nodes and overlapping communities in complex networks.