CHAPTER 3

3. LABEL PROPAGATION IN COMMUNITY DETECTION

3.1 INTRODUCTION

There exist various algorithms that identify community structures in large-scale real-world networks which were discussed in Chapter 2. Many of those algorithms require prior information about the number and size of communities. The size of the community is often not predictable beforehand in many complex networks. Particularly, in social networks where the size of the network is huge and heterogeneous in nature, it is impossible to predict about the number of communities initially. Moreover, it is computationally expensive to determine the community size. Hence, algorithms that are able to detect communities from complex networks without having any knowledge about the number of communities or the size of communities are essential. Label Propagation Algorithms (LPA) are such type of algorithms that detects community structure that uses network structure alone as its guide and requires neither a pre-defined objective function nor prior information about the communities.

The main idea behind LPA is to propagate the labels of node throughout the network using some technique and form communities through the process of label propagation itself. The idea of label flooding through a network originated from the L-shell method proposed by Bagrow and Bollt. (2005). The intuition of label flooding is that a single label can quickly become dominant in a densely connected group of vertices whereas it has trouble crossing a sparsely connected region. The labels are expected to be trapped inside a densely connected group of vertices. Based on the idea of local trapping of labels, the author proposed a local community detection method where a node is initialized with a label and propagates step by step through the neighbours until the end of the community is reached. The end of the community indicates the number of edges proceeding outward from the community and drops below a threshold value. LPA is a method which is effective, simple and near-linear time algorithm.

Reichardt and Bornholdt (2004) designed multi-state spin models where a spin is assigned to each node and applied to community detection. Q-state Potts model is one of the most popular model where q is the number of states that a spin may take, indicating the maximum number of communities (Reichardt and Bornholdt 2006). LPA is shown to be equivalent to a Potts model (Tibely and Kertesz 2008) and it belongs to the family of agent-based community detection algorithms. Label spreading is considered as a simplified but specific case of epidemic spreading of diseases where individuals are considered with infectious with their own unique disease. In the case of epidemic spreading, each person is infected by a disease that is spreading or existing in their neighbourhood. Similar to the process of disease spreading in epidemic network, label spreading can be treated in computer networks as community formation problem. As the spreading progresses, the number of labels decreases gradually and certain labels vanish due to domination of other labels. At the end of the propagation process, few labels exist and dominate the whole network and thus forming communities. The major limitation of label propagation algorithms is its epidemic nature, i.e. during the process of propagation certain labels tend to form giant communities dominating all other labels.

LPA may be considered as a simple opinion spreading model but with many competing opinions. During the opinion spreading process, each node adopts an opinion through an artificial unique identifier in agreement with the majority of its neighbours. At the end of the opinion spreading process, connected nodes with the same identifier form a community. In social network, a user can view, spread, accept or reject an opinion received from its neighbours. Labels of each node are treated as opinion in the LPA. Propagating the labels is similar to opinion spreading in social network. Opinions are chosen based on the opinions of the neighbour list, which in turn, form their opinions from their neighbours and so on. This resembles the characteristics of social network where a user adds the friend's friend into their friends list.

LPA algorithm provides desirable qualities such as easy implementation, fast execution and more suitable for the analysis of large systems. Due to the fast execution, label propagation techniques are considered as more suitable for the analysis of large

systems. In this chapter, various label propagation algorithms (Raghavan et al. 2007; Leung et al. 2009; Xie et al. 2011b and 2013a) that are capable to detect disjoint and overlapping communities are discussed.

3.2 LABEL PROPAGATION

Many community detection algorithms have been proposed and utilized with varying degrees of effectiveness in the community detection research. LPA which was first proposed by Raghavan et al**.** (2007) uses unique identifiers of nodes as labels and propagates the labels based on an agreement with the majority of the neighbour nodes and each node selects a label from its neighbourhood to adopt it as its label. LPA works as follows: Node x has neighbours $x_1, x_2 x_3, \ldots x_n$ and each neighbour carries a label denoting the community to which they belong to. Each node in the network chooses to join the community to which the maximum number of its neighbours belongs to, with ties broken uniformly and randomly. At the beginning, every node is initialized with unique label (called as identifier) and the labels propagate through the network. At every step of propagation, each node updates its label based on the labels of its neighbours. As labels propagate, densely connected groups of nodes quickly reach a consensus on a unique label as shown in Figure 3.1. At end of the propagation, most labels disappear and only few labels exist in the network. Label propagation algorithm reaches convergence when each node has the majority label of its neighbours. At the end of the convergence, nodes connected with the same label form a community. According to LPA, communities are defined as vertices having identical labels at convergence. The agreement between the nodes to spread the labels forms the basis for Label propagation algorithms.

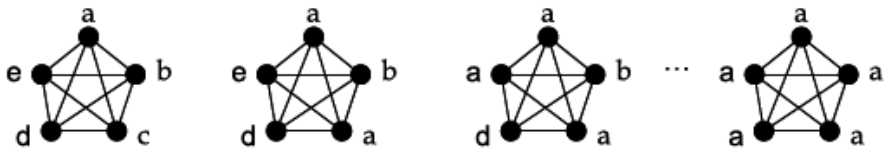

Figure 3.1 Node labels are updated from left to right (Raghavan et al. 2007)

The updating of labels can be done either synchronously or asynchronously while the label propagation progresses. In the process of label propagation, at an instance there exists some nodes in the network that have undergone iteration and the remaining nodes may not have undergone the iteration. In synchronous updating, a node x at t^{th} iteration updates its label based on the labels of its neighbours at iteration $t-1$. Hence, $C_x(t) = f\left(C_{x1}(t-1), C_{x2}(t-1), ...C_{xk}(t-1)\right)$ where $C_x(t)$ is the label of node x at time t. In asynchronous updating, a node at t^{th} iteration, updates its label based on the labels of its neighbours at iteration t as well as $t-1$. The new labels of the nodes which have undergone the updation during the current iteration t are taken and for the nodes which have not undergone the updation during the current iteration t, the labels at $(t-1)^{th}$ iteration are taken. Hence, asynchronous updating can be represented as $\mathrm{C_x(t) = f\left(C_{x_i 1}(t), ... C_{x_i m}(t),\ C_{x_i (m+1)}(t-1), ... C_{x_i k}(t-1)\right)}$ where $x_{i1.....}x_{im}$ are neighbours of x that have already been updated in the current iteration while $x_{i(m+1)}....x_{ik}$ are neighbours that are not yet updated in the current iteration.

The order of nodes to be updated at each iteration is randomly chosen. At the beginning of the algorithm, there are n different labels (where n is the number of nodes in the network) and as iteration progresses the number of labels reduces. The iteration should be performed until no node in the network changes its labels or until a stop condition is satisfied. At the end of the iteration, there are as many labels as there are communities and the LPA algorithm is shown in Figure 3.2.

LPA belongs to the family of agent based community detection algorithms. Due to the random tie breaking strategy, the algorithm produces different solutions at different runs. The stop criterion of this algorithm is similar to the definition of strong communities proposed by Radicchi et al. (2004). Each node in strong communities are expected to have strictly more neighbours within its community than outside, the communities obtained by the label propagation process require each node to have at least as many neighbours within its community as it has with each of the other communities.

The authors pointed out that even though they obtained different community structures at different runs, they were similar to each other. It is difficult to pick one solution as the best one among several different results, because one solution may be able to identify a community that was not discovered in the other and vice-versa. Hence, an aggregate of all the different solutions were found that provides a community structure containing the most useful information.

1. Initialize the labels at all nodes in the network. For a given node x, $C_x(0) = x$
2. Set $t = 1$.
3. Arrange the nodes in the network in a random order and set it to X .
4. For each $x \in X$ chosen in a specific order, let

$$C_x(t) = f\left(C_{x_i 1}(t), \ldots, C_{x_i m}(t), C_{x_i (m+1)}(t-1), \ldots, C_{x_i k}(t-1)\right)$$

   f returns the label occurring with the highest frequency among neighbours and ties are broken uniformly randomly.
5. If every node has a label that the maximum number of their neighbours has, then stop the algorithm, else, set $t = t+1$ and go to step 3.

Figure 3.2 Label Propagation Algorithm (Raghavan et al. 2007)

Since LPA algorithms are considered as fast and efficient, much research work has been progressed in the last decade. A major advantage of this algorithm in comparison with most other graph clustering algorithms is that it totally relies only on local information that can be quickly computed. Each node makes its own decision regarding the community to which it belongs to based on the communities of its immediate neighbours. Raghavan et al. (2007) found that irrespective of the number of nodes, 95% of the nodes are classified correctly by the end of the 5th iteration. When the algorithm terminates it is possible that two or more disconnected groups of nodes have the same label because two or more neighbours of a node receive its label and pass the labels in different directions which ultimately leads to different communities adapting the

same label. To separate this disconnected components, a simple breath-first search on the sub-networks was run after the termination of algorithm.

Raghavan et al. (2007) pointed out that if the set of nodes in the network that are likely to act as centers of attraction for their respective communities are known, then it would be sufficient to initialize such nodes with unique labels, leaving the remaining nodes unlabeled. If this algorithm is applied after identifying center nodes, then, the unlabeled nodes will have a tendency to acquire labels from their closest attractor and join that community. Also, restricting the set of nodes initialized with labels will reduce the range of possible solutions that the algorithm can produce. Since identifying such center nodes are difficult before identifying the community itself, in this algorithm all nodes are given equal importance and provided with unique labels before iteration starts.

The main advantage of LPA is its near-linear time complexity i.e. it runs linearly in the number of edges, thus linearly also in the number of nodes for sparse network. The number of iterations to converge appears independent of graph size or growing very slowly with it. Initializing every node with unique labels require $O(n)$ time. Each iteration takes linear time in the number of edges $O(m)$ and the time for processing disconnected communities is $O(m+n)$. As the number of iteration increases, the number of nodes that are classified correctly increases. This algorithm formed the basis for all the succeeding label propagation algorithms discussed in the next section.

3.3 RELATED WORK ON LABEL PROPAGATION

Numerous algorithms have been developed using label propagation techniques to detect disjoint communities (Raghavan et al. 2007; Leung et al. 2009; Subelj and Bajac 2011; Xie et al. 2013a) as well as for overlapping communities (Gregory 2010; Xie et al. 2011b; Wu et al. 2012). Many of these algorithms attempt to detect either disjoint or overlapping communities from static networks. This research work focuses only on static networks. Hence, some of the important algorithms that use labels for

communication between networks to form communities in static network are discussed in this Section.

3.3.1 Disjoint communities

In LPA proposed by Raghavan et al. (2007), a label can spread too far from its origin and produce a single large community (monster community) when applied to large web networks. Monster communities occupy more than half the nodes of the network. Leung (Leung et al. 2009) focused mainly on the problem of monster communities on large web networks and significantly improved the performance of community detection using label hop attenuation technique. This technique prevents the label from spreading too far from its origin. Each label l_n is associated with additional score s_n that is initially set to the value of 1 and decreases after each propagation as shown below.

$$s_n = \left(\max_{i \in N^{cn}(n)} s_i \right) - \delta$$

The attenuation ratio is represented as δ. When s_n reaches 0, the label stops propagating, thus eliminating the formation of a single large community. With hop attenuation the authors coupled node preference f_n which represents the node propagation strength, to achieve better performance of the LPA. The modified label propagating updating rule is reformulated as

$$C_n = \arg\max_{l} \sum_{i \in N^{l}(n)} f_i^{\alpha} s_i w_{ni}$$

where w_{ni} is the edge weight (equal to 1 for unweighted graphs) and □ is a parameter of the algorithm. Hop attenuation has proven to be a reliable technique for preventing monster communities. It has also been observed that hop attention may damage reasonable large communities since it limits the radius of community structure. Predicting the value of attenuation ration δ has been the main issue of the algorithm proposed by Leung et al. (2009). The authors used the values around 0.10 for δ and obtained good

result. The authors have also observed that large values of δ may prevent the natural growth of communities and have proposed a dynamic strategy that decreases δ. In the early iterations of the algorithm, large values of δ prevent a single label propagating too far to form strong communities. The value of δ is then decreased, to gradually relax the restriction and to allow formation of the actual communities that exists in the network topology. From the experiments on real-world networks, the authors proved that such a strategy has very good performance on larger networks.

Subelj and Bajec (2011) advanced the algorithm developed by Leung et al. (2009) and proposed different dynamic hop attenuation strategies, based on the hypothesis that hop attenuation should only be employed, when a community or a set of communities, is rapidly occupying a large portion of the network. Otherwise the labels should be allowed to reach the equilibrium unrestrained. This approach would retain the dynamics of label propagation and still prevent the emergence of a major community. The authors considered several strategies for detecting the emergence of a large community or a set of large communities and developed two unique strategies of community formation namely defensive preservation of communities where preference is given to the nodes in the core of each community and offensive expansion of communities where preference is given to the border nodes of each community. The authors proposed an advanced label propagation algorithm: the diffusion and propagation algorithm (DPA) that combines the two strategies in a hierarchical manner. The algorithm first applies the defensive strategy to the original network to produce large number of smaller communities (i.e. initial estimates of the communities). It then applies propagation strength to the core of each community, i.e. $f_n^{\alpha} = p_n$. Thus the updating rule

$$c_n = \arg\max_l \sum_{i \in N^l(n)} f_i^{\alpha} s_i w_{ni}$$

proposed by Leung et al. (2009) is modified as

$$c_n = \arg\max_l \sum_{i \in N^l(n))} p_i s_i w_{ni}$$

The labels of all the border nodes of a community are relabeled such that approximately one half of the nodes retain their original label. Then, the offensive strategy is applied on the constructed community network to extract the core of the network to form the whisker communities. The offensive strategy applies preference to the border of each community, i.e. $f_n^{\alpha} = 1 - p_n$ and the updating rule is modified as:

$$c_n = \arg\max_{l} \sum_{i \in N^l(n))} (1 - p_i) s_i w_{ni}$$

The above two strategies are recursively applied only to the core of the community, when the whisker communities are retained as identified communities. The recursion process is repeated until the extracted core contains all of the nodes of the network analyzed. DPA employs only local measures for community detection and does not require the number of communities to be specified beforehand. During each iteration of the algorithm, each edge of the network is visited (at most) twice. Thus the time complexity of a single iteration equals $O(m)$, with m being the number of edges.

In Label propagation algorithm proposed by Raghavan et al. (2007), at the early stage of the propagation process, the fraction of attempted updates that result in changes to new labels is high because most nodes are in a very diverse neighbourhood. But, after few iterations, the competition between communities is restricted only to their boundaries. Updates are unnecessary for nodes inside the community (passive nodes) as they do not change their labels after few iterations. The unnecessary updates of labels that fail to change their labels require additional time. So, the final convergence of LPA is delayed. This time delay can be easily reduced by book keeping the information about the boundaries of the currently existing communities. In order to improve the execution time of LPA, Xie and Szymanski (2011a) proposed an update rule based on neighbourhood strength driven label propagation. The basic idea of the improvement is to avoid unnecessary updates at each iteration of the algorithm, while maintaining the overall behavior of the algorithm and achieve higher speed of execution to improve the quality of the community detection. By storing the information about the community boundaries and avoiding re-calculation, neighbourhood strength driven label propagation algorithm

reduced the time complexity of LPA. The improvement does not require any threshold value or modification of the stop criterion.

The authors referred nodes whose neighbours have the same label as interior nodes and nodes that are not interior as boundary nodes. Nodes that attempt to change their labels during updating process were called as active nodes and nodes that do not change their labels were said as passive nodes. A node can be in three states: passive interior, passive boundary, active boundary. A boundary node could be either passive or active depending on its neighbourhood. The updating rule itself becomes a natural end of execution condition. The algorithm stops when every node becomes passive (i.e. convergence of a network) and detects only disjoint communities. The algorithm is shown in Figure 3.3.

1. At time t=0, construct the active node list containing all the nodes.
2. Randomly pick an active node, say i, from the list and attempt to adopt a new label according to the update rule. Since only active nodes are placed initially on the list and they remain on the list as long as they are active, each node selected for an update will change its label during the update.
3. First, check if the updated node became passive and if so, remove it from the list. Next, check all its neighbours for the change of status in the following three steps.
   (a) If an interior neighbour became an active boundary node, add it to the active node list.
   (b) Remove any previously active neighbour that became passive from the active node list.
   (c) Add any previously passive boundary neighbour that became active to the active node list.
4. If the active node list is empty, stop; otherwise, increase time *t* by one unit and go to step 2.

Figure 3.3 Neighbourhood strength driven label propagation (Xie et al. 2011a)

The complexity of the improved algorithm is unchanged. Initialization of the active node list requires $O(n)$ time. For randomly selecting a node $O(1)$ time is required. Updating the node i and its neighbours requires $O(d_i)$, where d_i is the degree of node i. By using the active node list, evaluating the convergence of the whole network is easy and takes exactly $O(1)$ by checking if the list is empty. The author proved that the number of iterations needed for the algorithm to converge is equal to the total number of effective updates.

Xie and Szymanski (2013a) proposed a stabilized LPA by introducing a set of operators to control and stabilize the propagation dynamics. The authors stabilized LPA and extended Markov Cluster Algorithm (MCL) approach that resulted in LabelRank algorithm. It stores, propagates and ranks labels in each node and relies on four operators namely propagation, inflation, cutoff and conditional update to stabilize the propagation dynamics. In this algorithm, an entire distribution of labels is maintained using n $1\times n$ vectors (n is the number of nodes in the network). The probability distribution $P_i(c)$ contains the label probabilities at each iteration and is different from adjacency matrix A defining the network structure. Each element of the probability matrix holds the current estimation of probability of node i observing label $c \in C$ taken from a finite set C. Initially, the probability matrix of the network is represented as

$$P_i(c) = 1/k_i \forall j \quad \text{s.t.} \quad A_{ij} = 1$$

During the propagation process each node broadcasts the labels to its neighbours at each time step and computes the new distribution $P_i(c)$ simultaneously as

$$P_j^{'}(c) = \sum_{j \in Nb(i)} P_j(c)/k_i, \forall c \in C$$

where $Nb(i)$ is a set of neighbours of node i and $k_i = |Nb(i)|$ is the number of neighbours. In matrix form this operator can be expressed as $A*P$ where A is the $n\times n$

adjacency matrix and P is the $n \times n$ label distribution matrix. Since each node keeps multiple labels received from its neighbours, LabelRank eliminates the need of tie breaking problem as in LPA and COPRA. The nodes with same highest probability are grouped as communities.

After computing the label probabilities, LabelRank uses inflation operator Γ_{in} on P to contract the propagation where in is the parameter taking real values. In MCL, the inflation is applied to adjacency matrix. But, in LabelRank, the inflation operator is applied to the label distribution matrix P (instead of a stochastic matrix or adjacency matrix) to decouple it from the network structure. After applying inflation operator

$$\Gamma_{in} P(c) = P_i(c)^{in} / \sum_{j \in C} P_i(j)^{in}$$

each value in $P_i(c)$ changes to $P_i(c)^{in}$. This operator increases the probabilities of labels that were assigned high probability during propagation and decrease the probabilities of labels that were assigned low probability during propagation. To reduce the memory utilization, a cutoff operator is applied to P to remove labels that are below threshold value. Inflation helps to decrease probabilities of labels to which propagation assigned low probability and cutoff efficiently reduces the space complexity, from quadratic to linear.

Due to cutoff and inflation operators, the number of labels in each node monotonically decreases and drops to a small constant within few steps. Since, LabelRank detects the highest quality communities far before convergence, if the iteration continues after convergence, the quality of the detected community decreases. Even after using inflation and cutoff operator, there is no guarantee that the algorithm performs efficiently because the updating process can be unnecessarily performed. Hence, a novel solution based on the conditional update operator is used to update a node when it is significantly different from its neighbours in terms of labels. Conditional update preserves detected communities and detects termination based on scarcity of

changes to the network. At the end of each iteration, the change is accepted only by nodes that satisfy the following update condition.

$$\sum_{j \in Nb(i)} isSubset\left(C_i^*, C_j^*\right) \leq q\kappa_i$$

In the condition mentioned above, C_i^* and C_j^* are the set of maximal labels that contain maximum probabilities at node i at previous time step and q is a parameter for conditional stop. Introduction of the new operator, conditional rule and stopping criteria preserves the speed of the LPA based algorithms. The output is deterministic because there is no randomness in the simulation. The operators resolve the randomness issue that is present in traditional LPA, by stabilizing the discovered communities in all runs of the same network. The LabelRank algorithm is shown in Figure 3.4.

The running time of LabelRank is $O(m)$, linear with the number of edges m. Adding selfloop takes $O(n)$, initialization of P takes $O(m)$, each of the four operators takes $O(m)$ on average and the number of iterations is usually $O(1)$. The author suggested that such type of stabilization as used in this algorithm is important for the class of all label propagation algorithms. Because, without stabilization, these type of algorithms cannot be applied to dynamic networks.

1. add selfloop to adjacency matrix A
2. initialize label distribution P
3. repeat

$$P' = A \times P$$
$$P' = \Gamma_{in} P'$$
$$P' = \Phi_r P'$$
$$P = \Theta_q(P', P)$$

until stop criterion satisfied

4. output communities based on P

Figure 3.4 Overall procedure of LabelRank Algorithm (Xie et al. 2013a)

3.3.2 Overlapping communities

Community Overlap Label PRopagation Algorithm (COPRA) is the first algorithm proposed to detect overlapping community using label propagation which is very fast and allows each node to belong to multiple communities at a time (Gregory 2010). In Label Propagation algorithm designed by Raghavan et al. (2007), a node contain single label to which it belongs to. To find overlapping communities, a node or multiple nodes in the network should contain more than one label. In COPRA, each node x is labeled with a set of pairs (c,b) where c is a community identifier and b is the belonging coefficient that indicates the strength of x's membership of community c such that all belonging coefficient for x sum to 1. During the label propagation process, label of x is set to the union of its neighbours label, belonging coefficients of the communities over all neighbours is summed and normalized. The label of a vertex at iteration t is always based on its neighbours label at iteration $t-1$. To retain more than one community identifier in each label, the algorithm deletes the pairs whose belonging coefficient is less than a threshold value $(1/v)$, where v is the parameter of the algorithm. If all pairs in a vertex label have a belonging coefficient less than the threshold, a pair that has the greatest belonging coefficient is retained and all others are deleted. If more than one pair has the same belonging coefficient below the threshold, then a random pair is selected.

Gregory (2010) suggested that using synchronous updating, results proved to be better than asynchronous updating method followed by Raghavan et al. (2007) and Leung et al. (2009). In order to prevent community identifier propagating too far and forming monster communities, after a propagating step, a node is checked whether it is still among the community identifier labeling node x. If so, the label of x is set to x only and all other community identifiers are deleted. This technique proved to improve the worst results because label x appears on x more than once and reduces the size of other communities because other communities no longer include x. Though, this technique was successful in improving the worst result, best results were also worsened due to the limits

imposed on community size. At the end of the propagation, each vertex whose labels contain community identifier c is placed in community c. This method might form communities that are subset of others. COPRA avoids this by keeping track of the subset relationships between communities.

Similar to LPA, the communities may not be connected at the end in COPRA. So, all disconnected communities are split into smaller connected ones. A novel termination condition is used to terminate the algorithm with the good solution. For low overlapping density networks, COPRA offers better performance. But, it produces a number of small size communities in certain networks due to the global vertex-independent parameter. Even though the algorithm produces different solutions each time it is run, all the solutions are good ones. If the value of the parameter v is greater, the algorithm deletes greater overlaps between communities. If the value of the v increases too much, community identifier starts propagating too far and the algorithm produces worst results. The execution time is lightly more than the linear time followed by LPA, but faster than CFinder and CONGO. According to the comparative analysis of various algorithms on benchmarks of Lancichinetti and Fortunato (2009a), COPRA has been marked as the most effective method once fed with correct parameters. The main procedure of COPRA is given in Figure 3.5.

1. To initialize, every vertex is given a unique label with belonging coefficient setting to 1.
2. Repeatedly, each vertex x updates its labels by summing and normalizing the belonging coefficients of vertices in the neighbor set of x. To avoid every vertex owning all community identifiers at the end, parameter v is used to limit the maximum number of communities to which any vertex can belong. Propagation stops if stop criterion is satisfied.
3. Remove communities that are totally contained by others.
4. Split discontinuous communities.

Figure 3.5 Overall procedure of COPRA (Gregory 2010)

If n is the number of vertices and m is the number of edges and v is the parameter (maximum number of communities per vertex), then the time complexity of COPRA in each phase is ①Initialization takes time $O(n)$ ② For constructing an updated label for each of the n vertices is $O(vm\log(vm/n))$ ③For iterating through all community identifiers of each of the n vertices the total time is $O(vn)$. For repeated iteration, time per iteration is $O(v(m+n)+v^3n))$ and the initial and final steps take time $O(v(m+n)+v^3n))$.

Xie (Xie et al. 2011b) proposed a fast Speaker-listener Label Propagation Algorithm (SLPA) which spreads labels according to dynamic interaction rules and maintains label distributions in the memory of each node. In SLPA, each node can be a speaker or a listener depending on whether it serves as an information provider or consumer. A node stores as many labels as it likes, based on the propagation experience in the stochastic processes which is driven by the underlying network structure. The more a node observes a label the more likely it will spread this label to other nodes. This is similar to mimicking people's preference of spreading most discussed opinions. The SLPA algorithm is shown in Figure 3.6. The significant characteristics of SLPA are: 1. a node accumulates knowledge of repeatedly observed labels instead of erasing all but one of them 2. no knowledge about the number of communities is required.

SLPA follows asynchronous updating scheme for updating the labels in the memory. The noteworthy feature of SLPA is memory for each node takes into account information that has been observed in the *past* to make current decision. This feature is not followed in other label propagation algorithms like LPA and CORPA where a node updates its label completely forgetting the old knowledge. This feature combines the accuracy of the asynchronous update with the stability of the synchronous update due to which the fragmentation issues (producing a number of small size communities) that is present in COPRA is avoided in SLPA. The algorithm simply stops when the predefined maximum number of iterations T is reached. In general, SLPA produces relatively stable outputs, independent of network size or structure, when T is greater than 20.

The SLPA identifies both node and community level overlapping structures. It is suitable for weighted, unweighted, directed and undirected networks. But, due to random tie breaking strategy, it produces different partitions in different runs, which is not desirable in applications like tracking the evolution of communities in a dynamic network. The initialization of labels requires $O(n)$ time where n is the total number of nodes. The outer loop is controlled by the user defined maximum iteration T. The inner loop is controlled by n. One speaking rule and one listening rule is followed in inner loop. For the speaking rule, selecting an element from the array requires $O(1)$ operation. For listening rule, since the listener needs to check all the labels from its neighbours, it takes $O(k)$ on average, where k is the average degree. The complexity of the dynamic evolution for the asynchronous update is $O(Tm)$ and $O(Tn)$ on an arbitrary network and sparse network respectively. Since each node has a memory of size T, in the post-processing, the thresholding operation requires $O(Tn)$ operations. The time complexity of the entire algorithm is $O(Tn)$ in sparse networks.

1. The memory of each node is initialized with this node's id (unique label).
2. Following steps are repeated until the stop criterion is satisfied:
   a. One unvisited node is randomly selected as a listener.
   b. Each neighbor of the selected node sends out a *single* label following certain *speaking rule*, such as selecting a random label from its memory with probability proportional to the occurrence frequency of this label in the memory.
   c. The listener accepts *one* label from the collection of labels received from neighbors following certain *listening rule*, such as selecting the most popular label from what it observed in the *current* step. The listener adds the most popular label received to its memory.
   d. Mark the listener visited.
3. Post-processing is applied in the memories and the threshold r is applied to output the communities.

Figure 3.6 Speaker-listener Label Propagation Algorithm (Xie et al. 2011b)

In COPRA, as v is vertex independent, if there are few vertices with a small number of community memberships and others with a large number of community memberships, it is hard to choose a suitable v to satisfy both kinds of vertices at the same time. To solve this problem. Wu et al. (2012) proposed a new balanced multi-label propagation algorithm (BMLPA) using a balanced belonging coefficients update strategy with vertex dependent parameter to detect overlapping communities in networks with various numbers of community memberships which cannot be solved well by COPRA. In COPRA, each vertex has at most v labels, while the balanced belonging coefficients (BBC) update strategy proposed by the author does not limit the number of communities that a vertex v may belong to. The BBC allows a vertex to have balanced belonging coefficients and belongs to any number of communities without a global limit on the largest number of community memberships as in COPRA.

A Rough Core (RC) extraction method is designed in BMLPA to initialize labels before label propagation. As many maximal cliques that highly overlap only indicate the same core, the problem is for a core indicated by more than one clique. Since monster communities can be formed from larger cliques, the author suggested that smaller cliques produce better final results. The rough core indicates the smallest maximal clique starting from two first chosen vertices. The author proved that RC initialization and the BBC update strategy can bring improvements in both quality and stability, especially for networks that contain vertices with various numbers of community memberships. Assuming n as the number of nodes, m as the number of edges, v_{avg} as the average number of labels to which each vertex belong and d_{avg} as the average degree, then the time complexity of BMLPA is: for initialization $O(n(\log n + v_{avg} d_{avg}))$, for propagation $O(v_{avg} m \log(v_{avg} m / n))$ with total time complexity of $O(n \log n)$ per iteration.

Lin et al. (2012) improved the LPA algorithm by using a node score which uncovers hub preference to detect overlapping community structure in complex network. In this algorithm, community sizes were considered into weights of edges avoiding monster communities. Raghavan et al. (2007) stated that LPA could identify hub nodes as those which kept oscillating among several communities. However this identification

process will lead to huge computational cost and unable to uncover the hub's preference. To solve this issue, the author observed the hub nodes among communities and found that hub nodes connect different communities with approximate weight and proposed a measurement D_x to present the hub preference of node i as follows:

$$Dx(i) = \frac{\sum_{k \in l_i} \left(W_i(k) - \overline{W}_i \right)^2}{n}$$

where $\overline{W_i} = \frac{\sum_{k \in l_i} W_i(k)}{n}$ is the average of $W_i(k)$ and n is the number of elements in l_i. The smaller $Dx(i)$ is, the more probable that node i is hub node. When, $Dx(i) = 0$ it means node i equally connects with all the communities. The value of threshold δ uncovers the scope of overlapping in complex network. The author modified LPA using $Dx(i)$ and δ to detect overlapping communities. During every updation, only nodes whose measurement Dx larger than δ are allowed to be infected by their neighbours. In asynchronous implementation of the LPA algorithm, each node calculates Dx in propagation step, so the modification does not increase computing time complexity. If N *is* the number of nodes in complex network and K is number of iteration, the computational cost of the improved algorithm is $O(Kn)$ which is also near linear time. In this algorithm, the nodes are sorted according to smaller values of Dx in each update step in order to further improve computational cost of the proposed algorithm. Moreover, the community size was taken into consideration for the process of generating node label to prevent monster community structure. The criterion followed for node updation in the algorithm is

$$L_i^{'} = \arg\max_K \frac{W_i(k)}{|K|^{\frac{1}{2}}} K \in l_i$$

which prevents monster community structure. Figure 3.7 lists the steps involved in the above mentioned algorithm.

1. Initialize each node in complex network with a unique label and then calculate the measurement Dx. Sort all the nodes according to the value of $Dx\,x$ in descending order and then put the sorted nodes into queue Q.
2. For each node v in Q, update its label, when $Dx(v) \geq \delta$. The parameter δ which decides the scope of overlapping is set. Resort all nodes according to Dx and put them into Q.
3. When none node updates its label or the iteration time is greater than a given threshold stop the algorithm. Otherwise, go to step 2.

Figure 3.7 Label Propagation Algorithm using hub preference (Lin et al. 2012)

Wei Hu (2013) proposed a novel Weighted Label Propagation Algorithm (WLPA), as an extension of SLPA, by introducing a similarity between any two vertices in a network based on the labels each vertex has received during label propagation and using this similarity as a weight of the edge between the two vertices in the next iteration of label propagation. Further, weights are added to an edge of two vertices that could be repeated in multiple label propagations, making it possible to refine the communities found in each repetition. SLPA outperforms the original LPA with extra information that is the array of labels. To further improve SLPA, a Wei Hu et al. (2013) introduced weight similarities as additional information to SLPA. The similarity of any vertices in a network is calculated as a weight of edge between the two vertices and this is passed to SLPA. The overview of WLPA algorithm is shown in Figure 3.8.

1. Run the steps 1 and 2 of SPLA algorithm.
2. Use the last 20% of the labels each vertex has received to compute the similarity between any two vertices.
3. Run SPLA using this similarity as a weight of the edge between the two vertices.

Figure 3.8 Weighted Label Propagation Algorithm (WLPA) (Wei Hu 2013)

This chapter summarized an overview of various algorithms such as LPA, DPA, COPRA, SLPA, WLPA and BMLPA that detects disjoint and overlapping communities in static network using the label propagation techniques. A recent review on overlapping community detection algorithms in static networks and that can be extended to dynamic networks is listed (Prabavathi and Thiagarasu, 2013c).

3.4 DISCUSSION

The problem of finding communities in complex networks using label propagation algorithms has been popular among researchers witnessed by an impressive number of valid works in this field. The important characteristics of Label propagation technique are it requires only local knowledge for detecting communities and it's near linear time complexity which makes the algorithm desirable for large networks. This research work has been motivated by the simplicity of label propagation approach and its efficient time complexity. Most of the overlapping community algorithms found in the research are extensions of disjoint community detection algorithms. Hence, a detailed background study on important disjoint community detection algorithms based on label propagation has been made. Recently, some researches consider the concept of overlap using label propagation techniques. A detailed study on various overlapping community detection algorithms that uses label propagation techniques has been made and their features, time complexity and limitations have been understood. The limitations of the listed algorithms were thoroughly analyzed to have a clear understanding of the problems of existing algorithms. Researchers pointed out that using only the network structure to detect communities without any requirement of external parameters for running the algorithm as the interesting feature of LPA and the dynamics of label propagation and formation of monster communities as the main limitations of LPA.

From the above review, it has been understood that label propagation algorithms are faster and lot of advances have been made in the technique of propagating labels. Most of the literature on label propagation techniques focuses on propagating single label. There are few algorithms that use multiple labels for propagation. In such case, the goodness of the algorithm depends upon the technique of how few labels are omitted

from updating from a group of labels and how other labels are considered for updation. This decision varies depending upon the criteria used in different algorithms. Since, it is an interesting problem to formulate the techniques for passing multiple labels from node to node in a network for detecting communities, label propagation problem has been considered for the research.