Incremental Community Detection in Social Networks Using Label Propagation Method

Mohammad Asadi, Foad Ghaderi
Tarbiat Modares University
Tehran, Iran
{m.asadi, fghaderi}@modares.ac.ir

***Abstract*—The structure of online social networks such as Facebook is continuously changing. Phenomena such as birth, growth, contraction, split, dissolution, and merging with other communities are issues that occur in the communities of online social networks over time. However, characteristics of the consecutive time slots of these networks depend on each other, and independent investigation of each time slot is not efficient for detecting communities in terms of execution time due to the big size of data in each time slot. In order to detect the changes in communities over time, there is a need for algorithms that can detect communities incrementally with proper precision. In this paper, we propose an unsupervised machine learning algorithm for incremental detection of communities using the label propagation method, called Incremental Speaker-Listener Propagation Algorithm (ISLPA). ISLPA can detect both overlapping and non-overlapping communities incrementally after removing or adding a batch of nodes and edges over time. Execution time and modularity comparison on a subset of Facebook dataset confirm that despite the reduced computational costs, the proposed algorithm has promising performance.**

I. Introduction

Nowadays, people have an easy access to the Internet in most areas of the world, and the usage of online social networks has significantly increased among them. Hence, the popularity of online social networks such as Instagram and Facebook has attracted the attention of researchers to investigate the users' behavior in social media.

Understanding the structure of social networks is of great importance when analyzing social networks. More specifically, it is desirable to detect and track the ever-chaining user communities in the network structure. Social network community detection has many applications in marketing, collaborative filtering, customer and society behavior analytics, national security, etc. [1].

In general, social networks can be represented in form of graphs with a set of nodes and edges. In such graphs, the nodes and edges are considered as users and their relationships, respectively [2]. In social networks, community structure can be interpreted as the sets of nodes that are densely connected internally [3].

These communities can be divided into two types of overlapping and non-overlapping structures [4]. In overlapping communities, one node can be a member of multiple communities simultaneously. However, in the non-overlapping communities, all communities must be separate, without even a single common node.

From the other side, online social networks such as Facebook are dynamic and their communities evolve over time by adding or removing users and changing their relationships [5]. Despite the growing number of community detection algorithms, those that have been developed for analyzing static graphs are not efficient for this purpose because they need extended execution time. While consecutive time slots of online social networks are dependent on each other [6], these algorithms consider each time slot independent of the others.

In this paper, we propose an unsupervised machine learning algorithm for incremental detection of both overlapping and non-overlapping communities over time. The algorithm is based on the label propagation method. In section II, we discuss the related work in community detection. In section III, we describe our proposed algorithm for incremental detection of communities called Incremental Speaker-Listener Propagation Algorithm (ISLPA). In section IV, we compare the results of the proposed algorithm with those of SLPA method applied to the Facebook New Orleans networks dataset. The results are reported for both overlapping and non-overlapping communities. Finally, we conclude this paper in section V.

II. Related Work

Early attempts for community detection were focused on non-overlapping communities [4]. However, real-world social network communities always overlap. For example, a person on a social network such as Facebook can belong to two groups of friends and family simultaneously. So, this feature in social networks has attracted the attention of researchers, and they have proposed different methods for detecting overlapping communities in recent years. The algorithms that are introduced for detecting the structure of overlapping communities can be grouped into 5 different main categories [7]: 1) Clique percolation, 2) Line graph and link partitioning, 3) Local expansion and optimization, 4) Fuzzy clustering, and 5) Agent-based and dynamic algorithms:

1) Clique percolation method (CPM) assume that a community includes a set of fully connected subgraphs which share common nodes, and it searches for adjacent cliques in order to detect overlapping communities. CFinder [8] is one of the algorithms implemented based on the Clique percolation

method. The computational complexity of this method in most real applications is polynomial and is not usually efficient for graphs from large social networks. CPMw [9] and SCP [10] are other methods that detect communities based on the Clique percolation. OLCPM [11], which can incrementally detect overlapping communities over time, is an online version of the CPM method. This method presents a considerable enhancement in the execution time compared with other methods that use clique percolation techniques. In summary, this class of methods is similar to pattern matching methods rather than community detectors. Their goal is to find specific and local structures in a network.

2) Also, the idea of partitioning the edges instead of nodes has been studied for community detection. A node is considered as an overlapping node if it is recognized that the connecting edges belong to different communities. Ahn et al. [12] partitioned the edges using hierarchical clustering according to the similarity of the edges. Evans et al. [13] converted the network into a weighted linear graph, whose nodes were equal to the edges of the main graph. Then, non-overlapping community detection algorithms were used. The partition obtained in a linear graph on the nodes makes it possible to obtain a partition on the edges of the main graph. CDAEO method [14] has introduced a post-processing approach which can determine the extent of an overlap. Kim et al. [15] extended the Map equation method for linear graphs. This method encodes the random walk path on a linear graph using the minimum description length.

3) The algorithms that use local expansion and optimization method are based on the expansion of a natural or partial community. Often, these methods have a local benefit function which measures the quality of communities. Based on the expansion of a partial community, Hamann et al. [16] proposed an algorithm that detects overlapping communities locally. This algorithm finds overlapping communities in weighted and unweighted networks.

4) In fuzzy detection algorithms, a membership vector is calculated for each node, which shows the belonging coefficient of the node to each community. To detect communities with fuzzy methods, one should know the number of communities. The algorithm proposed by Nepusz et al. [17] is based on the fuzzy detection method. This algorithm defines the detection of overlapping communities as a nonlinear constrained optimization problem, and simulated annealing (SA) techniques are used to approximate the optimization problem in this algorithm.

5) The label propagation methods are categorized in the agent-based and dynamic algorithms category. In label propagation methods, nodes with the same label are assigned to the same community. It is possible to assign a node to multiple communities and create overlapping communities as these methods allow a node to have more than one label in its memory. SLPA algorithm [18] is a speaker-listener based information propagation process. This algorithm is an extended version of the LPA algorithm [19] for detecting overlapping communities. More precisely, both of the LPA and SLPA algorithms have been developed as a community detector for graphs which their nodes do not have any label at the start of the algorithm, so these algorithms do the community detection and learning process in an unsupervised manner. From the other side, the input graph may contain a small set of labeled nodes and a large number of unlabeled nodes at the start of the algorithm. Hence, the semi-supervised learning process can be used for the community detection of unlabeled nodes [20]. Based on the label propagation techniques, Liu et al. [21] suggested a semi-supervised algorithm in order to find communities in the large networks statically. In this method, labels from the labeled nodes are propagated to their unlabeled neighbors. In summary, this class of methods excels at the running time and the output quality compared with the other explained categories [7].

In addition to the semi-supervised and unsupervised machine learning techniques such as label propagation, genetic and deep learning algorithms are other artificial intelligence methods which have been used to form an agent-based modeling in the community detection problem. CC-GA [22], which detects disjoint communities in static social networks, has been implemented based on the genetic algorithm. This algorithm produces better accuracy compared with other methods that have been developed based on the genetic algorithm. Also, in recent years, deep learning techniques have been used to extract useful patterns from social networks. Yang et al. [23] developed a nonlinear model using deep learning, which can statically detect non-overlapping communities in complex networks. DFuzzy algorithm [24] is a deep learning-based method which can detect overlapped communities. This method statically detects fuzzy clusters with high accuracy in the large complex networks.

In summary, most of the existing community detection methods have been developed for static graphs, and they aren't suitable for the detection of communities in the streaming networks. Also, the best algorithms that can detect communities over dynamic networks have the following limitations. Fanrong et al. [25] proposed a method for the incremental detection of overlapping communities in dynamic networks. This method is based on the link partitioning techniques, and it has been developed by extending the static community detector called DBLINK. Despite its high quality in detecting communities, this method has high computational cost. ILCD is an algorithm based on the local expansion and optimization method [26]. It detects overlapping communities incrementally in dynamic networks, but this algorithm cannot manage the node or edge deletion. LabelRankT [6] is an algorithm that detects non-overlapping communities incrementally in dynamic networks via label propagation techniques. This algorithm doesn't consider the overlap between communities while groups usually have common users in social media. SALPA [27], which can detect overlapping communities in temporal networks, is a method based on the combination of the LPA algorithm and spreading activation process. Although this method enhances the performance of the LPA on dynamic networks, it doesn't consider the direction between the users in social networks. In other words, SALPA only works on undirected networks. These limitations led us to propose a method for incremental detection of communities over time which has been described in the following section.

III. Proposed Algorithm

Since the proposed algorithm is based on the SLPA algorithm, we first introduce SLPA in section III-A. Then, by extending the SLPA in section III-B, we describe our proposed algorithm for the incremental detection of communities in section III-C.

A. SLPA:Speaker-Listener Label Propagation Algorithm

Often, accessing the labeled data is very limited and expensive for analyzing the structure of social media [20]. In practice, all data points are unlabeled in most of the available dataset such as Facebook. Hence, the unsupervised learning is a suitable choice for clustering a group of data points based on their similarity when researchers have a dataset without the ground-truth.

The most important methods for unsupervised clustering can be grouped into 4 categories [28]: 1) Partitioning methods, 2) Hierarchical methods, 3) Graph-based methods, and 4) Density-based approaches. Since social networks are modeled with a graph, social media researchers have used graph-based methods to analyze social networks in recent years. More specifically, label propagation is a graph-based approach for detecting communities. The LPA and SLPA are two efficient community detectors developed based on the label propagation techniques. The position of the SLPA algorithm has been hierarchically shown among machine learning methods in Fig. 1.

More precisely, the SLPA algorithm [18] is a speaker-listener based information propagation process. This algorithm is an extended version of the LPA algorithm which can detect overlapping communities. In general, SLPA is presented for detecting communities in static graphs. In summary, SLPA can be defined in three steps:

1) Initialization;
2) Evolution; and
3) Post-processing.

Given a graph $G = (V, E)$, there is a memory for each node $v_i \in V$ which is empty at first, i.e., $M_i = \{\}$. $N = |V|$ represents the number of nodes and $L = [l_1, l_2, \dots, l_{|v|}]$ considered as the unique label of nodes (the ID of each node).

The first step in SLPA is initialization. In this step, the label (ID) of each node is added to its memory, which means $M_i = \{l_i^1\}$.

In the evolution step, the labels are propagated between the nodes based on the listening and speaking rule. The speaking rule can be a weighted random selection among the existing labels in the speaker's memory, and the listening rule can be a random selection among the most received labels by the listener node. In this step, by defining T as a repeat parameter, the following steps are repeated T times:

1) The nodes of the graph are arranged in a random order and place within a list. According to the order of nodes in the list, steps (2) to (4) are repeated for all of the nodes.
2) One node is selected as a listener.

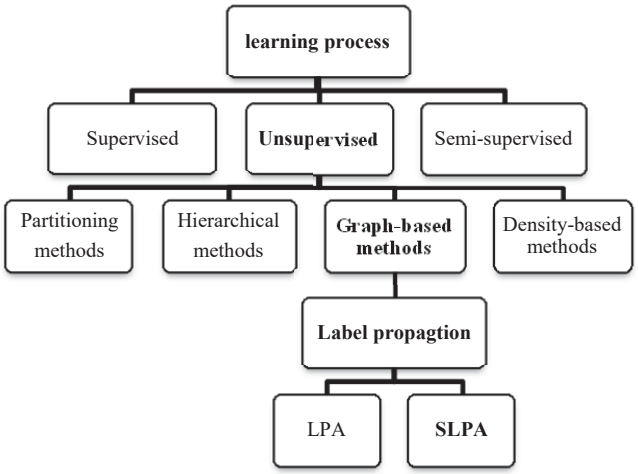

Fig. 1. The position of the SLPA algorithm among machine learning techniques

3) The listener's neighbors are identified. Then, each of these neighbors sends a label to the listener according to the speaking rule.
4) The listener node accepts one of the labels sent by the speakers based on the listening rule and adds this label to its memory.

In this step, T is usually considered to be a number greater than 20 [18].

After label propagation phase, each node $v_i \in V$ is associated with a sequence of labels $M_i = \{l_i^1, l_i^2, \dots, l_i^T\}$. The repetition number of labels in the memory of each node represents the node's belonging coefficient to each community. In the post-processing step, by defining r as a threshold parameter, the labels whose presence probability in the memory is less than this value are deleted from the node's memory. In general, r is a number between 0 and 1. To detect overlapping communities, the value of r is usually set to 0.3, and by choosing a number greater than 0.5 for r, the non-overlapping communities are detected [18]. After post-processing step, if more than one label exists in the node's memory, that node is considered as an overlapping node. The existence of overlapping nodes is caused to form an overlap between communities.

B. ESLPA: Extended Speaker-Listener Label Propagation Algorithm

Performing the label propagation for the users who have not changed their relationships between the two consecutive time slots increases the execution time and memory usage. Increasing in the execution time and memory usage is sensible when the number of these users is high. In this section, we introduce the extended version of the SLPA which is called ESLPA for the following purposes. The main purpose of the ESLPA is to create a function that detects communities only for the users who have changed their relationship compared with the previous time slot. Consequently, due to the high computational costs of labeling nodes, this action can significantly reduce the execution time without the

performance reduction. Another purpose of the ESLPA is to propagate the information from the previous time slot to the current time slot via boundary nodes. Due to the structural dependency between the consecutive time slots, this action can improve the analysis in the current time slot. In summary, we use ESLPA as a community detector of the changed nodes in our proposed method for incremental detection of communities.

The SLPA algorithm consists of three input arguments: 1) $G = (V, E)$ as an input graph, 2) T as a stop criterion and 3) r as a threshold. SLPA is suggested for detecting communities in static graphs [18]. In order to detect communities incrementally over time, SLPA can be extended as follows:

1) We added a parameter called $snapshot$ to SLPA's input arguments. This parameter indicates the time slot's number (we can assign the sequential numbers to consecutive time slots of the graph). Using this parameter, we can access graph information such as detected communities and the content of the node's memory in future time slots.
2) In SLPA, each node can be a listener or a speaker [18]. In the extended version of SLPA, we added a parameter called $SpeakerNodes$ to SLPA's input arguments. This parameter contains a list of the nodes which can be used only as a speaker and cannot be used as a listener. The labels of these nodes are copied from the previous time slot. So, information is propagated from the previous time slot to the current time slot.

Given a graph $G_{snapshot} = (V_{snapshot}, E_{snapshot})$ at time slot $snapshot$, there is a memory for each node $v_{(i,\ snapshot)} \in V_{snapshot}$ which is empty at first, i.e, $M_{(i,\ snapshot)} = \{\}$. $N_{snapshot} = |V_{snapshot}|$ represents the number of nodes in the graph, $Nodes_{snapshot} = \left[l_{(1,\ snapshot)}, l_{(2,\ snapshot)}, \dots, l_{(N,\ snapshot)}\right]$ considered as the unique label of the nodes (the ID of each node), and $SpeakerNodes_{snapshot} = [s_{(1,\ snapshot)}, s_{(2,\ snapshot)}, \dots, s_{(k,\ snapshot)}]$ represents a list of nodes in the graph which can be used only as a speaker and cannot be used as a listener.

The ESLPA consists of the following steps (the pseudo-code of ESLPA is given in Algorithm 1):

1) The first step in ESLPA is initialization. In this step, the label (ID) of each node is added to its memory, which means $M_{(i,\ snapshot)} = \{l^1_{(i,\ snapshot)}\}$.
2) The second step is label propagation. In this step, by defining T as a repeat parameter, the following steps are repeated T times:
   a) The nodes of the graph are arranged in a random order and placed within a list. Steps (b) to (c) are repeated for all of the nodes.
   b) One node is selected according to its order in the created list at step (a).
   c) If the selected node exists in the $SpeakerNodes_{snapshot}$ list, the label of this node at time slot *t* is copied from its label at time slot *t-1*. Then, the label is added to the node's memory. But if the selected node doesn't exist in the $SpeakerNodes_{snapshot}$ list, it is considered as a listener node. Then, the listener's neighbors are identified. Each of these neighbors sends a label to the listener according to the speaking rule. After that, the listener accepts one of the labels sent by the speakers based on the listening rule and adds this label to its memory. In this paper, the speaking rule has been considered as a weighted random selection among the existing labels in the speaker's memory, and the listening rule has been considered as a random selection among the most received labels by the listener node.
3) The last step in ESLPA is post-processing. After label propagation step, each node $v_{(i,\ snapshot)} \in V_{snapshot}$ is associated with a sequence of labels $M_{(i,\ snapshot)} = \{l^1_{(i,\ snapshot)}, l^2_{(i,\ snapshot)}, l^T_{(i,\ snapshot)}\}$. In the post-processing step, by defining r as a threshold parameter, the labels whose presence probability in the memory is less than this value are deleted from the node's memory.

To conclude, the ESLPA community detector is a general function which can do the learning process in both supervised and semi-supervised manner according to the $SpeakerNodes$ parameter. For more explanation, all nodes in the input graph do not have any label at the start of the algorithm if the $SpeakerNodes$ list is empty. In this case, the community detection is done in an unsupervised way such as the SLPA algorithm. But if the $SpeakerNodes$ list is not empty, the nodes in this list have labels and rest of the nodes in the input graph are unlabeled at the start of the algorithm. Consequently, the community detection is done in a semi-supervised manner by propagating labels from the labeled nodes to their unlabeled neighbors.

C. ISLPA: Incemental Speaker-Listener Label Propagation Algorithm

The structure of online social networks such as Facebook is changing over time, but consecutive time slots of these networks are dependent on each other. So, the independent study of each time slot is not effective for detecting communities in terms of the execution time. Indeed, the network's structure at time slot *t* is significantly dependent on the network's structure at time slot *t-1* and does not suddenly undergo major changes [6]. In this section, we propose an unsupervised algorithm based on the label propagation techniques, called Incremental Speaker-Listener Propagation Algorithm (ISLPA). ISLPA algorithm has been developed based on the changes between two consecutive time slots of the graph. The general idea of the proposed method is to use the information obtained from the graph's analysis at time slot *t-1* to improve the analysis at time slot *t*.

In general, ISLPA consists of the following steps (the pseudo-code of the ISLPA is given in Algorithm 2):

1) Firstly, consecutive time slots of the graph are given as an input to the ISLPA algorithm.
2) In this step, the ESLPA algorithm is used to detect communities at the first time slot. Since the graph at the first time slot is considered as a static graph, the

$SpeakerNodes_1$ parameter is calculated *Null* in ESLPA which means each node can be a listener or a speaker at the first time slot.

3) The following steps are repeated upcoming time slots:
   a) By comparing the two consecutive time slots *t* and *t-1*, added or deleted edges are detected at time slot *t*. According to the deleted and added edges, the changed nodes are identified at time slot *t*.
   b) All of the changed nodes are separated from the graph with their neighbors and edges. Separated nodes and edges are placed in a new graph called $Part\_graph_{snapshot}$.

      The nodes in $Part\_graph_{snapshot}$ that have not changed are recognized at time slot *t*. These nodes are placed in a list called $SpeakerNodes_{snapshot}$. This list contains the nodes that can be used only as a speaker and cannot be used as a listener. These nodes are considered as boundary nodes that propagate their labels obtained from time slot *t-1* to their neighbors at time slot *t*. Consequently, information is propagated from the previous time slot to the current time slot.
   c) ESLPA parameters ($Part\_graph_{snapshot}$ as the input graph, $SpeakerNodes_{snapshot}$, T, r, and $snapshot$) are sent to the ESLPA's input arguments. So, ESLPA is used to detect the communities of changed nodes at time slot *t*.
   d) In this step, the unchanged nodes in $G_{snapshot}$ are identified at time slot *t*. The communities of these nodes are copied from time slot *t-1*. After this step, the communities of all nodes have been detected completely at time slot *t*.

To conclude, the social media datasets which are given to the community detector algorithms are usually unlabeled. In other words, there is no node with the specific community at the start of the algorithm and all nodes need to be labeled. Hence, we considered this tip for the implementation of the ISLPA algorithm. As a result, when the ESLPA function is called at the first time slot, it detects communities in an unsupervised way due to the empty $SpeakerNodes_1$ list. So, the ISLPA algorithm is an unsupervised community detector. Also, the labels obtained from each time slot are used to assign the labels to unlabeled nodes at the next time slot.

The communities of social networks experience different kinds of evolution events over time [5]. We can see these events by incremental detection of communities. For example, we ran our proposed algorithm with parameters $T = 100$ and $r = 0.3$ on two consecutive simple graphs, G_1 and G_2 in Figs. 2 and 3, to show the different evolution events in the consecutive time slots. ISLPA has detected three communities in graph G_1 (communities have been separated from each other by the circle in Fig. 2). By deleting node 11 and adding edges $\{1, 7\}$, $\{2, 8\}$ and $\{9, 3\}$ graph G_1 has been updated as graph G_2.

ISLPA has detected two communities in graph G_2, i.e., node 1 has been detected as an overlapping node between the two detected communities in Fig. 3 (if we set $r = 0.5$, node 1 only belongs to the $community_{(2,1)}$, and we have 2 non-overlapped communities). The $community_{(1,1)}$ from graph G_1 has been dissolved in graph G_2, and its members have been merged with the $community_{(1,2)}$, so a new community has been created in graph G_2 called $community_{(2,1)}$. Also, $community_{(1,3)}$ from G_1 has grown by added node 1 to its members in graph G_2.

Algorithm 1 : ESLPA($G_{snapshot}$, $SpeakerNodes_{snapshot}$, T, r, $snapshot$)

$N_{snapshot} = |V_{snapshot}|$
$Nodes_{snapshot} = [l_{(1,\ snapshot)}, l_{(2,\ snapshot)}, \dots, l_{(N,\ snapshot)}]$
$SpeakerNodes_{snapshot} = [s_{(1,\ snapshot)}, s_{(2,\ snapshot)}, \dots, s_{(k,\ snapshot)}]$
for $i = 1 : N_{snapshot}$ **do**
  $M_{(i,\ snapshot)} = \{\}$;
end
Step 1: Initialization:
for $i = 1 : N_{snapshot}$ **do**
  $M_{(i,\ snapshot)}$.add($l^1_{(i,\ snapshot)}$);
end
Step 2 : Label propagation:
for $t = 1 : T + 1$ **do**
  $Nodes_{snapshot}$.RandomOrder( );
  for $i = 1 : N_{snapshot}$ **do**
    if $Nodes_{(i,\ snapshot)}$ exists in $SpeakerNodes_{snapshot}$ list **do**
      $l^t_{(i,\ snapshot)} = l^t_{(i,\ snapshot-1)}$;
      $M_{(i,\ snapshot)}$. add($l^t_{(i,\ snapshot)}$);
    end
    else
      $Listener = Nodes_{(i,\ snapshot)}$;
      $Speakers = Listener$.NeihgborsIdentifier( );
      for $j = 1 : len(speakers)$ **do**
        $send_j = Speakers_j$.WeightedRandomizedSelection( ) ;
      end
      $listen = listener$. RandomizedMostReceivedLable( );
      $M_{listener}.add(listen)$;
    end
  end
end
Step 3: post-processing
for $i = 1 : N_{snapshot}$ **do**
  delete labels with *probability* $< r$ from $Nodes_{(i,\ snapshot)}$ memory;
end
// **Output**
returns the communities of $G_{snapshot}$;

Algorithm 2 : ISLPA(G_1, G_2, …, G_N)

$T = 100$
$r = 0.3$ for overlapping community detection OR $r = 0.5$ for non-overlapping community detection.
ESLPA(G_1, *Null* , T, r, 1);
for $snapshot = 2 : N$ **do**
  DetectInsertEdges($G_{snapshot}$);
  DetectDeleteEdges($G_{snapshot}$);
  DetectChangedNodes($G_{snapshot}$);
  $Part\_graph_{snapshot}$ = Create_Part_graph($G_{snapshot}$);
  $SpeakerNodes_{snapshot}$= Detect_SpeakerNodes($Part\_graph_{snapshot}$)
  ESLPA($Part\_graph_{snapshot}$, $SpeakerNodes_{snapshot}$, T, r, $snapshot$)
  $UnChangedNodes[G_{snapshot}]$ = DetectUnChangedNodes($G_{snapshot}$);
    for j in $UnChangedNodes[G_{snapshot}]$ **do**
      $Community_{(i,\ snapshot)} = Community_{(i,\ snapshot-1)}$;
    end
end
// **output**
returns the communities of $G_{snapshot}$;

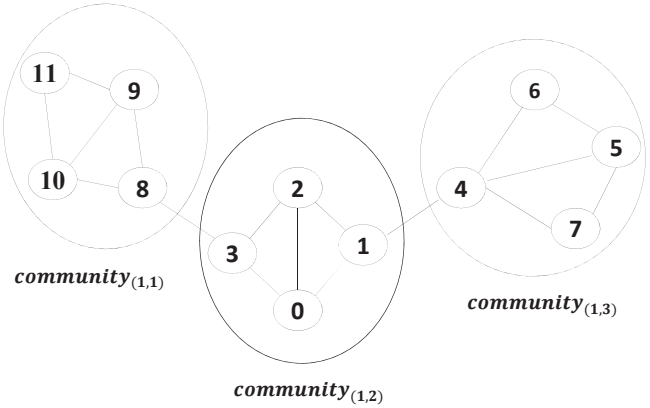

Fig. 2. G_1 with 11 nodes. Circles represent communities detected by ISLPA with parameters $T = 100$ and $r = 0.3$

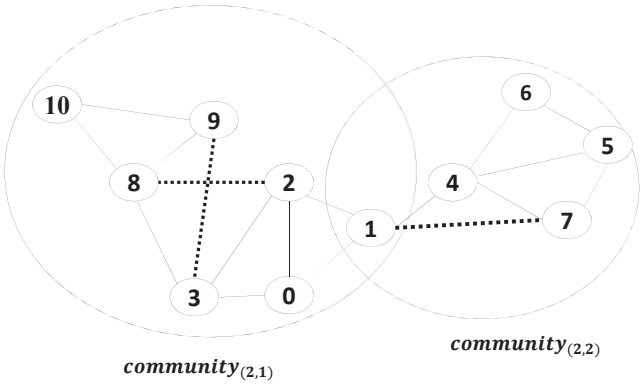

Fig. 3. $\boldsymbol{G_2}$ with 10 nodes that changed from $\boldsymbol{G_1}$ by dissolving, merging and growing. Node **1** has been detected as an overlapping node between the two communities by ISLPA with parameters $\boldsymbol{T = 100}$ and $\boldsymbol{r = 0.3}$

IV. Exprimental Results

In this section, we compare the performance of ISLPA with that of SLPA in terms of the execution time and the quality. This comparison is done for a subset of the Facebook dataset. The dataset contains a list of all of the user-to-user links from the Facebook New Orleans networks. The activity of more than 60,000 users has been collected in the Facebook New Orleans networks from Sep 5, 2006, to Jan 21, 2009, ordered temporally (Data is available at http://socialnetworks.mpi-sws.org/data-wosn2009.html). Since this dataset is available in the static form, we separated the users' weekly activities in order to generate streaming data for the purpose of incremental detection of communities. To this end, we used the first-week information as a first time slot and updated information weekly. Eventually, we created 122-time slots of Facebook New Orleans networks for our test. The number of nodes in the first time slot is 1471 with 863 edges while the number of nodes in the last time slot is 61096 with 327458 edges.

Since the SLPA algorithm detects communities statically and does not use the information of the past time slots, all nodes must participate in the label propagation phase to be identified their communities in each time slot. But, the ISLPA algorithm in the label propagation phase only reinitializes the label distribution of the nodes which have been changed compared with the previous time slot. The communities of the unchanged nodes are gotten from the previous time slot. Reducing the number of nodes and edges in the label propagation phase can save time and memory. Fig. 4 and 5 show the number of nodes and edges from the Facebook dataset which participate in the label propagation phase for the static and incremental detection of communities.

Similar to SLPA, the ISLPA method generates stable outputs for different runs when *T* is greater than 20 [18]. For effective detection of overlapping communities, the value of *T* is set to 100 and the value of *r* is set to 0.3. By choosing a number greater than 0.5 for *r*, ISLPA can also detect non-overlapping communities.

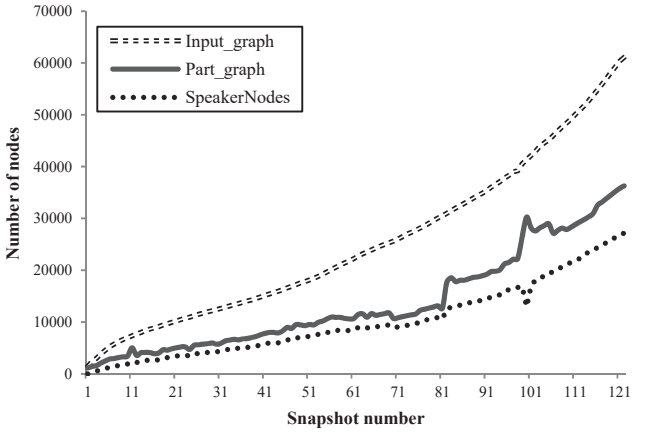

Fig. 4. The number of nodes from the 122-time slots of the Facebook New Orleans networks which are used in the label propagation phase for static (*Input_graph*) and incremental (*Part_graph*) detection of communities. Also, *SpeakerNodes* represents the number of nodes which are used only as a speaker and cannot be used as a listener in the incremental detection of communities.

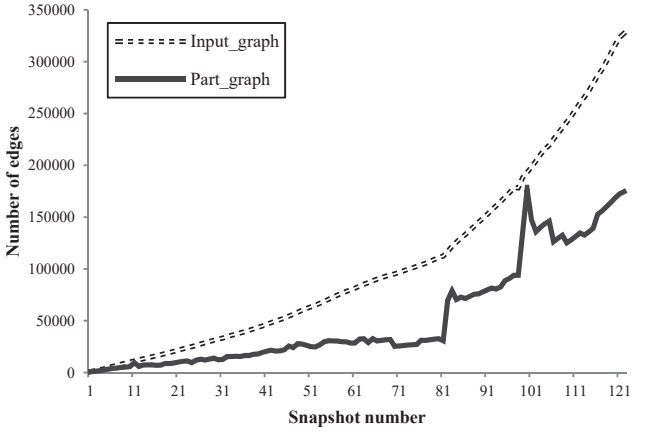

Fig. 5. The number of edges from the 122-time slots of the Facebook New Orleans networks which are used in the label propagation phase for static (***Input_graph***) and incremental (***Part_graph***) detection of communities.

In section IV-A, we compare the execution time of ISLPA with that of SLPA for 122-time slots of Facebook New Orleans networks. Also, modularity comparison is reported between ISLPA and SLPA for our experimental dataset in section IV-B. Moreover, execution time and modularity results are reported for both overlapping and non-overlapping detection of communities.

A. Execution time Comparison

The time complexity between two consecutive time slots in ISLPA can be calculated as follows. Each of lines (5) to (9) in Algorithm 2, can be executed in $O(1)$. The worst case of time complexity for the line (10) is when all of the nodes are changed in the input graph. In this case, the time complexity is $O(Te)$, which T is a small constant (In this paper, we set T to 100), and e is the total number of edges. Also, line (11) can be executed in $O(1)$. Finally, $O(n)$ is a time complexity of the lines (12) to (14), which n is the number of the unchanged nodes at time slot t. Thus, the overall time complexity of ISLPA between two consecutive time slots is $O(Te)$ in the worst case, implying $O(e)$ in general.

Also, the time complexity of SLPA is $O(Tm)$ [18], which T is a Stop Criterion, and m is the total number of edges.

We implemented both SLPA and ISLPA algorithms in Node.js platform. We used Windows 7 with 16 GB RAM and Intel i5-3210M (2.5 GHz) CPU for this implementation.

By setting parameter values as T=100 and r=0.3, the execution time results have been compared between two SLPA and ISLPA algorithms for the detection of overlapping communities in each time slot. Fig. 6 shows this comparison for 122-time slots of the Facebook New Orleans networks. SLPA detects communities statically in each time slot while ISLPA detects communities incrementally over time.

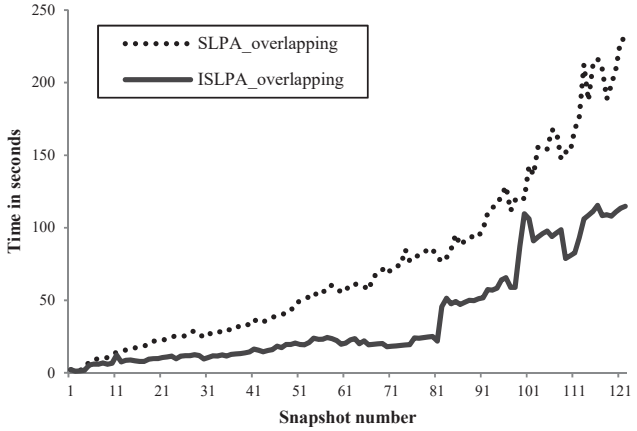

Fig. 6. Execution time comparison between SLPA and ISLPA for 122-time slots of the Facebook New Orleans networks. The parameter values are set to $T = 100$ and $r = 0.3$, the execution time results have been reported for the detection of overlapping communities.

The execution time results have been compared for the detection of non-overlapping communities for 122-time slots of the Facebook New Orleans networks by setting parameters $T = 100$ and $r = 0.5$. Fig. 7 shows the execution time comparison between SLPA and ISLPA for our experimental dataset.

Finally, the average execution time of both SLPA and ISLPA algorithms for each time slot has been reported in Fig. 8. This report shows the results of both overlapping and non-overlapping community detection cases. The results show that for static detection of communities by SLPA algorithm, about 75 seconds is averagely consumed for each time slot while for incremental detection of communities by ISLPA algorithm, about 37 seconds is consumed for each time slot averagely. Thus, ISLPA has detected both overlapping and non-overlapping communities about 2 times faster than SLPA.

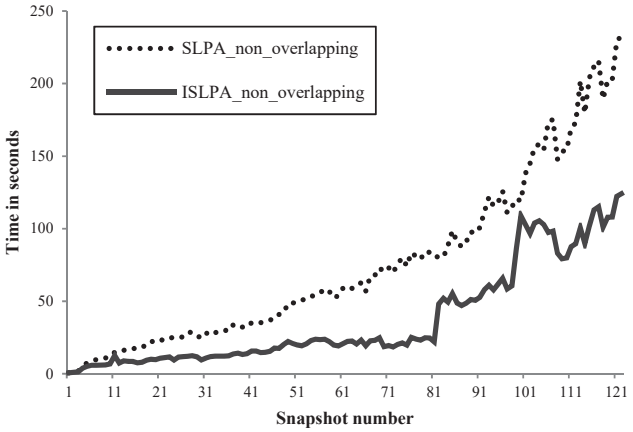

Fig. 7. Execution time comparison between SLPA and ISLPA for 122-time slots of the Facebook New Orleans networks. The parameter values are set to $T = 100$ and $r = 0.5$, the execution time results have been reported for the detection of non-overlapping communities.

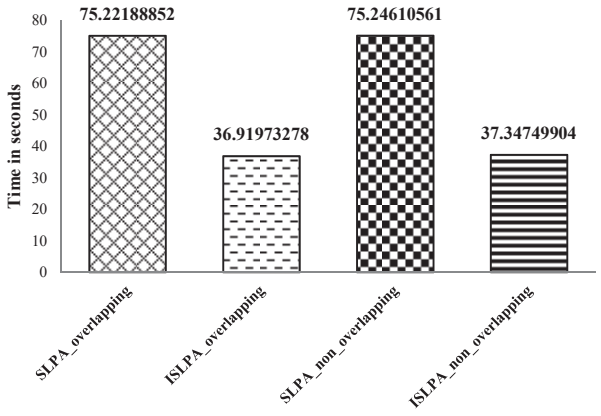

Fig. 8. The average execution time for each time slot of the Facebook New Orleans networks. Results have been reported for both overlapping and non-overlapping community detection by SLPA and ISLPA algorithms.

B. Modularity Comparison

Modularity is a metric which is used to measure the quality of the detected communities in graphs. The modularity value lies in the range [-1, 1]. More precisely, 1 means the highest algorithm productivity and -1 the lowest. The function introduced by Newman [29] is used to measure the quality of the detected non-overlapping communities. This function is defined as follows:

$$Q = \frac{1}{2|E|}\sum_{ij}[A_{ij} - \frac{K_i\ K_j}{2|E|}]\ \delta(c_i, c_j) \qquad (1)$$

where E represents the number of edges in the graph, A is the adjacency matrix, c represents the node's community, and K is the node's degree. If node i and node j place in the same community $\delta(c_i, c_j)$ is equal to 1 and otherwise, it is 0.

Newman's modularity is used to measure the quality of the detected non-overlapping communities. However, in social networks nodes can belong to more than one community. Nicossia et al. [30] has extended Newman's modularity based on the edges which can measure the quality of the detected overlapping communities. This extension is defined as follows:

$$Q_{ov} = \frac{1}{2|E|}\sum_{c\in C}\sum_{i,j\in c}[r_{ijc}\,A_{ij} - s_{ijc}\frac{K_i\ K_j}{2|E|}] \qquad (2)$$

where each community c is a member of C, which is the set of all the communities. Also, the belonging coefficient of edge $e = (i,j)$ to community c is calculated as follows:

$$r_{ijc} = \mathrm{F}(a_{i,c}, a_{j,c}) = \frac{1}{(1+e^{-f(a_{i,c})})(1+e^{-f(a_{j,c})})} \qquad (3)$$

where $a_{i,c}$ and $a_{j,c}$ show the belonging coefficient of nodes i and j to community c. Also, $f(a_{i,c})$ is a simple linear function which is computed as follows:

$$f(x) = 2px - p\ ,p \in \mathbb{R} \qquad (4)$$

In papers that detect overlapping communities [31], [32], p has considered to be 30. Also, s_{ijc} in (2) is defined as follows:

$$s_{ijc} = \frac{\sum_{k\in V}\mathrm{F}(a_{i,c}, a_{k,c})\ \sum_{k\in V}\mathrm{F}(a_{k,c}, a_{j,c})}{|V|^2} \qquad (5)$$

where V represents the number of nodes in the graph.

In order to compare, first we set parameters $T = 100$ and $r = 0.5$. We used (1) to measure the modularity of the detected non-overlapping communities in our experimental dataset. Next, by setting $T = 100$ and $r = 0.3$, we used (2) to calculate the modularity of the detected overlapping communities over 122-time slots of the Facebook New Orleans networks. Modularity comparison between two SLPA and ISLPA algorithms have been reported in Fig. 9. Test results show that ISLPA can detect both overlapping and non-overlapping communities with the accuracy close to the SLPA's accuracy over 122-time slots. As a result, despite the reduced computational costs, the ISLPA has promising performance.

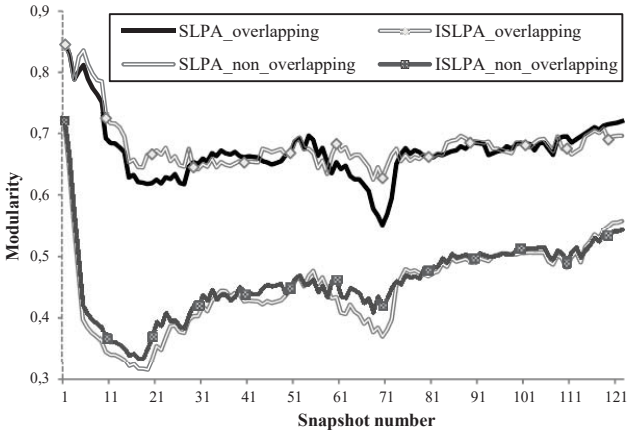

Fig. 9. Modularity comparison between SLAP and ISLPA for 122-time slots of the Facebook New Orleans networks. By setting $T = 100$ and $r = 0.5$, modularity has been reported for the detection of non-overlapping communities. Also, by setting $T = 100$ and $r = 0.3$, modularity has been reported for the detection of overlapping communities.

V. Conclusion

Community detection is one of the important issues in social network analysis. The consecutive time slots of social networks such as Facebook are highly correlated with each other in terms of the structure. Thus, the independent study of each time slot is not efficient for detecting communities in terms of the execution time and memory usage. In this paper, we introduced an unsupervised machine learning algorithm for incremental detection of communities using the label propagation method, called ISLPA. ISLPA can detect both overlapping and non-overlapping communities on directed and undirected networks over time.

Test results such as execution time and modularity on the 122-time slots of the Facebook New Orleans networks showed that ISLPA can detect both overlapping and non-overlapping communities about 2 times faster than SLPA with the modularity accuracy close to the SLPA's modularity accuracy.

References

[1] H. Chen, R. H. L. Chiang, and V. C. Storey, "Business intelligence and analytics: from big data to big impact," Management Information Systems Quarterly (MISQ), vol. 36, No. 4, pp. 1165-1188, Dec. 2012.

[2] M. Khatoon and W. A. Banu, "A Survey on Community Detection Methods in Social Networks," International Journal of Education and Management Engineering (IJEME), Vol. 5, No. 6, 2015.

[3] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," Physical Review E (PRE), vol. 69, p. 026113, Feb. 2004.

[4] S. R. Chintalapudi and M. H. M. K. Prasad, "A survey on community detection algorithms in large scale real world networks ," Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on. 2015: IEEE.

[5] G. Palla, A. L. Barabasi, and T. Vicsek, "Quantifying social group evolution," Nature 446, No. 7136, pp. 664-667, 2007.

[6] J. Xie, M. Chen, and B. K. Szymanski, "LabelRankT: incremental community detection in dynamic networks via label propagation," DyNetMM '13 Proceedings of the Workshop on Dynamic Networks Management and Mining, pp. 25-32, 2013.

[7] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: the state of the art and comparative study," ACM Computing Surveys (CSUR), vol. 45, No. 43, 2013.

[8] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, “Uncovering the overlapping community structure of complex networks in nature and society,” Nature, vol. 435, No. 7043, pp. 814-818, 2005.

[9] I. Farkas, D. 'Abel, G. Palla, and T. Vicsek, “Weighted network modules,” New Journal of Physics, vol. 9, No. 6, p. 180, 2007.

[10] J. M. Kumpula, M. Kivelä, K. Kaski, and J. Saramäki, “Sequential algorithm for fast clique percolation,” Physical Review E (PRE), vol. 78, No. 2, p. 026109, 2008.

[11] S. Boudebza, R. Cazabet, F. Azouaou, and O. Nouali, “OLCPM: An Online Framework for Detecting Overlapping Communities in Dynamic Social Networks,” Journal of Computer Communications, In press, 2018.

[12] Y. Y. Ahn, J. P. Bagrow, and S. L Eehmann, “Link communities reveal multiscale complexity in networks,” Nature, vol. 466, No. 7307, pp. 761-764, 2010.

[13] T. S. Evans and R. Lambiotte, “Line graphs of weighted networks for overlapping communities,” The European Physical Journal B-Condensed Matter and Complex Systems, vol. 77, No. 2, pp. 265-272, 2010.

[14] Z. Wu, Y. Lin, H. Wan, and Sh. Tian, “A fast and reasonable method for community detection with adjustable extent of overlapping,” in Intelligent Systems and Knowledge Engineering (ISKE), 2010 International Conference on. 2010: IEEE.

[15] Y. Kim and H. Jeong, “Map equation for link communities,” Physical Review E (PRE), vol. 84, No. 2, p. 026110, 2011.

[16] M. Hamann, E. Röhrs, and D. Wagner, “Local Community Detection Based on Small Cliques,” *Algorithms,* vol. 10, No. 3, p. 90, 2017.

[17] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó “Fuzzy communities and the concept of bridgeness in complex networks,” Physical Review E (PRE), vol. 77, No. 1, p. 016107, 2008.

[18] J. Xie, B.K. Szymanski, and X. Liu, “SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process,” Data Mining Workshops (ICDMW), 2011 11th International Conference on. 2011: IEEE.

[19] U. N. Raghavan, R. Albert, and S. Kumara, “Near linear time algorithm to detect community structures in large-scale networks," Physical Review E (PRE), vol. 76, p. 036106, 2007.

[20] X. Zhu and Z. Ghahramani, “Learning from labeled and unlabeled data with label propagation,” Carnegie Mellon University, Technical Report CMU CALD 02 107, 2002.

[21] D. Liu, H. Y. Bai, H. J. Li, and W. J. Wang, “Semi-supervised community detection using label propagation,” International Journal of Modern Physics B, vol. 28, No. 29, 2014.

[22] A. Said, R. A. Abbasi, O. Maqbool, A. Daud, and N. R. Aljohani, “CC-GA: A clustering coefficient based genetic algorithm for detecting communities in social networks,” Applied Soft Computing (ASC), vol. 63, pp. 59–70, Feb. 2018.

[23] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, “Modularity based community detection with deep learning,” IJCAI’16 Proceedings of the twenty-fifth international joint conference on artificial intelligence, pp. 2252–2258, 2016.

[24] V. Bhatia, and R. Rani, “Dfuzzy: a deep learning-based fuzzy clustering model for large graphs,” Knowledge and Information Systems (KAIS), doi:10.1007/s10115-018-1156-3, 2018.

[25] F. Meng, F. Zhang, M. Zhu, Y. Xing, and J. Shi, “Incremental Density-Based Link Clustering Algorithm for Community Detection in Dynamic Networks,” Mathematical Problems in Engineering, vol. 2016, Article ID 1873504, 2016.

[26] R. Cazabet, F. Amblard, and C. Hanachi, “Detection of overlapping communities in dynamical social networks,” in Social Computing (SocialCom), 2010 International Conference on. 2010: IEEE.

[27] M. Sattari and K. Zamanifar, “A spreading activation-based label propagation algorithm for overlapping community detection in dynamic social networks,” Data & Knowledge Engineering (DKE), vol. 113, pp. 155–170, Jan. 2018.

[28] D. Xu and Y. Tian, “A Comprehensive Survey of Clustering Algorithms,” Annals of Data Science (AODS), vol. 2, No. 2, pp. 165–193, 2015.

[29] M. E. J. Newman, “Modularity and community structure in networks,” Proceedings of the National Academy of Sciences, vol. 103, no. 23, pp. 8577–8582, 2006.

[30] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri, “Extending the definition of modularity to directed graphs with overlapping communities,” Journal of Statistical Mechanics: Theory and Experiment, vol. 2009, no. 03, p. P03024, 2009.

[31] J. Xie, S. Kelley, and B. K. Szymanski, “Overlapping community detection in networks: The state-of-the-art and comparative study,” ACM Computing Surveys (CSUR), vol. 45, no. 4, pp. 43:1–43:35, Aug. 2013.

[32] S. Gregory, “Finding overlapping communities in networks by label propagation, ” New Journal of Physics, vol. 12, no. 10, p. 103018, 2010.