Far East Journal of Electronics and Communications

http://www.pphmj.com
http://dx.doi.org/10.17654/EC018010001
Volume 18, Number 1, 2018, Pages 1-27 **ISSN: 0973-7006**

A REVIEW ON OVERLAPPING AND NON-OVERLAPPING COMMUNITY DETECTION ALGORITHMS FOR SOCIAL NETWORK ANALYTICS

Edi Surya Negara and Ria Andryani

Faculty of Computer Science
Universitas Bina Darma
Jl. A. Yani. No. 3 Plaju
Palembang, Indonesia
e-mail: e.s.negara@binadarma.ac.id
ria.andryani@binadarma.ac.id

Abstract

Community detection is a common problem that exists in the data graph analytics as the social networks analytics. In the context of social networks, community detection is aimed at to find a group or community that has connectedness between individuals (nodes) with high-intensity interaction. In general, types of group on the social networking can be divided into the overlapping and non-overlapping. We provide an overview of some of the algorithms of overlapping and non-overlapping community detections available today to perform an analysis or a breakdown of the data of social networking. The algorithms for overlapping community detection are: (1) local seed selection algorithm; (2) seed set expansion algorithm; (3) speaker listener label propagation algorithm (SPLA) and the algorithms for detection of non-overlapping community are: (4) multithreaded

Received: May 6, 2017; Revised: July 7, 2017; Accepted: July 18, 2017
Keywords and phrases: community detection, data graph analytic, data mining, graph clustering, social network analytics.

community detection algorithm (MCD); (5) core groups graph cluster algorithm (CGGC); (6) complex network cluster detection (CONCLUDE); (7) dense subgraph extraction (DSE).

1. Introduction

Big data is known as a part of business intelligent which works to produce digital contents in the Internet, especially in social media. Phenomena of big data give opportunity to create an availability of data to be functioned in producing information based on social network, exactly by having social networking analysis. Interactions of social media users with the digital contents are used to create a model for structures of social media networking in which it can be found from the thousands until million nodes of the graphs. The relation of interactions between nodes and graphs at social network can be found into interconnection graphs [15]. Social interaction graph is defined as the connection of social networking with node only, but, in other concept, the social interaction graph is known as the relation based on the interaction of one node with another node in the networking. Face of the interaction at social interaction graphs shows a circumstance of information or moving of data from the sources to the destination, such as the interaction of email, namely: sender with receiver or the interaction of telephone communication, etc.

In the reality, the interaction or relation of social media users are known as the representation of the social network. The social network is defined as a network which shows the interaction between one individual with the others or one group with another one. In this case, the context is named by node based on social interaction, such as: friendship, followers and it is called as *edge* too. The existence of wide social network with thousands until million nodes is used to be a part of the network. It is used to know the structures and characteristics of the social network. Social network analysis is a method to analyze the structures of the individuality, group, or the network as whole in the social network. Moreover, the analysis is functioned to understand the network based on complete network or to analyze a node (ego network).

Social network analytics can be done by using data sources of social media, such as twitter, Facebook, and the others, such as the use of data extraction twitter with geospatial data to determine the location of the source of the emergence of public issues on social media [19]. Data of friendship and the followers from the existing network, such as social media or network have been functioned into many studies, namely: terrorism [6, 30], criminal [28], business and economy [25, 14], healthiness and epidemiology [4, 10, 8, 22], social development [31], urban development [9, 17], [36].

Regarding the previous usage, analysis of social network can be focused at the communities in the network too, exactly by using the community detection. In the networking, the community detection is looked as the instrument to understand structures and characteristics of the complex networks and analyze the beneficial of information too.

Based on the terrorism study, community detection is used to detect structures of networking from the terrorist. By using the existing data, community detection functions to detect individuals (nodes) which master in the community through centrality analytics [3]. Moreover, the law enforcer uses the community detection to reduce systematical crime like agent of narcotics. In this case, the community detection is functioned to find the pattern and structures of communication of the agents in their community. In the business and economy, the community detection is used to offer marketing. By having the existing networking data, it can be known which of the users are consumptively toward the products, even the data can be used to look the location of the users. Thus, the analysis is the key to determine a map of target promotion and target selling of the products [29]. In the healthiness study, the graph analytics is used to determine pattern of virus development and the community detection works to find the communities which are victims of the virus infection [10]. In fact, there are many functions of the graph analytics and community detection in other studies.

Informally, communities are defined as the group with high interconnection and interaction in a network. Identification of the community in the social networking is to be complex problem because there are many

different concepts of the community and the complexity of different dynamic networks. Moreover, the difference in characteristics of the communities in the real network, like non-overlapping and overlapping is another problem in detecting communities by using the existing algorithms. Non-overlapping network is familiar with nodes of community in the network in which they do not have relation with another network. It is named as disjoint, while the overlapping network is a condition of nodes in the network to be a part of another community and it is called as a *node* with many communities.

Based on the interactions that occur between the nodes in the graph, the community can also be defined as a subset of nodes in a graph that has a direction, where edges in this subset show the direction of interactions that occur within the graph, also called the *community* based on directed graphs. Malliaros and Vazirgiannis classify several approaches used to graph clustering and community detection on the directed network: (i) naive graph transformation approaches, (ii) transformations maintaining directionality, (iii) approaches that extend objective functions and methodologies to directed networks, and (iv) alternative approaches [18]. Some algorithms used to detect communities on directed graphs are the propagation label algorithm and the infomap algorithm [1]. Tests performed by [1] artificial and real graph data, propagation label algorithms have good performance in scalability, while the infomap algorithm has the best performance for accuracy and computational performance [1].

Looking at the previous explanation, the community detection has been used into many studies belonging to the problem of social network analytics or graph data analytics research. For this article, we take it to review the algorithm of community detection which focused to algorithms used in detecting communities based on non-overlapping and overlapping.

2. Mathematical Background and Complexity in Social Network

On social networks, each individual is denoted as a single point or node with the symbol V, while the relationship between nodes is called the *edges* with the symbol E. In this section, there is a discussion of the mathematical

background and complexity of community detection in social networks.

A. Graph theory

Graph theory is the basic theory that is used to define social network notation. Graph $G = (V; E)$ is a set of nodes and edges that are connected to each other. A graph is sometimes called an *undirected graph*. For an edge, $e = (u, v) \in E$, the nodes *u* and *v* are called the *ends* of *e*. For each edge of a graph, it can be said that the edge connects the nodes or the nodes are adjacent to each other. An edge is a loop if it connects a node to itself, $e = (v, v)$. An edge that has the same ends as another edge is called a *multiple edge*. A graph with no multiple edges or loops is called *simple*. An undirected graph can be represented as a point for each node of a graph and as a line between points for each edge, not necessarily straight.

A directed graph $G = (V; E)$ is a graph, where the edges *E* are ordered pairs of nodes. An edge $e = (u; v) \in E$ indicates an edge from *u* to *v*, where *u* is the tail of the edge and *v* is the head of the edge.

A path in a graph (or directed graph) is an alternating sequence of nodes and edges, $v_0, e_1, v_1, e_2, v_2, ..., e_p, v_p$, where v_i are distinct nodes and e_i indicates that v_{i-1} is adjacent to v_i. A directed path is a path, where the edge e_i, indicates an edge from v_{i-1} to v_i. A graph is connected if there exists a path between every two nodes. A graph that is not connected is disconnected.

The complement of a graph *G* is a graph *H*, where *H* has the same node set as *G* and the nodes of *H* are only adjacent if they were not adjacent in *G*.

A subgraph, $H = (V', E')$ of a graph, $G = (V, E)$, is a graph with nodes that are subset of $G, V' \subseteq V$ and edges that are a subset of the edges of $G, E' \subseteq E$. An induced or full subgraph, $H = (V', H')$ of graph $G = (V, E)$ has a node set that is strictly a subset of the node set of $G, V' \subset V$, and every edge of E' meeting with the node set V' in *H*. A connected component of a graph is subgraph that is connected.

The order of a graph is the number of nodes, $|V|$, within a graph, often known as *n*. The size of a graph is the number edges, $|E|$, within the graph, often known as *m*. The degree of node, *v*, is the number of edges connecting to, or adjacent to *v* noted as $\deg(v)$ or $d(v)$. Similarly for directed graphs, we say a node has an outgoing degree, k_i^{out}, and incoming degree, k_i^{in}, indicating the number of edges coming from a node and coming into node, respectively. A weighted graph has a weighting function $w(ij)$ on the edges $w(ij)$, where $w(ij) \in R$. In a weighted graph, the strength of a node, s_i, is defined to be the sum of weights of adjacent edges. The size of a weighted graph is the sum of weights of all edges *W*.

A cut edge of a graph is an edge of a graph that when removed causes the graph to become disconnected. A set of edges that when removed causes the graph to become disconnected is known as cut set [32].

B. Complexity

The growth of social networks dynamic is directly proportional to the level of complexity of the network. Complexity is one factor that determines the effectiveness of the algorithm used to analyze social networks. *Big O notation* is a mathematical notation that is used mainly in the field of computer science. This notation is used to express the effectiveness of an algorithm. This notation works by taking into account the input provided by the user. *Big O notation* is the comparison of the growth of an algorithm, $f(n)$, to another function $g(n)$ as a way of determining whether functions are equivalent. We say $f(n) = O(g(n))$ if for some n_0 and all $n \geq n_0$, then there exists $C \in R$ such that $|f(n)| \leq C|g(n)|$.

3. Community Detection Algorithms

Along with the growth of social networking, group dynamics, as well as differences in the complexity of the group on a network into its own problems are to be solved in community detection. Currently, there are many

algorithms that have evolved to perform graph clustering and community detection, both on the community overlapping or non-overlapping. In this section, we present the algorithms for some non-overlapping which have been developed and used to perform clustering graph or community detection in social networks.

Table 1. Community detection algorithm evaluated

No	No Algorithms	Overlapping	Non-overlapping	Directed	Undirected
1	Local seed selection algorithm	√			√
2	Seed set expansion algorithm	√			√
3	Speaker listener label propagation algorithm (SPLA)	√		√	
4	Multithreaded community detection algorithm (MCD)		√		√
5	Core groups graph cluster algorithm (CGGC)		√		√
6	Complex network cluster detection (CONCLUDE)		√	√	
7	Dense subgraph extraction (DSE)		√	√	

In the network theory or graph theory, overlapping community detection is one node having multiple community memberships in the networks. Non-overlapping community detection is a type of detection which assumes the network that can be partitioned into dense regions where nodes have more connections amongst each other than those with the rest of networks.

In this section, we describe seven states of the art algorithms for overlapping and non-overlapping community detections. Three algorithms (local seed selection algorithm, seed set expansion algorithm, speaker listener label propagation algorithm (SPLA)) can be used to detect overlapping community. Four algorithms (multithreaded community detection algorithm (MCD), core groups graph cluster algorithm (CGGC), complex network cluster detection (CONCLUDE) and dense subgraph extraction (DSE)) can be used to detect non-overlapping community. Three algorithms (speaker listener label propagation algorithm (SPLA), complex network cluster detection (CONCLUDE) and dense subgraph extraction (DSE)) can be used to detect directed graph (directed community), and four algorithms (local seed selection algorithm, seed set expansion algorithm, multithreaded community detection algorithm (MCD) and core groups graph cluster

algorithm (CGGC)) can be used to detect undirected graph (undirected community). For an overview of the algorithms, see Table 1.

A. Local seed selection algorithm

Local seed selection algorithm is a community detection algorithm that works by detecting the most potential local seed. This algorithm is the elaboration of a definition in the local community. Local community is defined as the detection of connectivity between nodes in the subnetwork that is focused only on ego network. Problems of local community structure were first proposed by Clauset on [7].

Clauset defined the local modularity R in order to solve the local community detection problem [7]. This metric mainly focuses on the connectivity of vertices in S:

$$R = \frac{\sum_{ij} S_{ij}\delta_{(i,\,j)}}{\sum_{ij} S_{ij}}, \tag{1}$$

where $\delta_{(i,\,j)}$ is 1 when either $v_i \in s$ and $v_j \in C$ or v_i in C and $v_j \in S$, and is 0 otherwise; S_{ij} is 1 when nodes i and j are connected and is 0 otherwise. Clauset's algorithm always chooses the node which results in the largest ΔR as the one that is inserted into C. However, this approach asks for predefined parameter that is used to determine the size of C. Meanwhile, its result is influenced by the source node. Figure 1 shows the local seed selection algorithm or Clauset algorithm.

Algorithm. The general algorithm for the greedy maximization of local modularity, as given in the text.

add v_0 to $\mathcal{C}$

add all neighbors of v_0 to $\mathcal{U}$

set $\mathcal{B} = v_0$

while $|\mathcal{C}| < k$ **do**

  for each $v_j \in \mathcal{U}$ **do**

    compute ΔR_j

  end for

  find v_j such that ΔR_j is maximum

  add that v_j to $\mathcal{C}$

  add all new neighbors of that v_j to $\mathcal{U}$

  update R and $\mathcal{B}$

end while

Figure 1. Local seed selection algorithm (source: [7]).

In the working of the algorithm local seed selection algorithm, initially we place the source node in the community, $v_0 = C$, and place its neighbors in U. At each step, the algorithm adds to C (and to B, if necessary) the neighboring node that results in the largest increase (or smallest decrease) in R, breaking ties randomly. Finally, we add to U newly discovered nodes, and update our estimate of R. This process continues until it has agglomerated either to a given number of nodes k, or it has discovered the entire enclosing component, whichever happens first.

The computation of the ΔR_j associated with each $v \in U$ can be done quickly. Using an expression derived from equation (1):

$$\Delta R_j = \frac{x - R_y - z(1 - R)}{T - z + y}, \tag{2}$$

where x is the number of edges in T that terminated a v_j, y is the number of edges that will be added to T by the agglomeration of v_j (i.e., the degree of v_j is $x + y$) and z is the number of edges that will be removed from T by the agglomeration. Because ΔR_j depends on the current value of R, and on the y and z that correspond to v_j, each step of the algorithm takes time proportional to the number of nodes in U. This is roughly kd, where d is the mean degree of the graph. We note that this will be a significant overestimate for graphs with non-trivial clustering coefficients, or significant community structure, when C is a large portion of the graph. Thus, in general, the running time for the algorithm is $O(k^2 d)$ or simply $O(k^2)$ for a sparse graph, i.e., when $m \sim n$. As it agglomerates nodes, the algorithm outputs a function $R(t)$, the local modularity of the community centered on v_0 after t steps, and a list of vertices paired with the time of their agglomeration.

B. Seed set expansion algorithm

Seed set expansion algorithm is one of the most elegant method of overlapping community detection algorithms [34], but the algorithm is computationally expensive to perform detection in the network community analysis using maximal cliques as seeds [27]. This algorithm works by local expansion and optimization based on growth natural community [16]. Selection of seed kernel is based on the distance as the *k*-means and spectral clustering. A function of distance on the choice of seed can determine the good seed on a node in the network. In particular, this algorithm proposes a strategy involving many computing clusters using a weighted kernel *k*-means algorithm on the graph (algorithms Graclus) [12].

To measure the cluster quality in seed set expansion algorithm using cut, conductance and normalized cut, represent a graph with n nodes as an $n \times n$ adjacency matrix A such that $A_{ij} = e_{ij}$ is the weight of an edge between nodes i and j, or $A_{ij} = 0$ if there is no edge. Assume that all the graphs are undirected graphs, A is symmetric. Let us define links (C_p, C_q) to be the sum of edge weights between node sets C_p and C_q.

Cut. The cuts of cluster C_i is defined as the sum of edge weights between C_i and its complement, $V \backslash C_i$. That is,

$$cut(C_i) = Links(C_i, V \backslash C_i). \quad (3)$$

Conductance. The conductance of a cluster is defined to be the cut divided by the least number of edges incident on either set C_i or $V \backslash C_i$:

$$cond(C_i) = \frac{links(C_i, V \backslash C_i)}{\min(links(C_i, V), links(V \backslash C_i, V))}. \quad (4)$$

By definition, $condC_i = cond(V \backslash C_i)$. The conductance of a cluster is the probability of leaving that cluster by a one-hop walk starting from the smaller set between C_i and $V \backslash C_i$.

Normalized cut. The normalized cut of a cluster is defined by the cut with volume normalization as follows:

$$ncut(C_i) = \frac{links(C_i, V \backslash C_i)}{links(C_i, V)}. \quad (5)$$

Notice that $ncut(C_i)$ is always lesser than or equal to $cond(C_i)$.

Seed set expansion algorithm consists of four phases as shown in Figure 2. These are: filtering, seeding, seed set expansion, and propagation [33].

In the *filtering phase*, this phase removes regions of the graph that are trivially separable from the rest of the graph, so will not participate in overlapping clustering. In the seeding phase, find seeds in the filtered graph, and in seed set expansion phase expand the seed sets using a personalized PageRank clustering scheme. Finally, in the propagation phase, further expand the communities to the regions that were removed in the filtering phase.

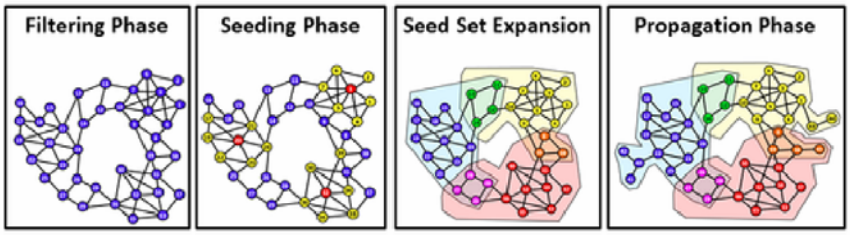

Figure 2. Seed set expansion algorithm phase (source: http://hindawi.com).

The goal of an effective seeding strategy is to identify a diversity of vertices that lie within a cluster of good conductance. For seeding phase, there are two methods used to determine the best seeding, namely: (1) Graclus centers and (2) spread hubs [33].

Graclus centers. One way to achieve these goals is to first apply a high quality and efficient graph partitioning scheme in order to compute a collection of sets with fairly small conductance. For each set (cluster), we find the most central node according to the kernel that corresponds to the normalized cut measure. The idea here is roughly that we want something that is close to the partitioning which ought to be good but that uses overlap to produce better boundaries between the partitions. See Figure 3 for complete the procedure of the algorithm.

Algorithm. Seeding by Graclus centers

Input: Graph G, the number of seeds k.

Output: the seed set $\mathcal{S}$.

1: Compute exhaustive and non-overlapping clusters $\mathcal{C}_i$ $(i = 1, ..., k)$ on G.
2: Initialize $\mathcal{S} = \varnothing$.
3: **for** each cluster $\mathcal{C}_i$ do
4: **for** each vertex $v \in \mathcal{C}_i$ do
5: Compute $\texttt{dist}(v, \mathcal{C}_i)$ using (4).
6: **end for**
7: $\mathcal{S} = \{\arg\min_v \texttt{dist}(v, \mathcal{C}_i)\} \bigcup \mathcal{S}$.
8: **end for**

Figure 3. Algorithm seeding by Graclus centers (source: [33]).

Spread hubs. From another viewpoint, the goal is to select a set of well distributed seeds in the graph, such that they will have high coverage after we expand the sets. This algorithm greedily chooses an independent set of k points in the graph by looking at nodes in order of decreasing degree. Full procedure of this algorithm is shown in Figure 4.

Algorithm. Seeding by spread hubs

Input: Graph $G = (\mathcal{V}, \mathcal{E})$, the number of seeds k.

Output: the seed set $\mathcal{S}$.

1: Initialize $\mathcal{S} = \varnothing$.
2: All vertices in $\mathcal{V}$ are unmarked.
3: **while** $|\mathcal{S}| < k$ do
4: Let $\mathcal{T}$ be the set of unmarked vertices with max degree.
5: **for each** $t \in \mathcal{T}$ do
6: **if** t is unmarked then
7: $\mathcal{S} = \{t\} \bigcup \mathcal{S}$.
8: Mark t and its neighbors
9: **end if**
10: **end for**
11: **end while**

Figure 4. Algorithm seeding by spread hubs (source: [33]).

Local optimal egonets. This strategy was presented in [13]. Let $ego(s)$ denote the egonet of vertex s which is defined to be the union of s and its neighbors. The reference [13] takes an egonet whose conductance is smaller than the conductance of any of its neighbors' egonets, that is, they select a seed s such that for all v adjacent to s,

$$cond(ego(s)) \leq cond(ego(v)). \tag{6}$$

Random seeds. Given the number of seeds k, randomly take k seeds in the graph. Andersen and Lang gave some theoretical justification for why this method should be competitive [2].

In the *seed set expansion phase* is performed the expansion of the entire network to expand the cluster surrounding the seeds. The most effective technique to do this expansion is a personalized PageRank vector [21]. Figure 5 shows procedures and algorithms in expanding to find a low conductance set near a seed.

Algorithm. Find a low conductance set near a seed

Input: Graph G, seed set $\mathcal{S}$, PageRank link-following probability parameter $0 < \alpha < 1$, accuracy $\varepsilon > 0$

Output: Low conductance set $\mathcal{C}$.

1: Initialize x_v, $r_v = 0$ for $v \in \mathcal{V}$, set $r_v = 1/|\mathcal{S}|$ for all $v \in \mathcal{S}$

2: **while** Any $r_v > \deg(v)\varepsilon$, set v to this vertex **do**

3: Update $x_v = x_v + (1 - \alpha)r_v$.

4: For each $(v, u) \in E$,

update $r_u = r_u + \alpha r_v/(2\deg(u))$

5: Update $r_v = \alpha r_v/2$

6: **end while**

7: Sort vertices by decreasing $x_v/\deg(v)$

8: For each prefix set of vertices in the sorted list, compute the conductance of that set and set $\mathcal{C}$ to be the set that achieves minimum.

Figure 5. Algorithm find a low conductance set near a seed (source: [33]).

In the *propagation phase*, for each detached whisker connected via a bridge, we add that piece to all of the clusters that utilize the other node in the bridge. This procedure is described in Figure 6:

Algorithm. Propagation module

Input: Graph $G = (\mathcal{V}, \mathcal{E})$, biconnected core $G_C = (\mathcal{V}_C, \mathcal{E}_C)$, communities of $G_C : \mathcal{C}_i \ (i = 1, ..., k) \in \mathcal{C}$.

Output: communities of *G*.

1: **for** each $\mathcal{C}_i \in \mathcal{C}$ **do**
2: Detect bridges $\mathcal{E}_{B_i}$ attached to $\mathcal{C}_i$.
3: **for** each $b_j \in \mathcal{E}_{B_i}$ do
4: Detect the whisker $w_j = (\mathcal{V}_j, \mathcal{E}_j)$ which is attached to b_j.
5: $\mathcal{C}_i = \mathcal{C}_i \cup \mathcal{V}_j$.
6: **end for**
7: **end for**

Figure 6. Propagation module (source: [33]).

C. Speaker-listener label propagation algorithm (SLPA)

SLPA. The speaker-listener label propagation algorithm is a community detection algorithm that is classified as a methodological principle of community detection algorithms or graph clustering using principle 'model-based'. The algorithm works by considering the dynamic processes that occur in the network to detect or find a community using a statistical model to produce parts of the network into a community. SLPA is an extension of label propagation algorithm (LPA) [23]. SLPA can work with both types of networks that identify non-overlapping and overlapping by spreading a label that represents community memberships between nodes in the graph [35].

This algorithm works by initiation of each node in the network with a unique label. Then the algorithm will assign one node to act as an information provider and other nodes will act as a listener or consumer information. At SLPA, each node has a memory and takes into account the label to make fast decisions. This allows the SLPA to avoid producing a number of communities of small size compared to other algorithms. Figure 7 shows that the SLPA algorithm works through three stages.

Algorithm. *SLPA*(T, r)

T: the user defined maximum iteration

r: post-processing threshold

(1) First, the memory of each node is initialized with a unique label.

(2) Then, the following steps are repeated until the maximum iteration *T* is reached:

a. One node is selected as a listener.

b. Each neighbor of the selected node randomly selects a label with probability proportional to the occurrence frequency of this label in its memory and sends the selected label to the listener.

c. The listener adds the most popular label received to its memory.

(3) Finally, the post-processing based on the labels in the memories and the threshold *r* is applied to output the communities.

Figure 7. SLPA algorithm (source: [35]).

Post-processing and community detection. In SLPA, the detection of communities is performed when the stored information is post-processed. Given the memory of a node, SLPA converts it into a probability distribution of labels. Since labels represent community id's, this distribution naturally denotes the *strength* of association to communities to which the node belongs. To produce crisp communities in which the membership of a node to a given community is binary, i.e., either a node is in a community or not, a simple thresholding procedure is performed. If the probability of seeing a particular label during the whole process is less than a given threshold $r \in [0, 0.5]$, then this label is deleted. After thresholding, connected nodes having a particular label are grouped together and form a community. If a node contains multiple labels, then it belongs to more than one community and is called an *overlapping node*. A smaller value of *r* produces a larger number of communities. However, the effective range is typically narrow in practice. When $r \geq 0.5$, SLPA outputs disjoint communities [35].

Complexity. The initialization of labels requires $O(n)$, where *n* is the total number of nodes. The outer loop is controlled by the user defined

maximum iteration *T*, which is a small constant which in our experiments was set to 100. The inner loop is controlled by *n*. Each operation of the inner loop executes one speaking rule and one listening rule. The speaking rule requires exactly $O(1)$ operation. The listening rule takes $O(\bar{k})S$ on average, where $\bar{k}$ is the average node degree. In the post-processing, the thresholding operation requires $O(Tn)$ operations since each node has a memory of size *T*. In summary, the time complexity of the entire algorithm is $O(Tn\bar{k})$ or $O(Tm)$, linear with the total number of edges *m*. The execution times for synthetic networks were averaged for each *k* over networks with different structures, i.e., different degree and community size distributions. The results shown in Figure 8 confirm the linear scaling of the execution times [35].

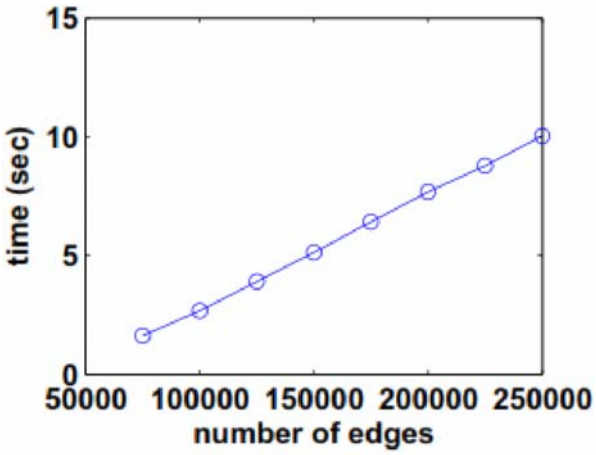

Figure 8. The execution times of SLPA in synthetic networks with $n = 5000$ and average degree $\bar{k}$ varying from 10 to 80 (source: [35]).

D. Multithreaded community detection (MCD)

The *multithreaded community detection* (*MCD*) works by creating separate partitions to graph the data entered. Grouping starts by making each node a community apart putting them together into a group (*community*) taking into account the negative conductance modularities until it reaches the maximum value [24].

To do clustering algorithm, maintain the graph group (*community*), where each group (*community*) is represented by a single *node*. In clustering graph community each iteration of the algorithm calculates the change to metric optimization after the merger of the two groups (*community*). This algorithm maintains a community graph where every node represents a community, edges connect communities when they are neighbors in the input graph, and weights count the number of input graph edges either collapsed into a single community graph edge or contained within a community graph node. This algorithm currently does not require counting of the vertices in each community. Such an extension is straightforward.

From a high level, this algorithm repeats the following steps until reaching some termination criteria [24]:

(1) associate a score with each edge in the community graph, exiting if no edge has a positive score,

(2) greedily compute a weighted maximal matching using those scores,

(3) contract matched communities into a new community graph.

E. Core groups graph cluster (CGGC)

Core groups graph cluster (CGGC) algorithm is an ensemble learning method for grouping data graph. The method combines several methods of different clustering to decide the final partition of *node* within a group (community) [20].

The working of the algorithm is a clustering algorithm to run as much as *k* times to generate *k* a different partition of nodes. A graph is constructed by using a partition of *P* as nodes and final grouping step performed on smaller charts to find the final partition. Improvements to these methods were made by *k* partitioning iteratively until early achieved the best partition.

CGGC algorithm uses the agreements of several clusterings with fair quality to decide whether a pair of vertices should belong to the same cluster. The groups of vertices which are assigned to the same cluster in every clustering (i.e., the maximal overlaps of the clusterings) are denoted as core

groups. To abstract from any specific quality measure, this algorithm uses the term good partition for a partition that has a good quality according to an arbitrary quality measure. The CGGC scheme consists of the following steps [20]:

(1) Create a set *S* of *k* good partitions of *G* with base algorithm A_{intial}.

(2) Identify the partition $\hat{P}$ of the maximal overlaps in *S*.

(3) Create a graph $\hat{G}$ induced by the partition $\hat{P}$.

(4) Use base algorithm A_{final} to search for a good partition of $\hat{G}$.

(5) Project partition of $\hat{G}$ back to *G*.

F. Complex network cluster detection (CONCLUDE)

The complex network cluster detection (CONCLUDE) aims to combine the accuracy of global method with the efficiency of local methods. This is done by incorporating knowledge of the topology of the entire graph to a heuristic algorithm to *community detection* [11].

This works in two stages: In the first stage, it uses an information propagation model, based on random and non-backtracking walks of finite length, to compute the importance of each edge in keeping the network connected (called *edge centrality*). Then edge centrality is used to map network vertices onto points of a Euclidean space and to compute distances between all pairs of connected vertices. In the second stage, CONCLUDE uses the distances computed in the first stage to partition the network into clusters.

The working of the algorithm is the first κ path edge centrality of *edge* of the graph are computed using ERW-K path algorithms. κ path edge centrality is a measure how importance of an *edge* in the graph. Thus κ path edge centrality is used to calculate the distance between all the *nodes* connected in the graph. Recently, the distance is calculated by measuring the weight of the resulting edge of the partition by using Louvain method. The

pseudo-code describing the various steps performed by CONCLUDE is reported in Figure 9.

Algorithm. CONCLUDE ($G = \langle V, E \rangle$: a graph, κ: an integer, ρ: an integer)

1: $\vec{\omega} \leftarrow$ ERW-Kpath(G, κ, ρ)

2: $\Delta \leftarrow$ Compute-Pairwise-Distances(G, $\vec{\omega}$)

3: $C \leftarrow$ Louvain-Method(G, Δ)

Figure 9. CONCLUDE algorithm (source: [11]).

CONCLUDE algorithm finds communities adopting the paradigm of the network modularity maximization and exploits an approximate technique inspired by the Louvain method (LM).

Louvain method (LM) is a computationally effective algorithm. Thus it is well-suited for partitioning large networks. The input of Louvain method (LM) is a weighted network $G = \langle V, E, W \rangle$ being W the weights associated with each edge. Louvain method (LM) operates as follows:

$$\Delta Q = \frac{\Sigma_c + k_i^c}{2m} - \left(\frac{\Sigma_c + k_i}{2m} \right)^2 - \left[\frac{\Sigma_c}{2m} - \left(\frac{\Sigma_c}{2m} \right)^2 - \left(\frac{k_i}{2m} \right) \right]. \qquad (7)$$

The advantage of this approach with respect to the original LM is twofold: first, obtain the splitting of clusters connected by edges with low distance, which is a global feature, maximizing the network modularity, while LM only relies on local information (i.e., vertex neighborhood); second, LM strategy is able to produce an edge weighting, while the original LM and most of current clustering algorithms cannot infer edge weights in case of unweighted networks [11].

G. Dense subgraph extraction (DSE)

Dense subgraph extraction (DSE) algorithm is one of the methods that can be used to detect community inspired from blocking matrix. Matrix blocking is the process of reordering the rows and columns so that a critical mass of no element is along or near the diagonal [5]. The blocking algorithm

presented in [26] groups similar columns (and rows) of a matrix according to a cosine similarity measure.

Dense subgraph extraction (DSE) algorithm considers the following three types of graphs, with an appropriate definition of density for each one [5].

(1) Undirected graphs. Undirected graphs are the most common models of networks, where the directions of the connections are unimportant, or can be safely ignored. A natural definition of the graph density is

$$d_G = \frac{|E|}{|V|(|V|-1)/2}. \tag{8}$$

Note that $d_G \in [0, 1]$ and a subgraph has the density one if and only if it is a clique. DSE algorithm for the case of undirected graph is shown in Figure 10.

Algorithm. Finding dense subgraphs of a sparse undirected graph

Input: Sparse undirected graph *G*, density threshold $d_{\min}$.

1: Compute the matrix *M* as defined in (4).
2: Sort the largest *t* nonzero entries of *M* in ascending order, where $t = nz(A)$. Denote the sorted array by *C*.
3: Construct the hierarchy *T* according to the sorted vertex pairs designated by *C*.
4: COUNT-VERTICES-AND-EDGEST(T, G)
5: Compute *r.density* for all nodes *r* of *T* according to (1).
6: EXTRACT-SUBGRAPHS$(T.root)$
7: **function** COUNT-VERTICES-AND-EDGES (T, G)
8: Initialize $r.num\_edg \leftarrow 0$ for all nodes *r* of *T*.
9: Construct the LCA data structure for *T*.
10: **for all** edge $\{i, j\}$ of *G* do
11: Find the lowest common ancestor *r* of *i* and *j*.
12: $r.num\_edge \leftarrow r.num\_edge + 1$
13: end for

```
14:   COUNT-VERTICES-AND-EDGES-WRAP-UP(T.root)
15: end function
16: function COUNT-VERTICES-AND-EDGES-WRAP-UP(r)
17:     if r.left ≠ nil and r.right ≠ nil then
18:         COUNT-VERTICES-AND-EDGES-WRAP-UP(r.left)
19:         COUNT-VERTICES-AND-EDGES-WRAP-UP (r.right)
20:     end if
21:     if r.left ≠ nil and r.right ≠ nil then
22:         r.num _ vertex ←
             r.left.num _ vertex + r.right.num _ vertex
23:         r.num _ edge ←
             r.left.num _ edge + r.right.num _ edge + r.num _ edge
24:     else
25:         r.num _ vertex ← 1
26:     end if
27: end function
28: function EXTRACT-SUBGRAPH(r)
29:     if r.density > d_min then
30:         Output the leaves of the subtree rooted at r.
31:     else if r.left ≠ nil and r.right ≠ nil then
32:         EXTRACT-SUBGRAPHS(r.left)
33:         EXTRACT-SUBGRAPHS (r.right)
34:     end if
35: end function
```

Figure 10. Finding dense subgraph of a spare undirected graph (source: [5]).

(2) Directed graphs. The density of a directed graph is

$$d_G = \frac{|E|}{|V|(|V|-1)}, \tag{9}$$

since the maximum number of possible directed edges cannot exceed $|E||V|(|V|-1)$. In other words, the density of a directed graph also lies in the range from 0 to 1.

(3) Bipartite graphs. A bipartite graph is an undirected graph whose node set V can be partitioned into disjoint subsets V_1 and V_2, such that every edge connects a vertex from V_1 and one from 2. DSE considers the following alternative definition for the density of a bipartite graph:

$$d_G = \frac{|E|}{|V_1||V_2|}. \tag{10}$$

According to this definition, $d_G \in [0, 1]$ and a subgraph has the density one if and only if it is a biclique. DSE algorithm for the case of bipartite graph is shown in Figure 11.

Algorithm. Finding dense subgraphs of a sparse bipartite graph

---

Input: Sparse bipartite graph G, density threshold $d_{\min}$.

1: [Densification:] Modify the adjacency matrix A of G into $\hat{A}$ as defined in (6).

2: Compute the matrix $\hat{M}$ as defined in (7).

3: Sort the largest t nonzero entries of $\hat{M}$ in ascending order, where $t = nz(\hat{A})$. Denote the sorted array by C.

4: Construct the hierarchy T according to the sorted vertex pairs designated by C.

5: COUNT-VERTICES-AND-EDGES(T, G). [Instead of counting the number of vertices $r.num\_vertex$ for each subgraph, count the number of vertices that belong to each partite set for each subgraph, in a similar way.]

6: Compute $r.density$ for all nodes r of T according to (3).

7: EXTRACT-SUBGRAPHS$(T.root)$

Figure 11. Finding dense subgraphs of sparse bipartite graph (source: [5]).

4. Conclusion

We have presented some algorithms can be used in detecting community on social networks. Local Seed Selection Algorithm and Seed Set Expansion Algorithm detects community by detecting the seeds over a local network and is able to detect networks or sub-networks that are overlapping, while Speaker Listener Label Propagation Algorithm (SPLA) using the global method of detection and is capable of detecting networks or sub-networks that are overlapping. Multithreaded Community Detection Algorithm (MCD) and Core Groups Graph Cluster Algorithm (CGGC) working on the network which is non-overlapping with the approach of weighted graphs while the Complex Network Cluster Detection (CONCLUDE) and Dense subgraph extraction (DSE) also works to detect the network that is non-overlapping with the approach of directed graphs.

Acknowledgement

The authors thank the anonymous referees for their valuable suggestions which led to the improvement of the manuscript.

References

[1] S. Agreste, P. De Meo, G. Fiumara, G. Piccione, S. Piccolo, D. Rosaci, G. M. Sarne and A. Vasilakos, An empirical comparison of algorithms to find communities in directed graphs and their application in web data analytics, IEEE Transactions on Big Data 3 (2017), 289-306.

[2] R. Andersen and K. J. Lang, Communities from seed sets, Proceedings of the 15th International Conference on World Wide Web, ACM, 2006, pp. 223-232.

[3] A. Berinji, Detecting key players in terrorist networks, Ph. D. Dissertation, Uppsala University, 2011.

[4] H. A. Carneiro and E. Mylonakis, Google trends: a web-based tool for real-time surveillance of disease outbreaks, Clinical Infectious Disease 49(10) (2009), 1557-1564.

[5] J. Chen and Y. Saad, Dense subgraph extraction with application to community detection, IEEE Transactions on Knowledge and Data Engineering 24(7) (2012), 1216-1230.

[6] P. Choudhary and U. Singh, A survey on social network analysis for counter-terrorism, International Journal of Computer Applications 112(9) (2015), 24-29.

[7] A. Clauset, Finding local community structure in networks, Phys. Rev. E 72(2) (2005), 026132.

[8] C. Corley, D. Cook, A. Mikler and K. Singh, Using web and social media for influenza surveillance, Advances in Computational Biology, Ser. Advances in Experimental Medicine and Biology, H. Arabnia, ed., Springer, New York, Vol. 680, 2010, pp. 559-564.

[9] J. Cranshaw, R. Schwartz, J. Hong and N. Sadeh, The livehoods project: utilizing social media to understand the dynamics of a city, Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media (ICWSM) 2012, Association for the Advancement of Artificial Intelligence, Association for the Advancement of Artificial Intelligence, 2012.

[10] A. Culotta, Towards detecting influenza epidemics by analyzing twitter messages, Proceedings of the First Workshop on Social Media Analytics (SOMA'10), Association for Computing Machinery Association for Computing Machinery, 2010, pp. 115-122.

[11] P. De Meo, E. Ferrara, G. Fiumara and A. Provetti, Mixing local and global information for community detection in large networks, J. Comput. System Sci. 80(1) (2014), 72-87.

[12] I. S. Dhillon, Y. Guan and B. Kulis, Weighted graph cuts without eigenvectors a multilevel approach, IEEE Transactions on Pattern Analysis and Machine Intelligence 29(11) (2007), 1-14.

[13] D. F. Gleich and C. Seshadhri, Vertex neighborhoods, low conductance cuts, and good seeds for local community methods, Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2012, pp. 597-605.

[14] C. Holsapple, S. Hsiao and R. Pakath, Business social media analytics: definition, benefits, and challenges, Proceedings of the 20th Americas Conference on Information Systems (AMCIS2014), Association for Information Systems, Association for Information Systems, 2014.

[15] M. Huber, M. Mulazzani, M. Leithner, S. Schrittwieser, G. Wondracek and E. Weippl, Social snapshots: digital forensics for online social networks, Proceedings of the 27th Annual Computer Security Applications Conference, ACM, 2011, pp. 113-122.

[16] A. Lancichinetti and S. Fortunato, Community detection algorithms: a comparative analysis, Phys. Rev. E 80(5) (2009), 056117.

[17] X. Long, L. Jin and J. Joshi, Exploring trajectory-driven local geographic topics in foursquare, Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp'12), Association for Computing Machinery, Pittsburgh, United States: Association for Computing Machinery, 2012, pp. 927-934.

[18] F. D. Malliaros and M. Vazirgiannis, Clustering and community detection in directed networks: a survey, Phys. Rep. 533(4) (2013), 95-142.

[19] E. S. Negara, R. Andryani and P. H. Saksono, Twitter data analytics: geospatial data extraction and analysis, J. INKOM 10(1) (2016), 27-36.

[20] M. Ovelgönne and A. Geyer-Schulz, An ensemble learning strategy for graph clustering, Graph Partitioning and Graph Clustering 588 (2012), 187.

[21] J.-Y. Pan, H.-J. Yang, C. Faloutsos and P. Duygulu, Automatic multimedia cross-modal correlation discovery, Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2004, pp. 653-658.

[22] M. Paul and M. Dredze, You are what you tweet: analyzing twitter for public health, Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM) 2011, Association for the Advancement of Artificial Intelligence, Association for the Advancement of Artificial Intelligence, 2011, pp. 265-272.

[23] U. N. Raghavan, R. Albert and S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76(3) (2007), 036106.

[24] J. Riedy, D. A. Bader and H. Meyerhenke, Scalable multi-threaded community detection in social networks, Parallel and Distributed Processing Symposium Workshops and Ph. D. Forum (IPDPSW), 2012 IEEE 26th International, IEEE, 2012, pp. 1619-1628.

[25] M. Rosemann, M. Eggert, M. Voigt and D. Beverungen, Leveraging social network data for analytical crm strategies - the introduction of social bi, Proceedings of the 20th European Conference on Information Systems, Barcelona, Spain, 2012, p. 95.

[26] Y. Saad, Finding exact and approximate block structures for ILU preconditioning, SIAM J. Sci. Comput. 24(4) (2003), 1107-1123.

[27] H. Shen, X. Cheng, K. Cai and M.-B. Hu, Detect overlapping and hierarchical community structure in networks, Phys. A 388(8) (2009), 1706-1712.

[28] M. K. Sparrow, The application of network analysis to criminal intelligence: an assessment of the prospects, Social Networks 13(3) (1991), 251-274.

[29] K. Thiel, T. Kötter, M. Berthold, R. Silipo and P. Winters, Creating Usable Customer Intelligence from Social Media Data: Network Analytics Meets Text Mining, KNIME, 2012.

[30] B. Thuraisingham, Data mining for counter-terrorism, Data Mining: Next Generation Challenges and Future Directions, 2004, pp. 157-183.

[31] UN Global Pulse, Mining Indonesian tweets to understand food price crises, UN Global Pulse, Methods Paper, 2014.

[32] D. B. West, Introduction to Graph Theory, Prentice Hall Upper Saddle River, Vol. 2, 2001.

[33] J. J. Whang, D. F. Gleich and I. S. Dhillon, Overlapping community detection using seed set expansion, Proceedings of the 22nd ACM International Conference on Conference on Information and Knowledge Management ACM, 2013, pp. 2099-2108.

[34] J. Xie, S. Kelley and B. K. Szymanski, Overlapping community detection in networks: the state-of-the-art and comparative study, ACM Computing Surveys 45(4) (2013), 35.

[35] J. Xie and B. K. Szymanski, Towards linear time overlapping community detection in social networks, Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2012, pp. 25-36.

[36] E. S. Negara, D. Kerami, I. M. Wiryana and T. B. M. Kusuma, Researchgate analysis to measure the strength of Indonesian research, Far East Journal of Electronics and Communications 17(5) (2017), 1177-1183.