# Community Detection with Similarity Transition

Jian-Wei Lee, Hung-Wen Peng, and Shie-Jue Lee
Department of Electrical Engineering
National Sun Yat-Sen University
Kaohsiung 80424, Taiwan
Email: {cwlee, wen}@water.ee.nsysu.edu.tw, leesj@mail.ee.nsysu.edu.tw

***Abstract*—Analysis of social networks, in particular discovering communities within networks, has been a focus of recent work and has a variety of applications in many fields. Usually a network is converted to a graph, from which a set of dense subgraphs are identified and regarded as distinct communities. This paper presents a method for identifying a set of dense subgraphs in a given graph. We exploit the idea of matrix column or row similarities, in order to find the vertices of concern in the graph. We apply punishment to the relation matrix and use it to complement the adjacency matrix several times according to the diameter of the graph. Furthermore, not every participating node in the network is required to belong to a certain community. The effectiveness of the proposed method is demonstrated in several artificial and real-life examples.**

*Keywords:* Dense subgraph, social network, community.

## I. Introduction

Researchers are interested in the study of networks describing the topologies of a wide variety of systems, such as the world wide web, social and communication networks, biochemical networks, etc. Identifying the community structure is critical to realizing the the structural and functional properties of the networks [5]. The issue of community detection in social media has received an enormous amount of attention in recent years. Many systems of scientific interest can be represented as graphs, with nodes or vertices linked together in couples by lines or edges. Dense subgraphs in a graph are often interpreted as communities in a corresponding network [1]–[4], since a complex network is usually consists of distinct communities within which the connections between the communities are much fewer than those inside the same community.

Many methods have been proposed to explore the community structure of complex networks [6]. There are some characteristics in real networks like overlapping, directed, and weighted communities. A node may belong to more than one community in the case of overlapping communities [7]. For instance, students in a college may select the same course. The algorithm EAGLE (agglomerativE hierarchicAl clusterinG based on maximaL cliquE) is proposed to reveal both the overlapping and hierarchical community structures in a network. Chen *et al.* [8] proposed a method for detecting dense subgraphs from an undirected and un-weighted graph. The method simply uses the relation matrix of cosine and deletes the weight edges in ascending order. The deletion is performed recursively until dense subgraphs are obtained. In directed graphs, the directions of the edges aiming to vertices definitely can allow us to learn the relations among the vertices. A graph is a weighted graph if a number or weight is assigned to the edge. The weighted edge represents the importance between the vertices [9]. The Louvain method, proposed by Blondel *et al.* [10], is one of the most popular algorithms in the field of weighted community detection.

In this paper, we propose a method based on the one proposed in [8]. Our goal is to identify dense subgraphs from an undirected and un-weighted graph. Firstly, we use the cosine to denote the similarity of any two nodes. A larger cosine shows that the two nodes belong to the same community with a higher probability. In other words, the two nodes have a higher chance to share more neighbors. Secondly, we do the punishment to the matrix $M$ that stores the cosines of the adjacency matrix $A$. That is, we multiply a ratio to the matrix $M$. The punished matrix is called $M_p$. The next step is replace the adjacency matrix $A$ by $M_p$. The result is called $A'$. We do these steps repeatedly according to the diameter of the graph. Thus, we construct a weighted graph $G'(V, E')$ based on the final $M_p$. Finally, a top-down hierarchical clustering is performed by deleting edges $e' \in E'$ one at a time. When $G'$ turns into disconnected first, $V$ is separated in two subsets. The process is terminated when the density of the obtained subsets exceeds a certain density threshold $d_{min}$.

The remaining of this article is organized as follows. In Section II, we describe the idea behind our approach to finding dense subgraphs in a graph. In Section III, we show the algorithm and give an example for illustration. In Section IV, we present some experiments on artificial and real-life data sets. Finally, we give a conclusion in Section V.

## II. Proposed Method

Given a graph $G(V, E)$ which consists of the vertex set $V$ and the edge set $E$, we are concerned about finding dense subgraphs of $G$. In other words, we'd like to extract from $G$ the subgraphs whose density is greater than a specified threshold. Here, we only focus on undirected graphs. An undirected graph is one in which edges have no orientation. They are the most common graphs in mathematics and computer science. For an undirected graph $G(V, E)$, its density $d_G$ is defined as

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

where $|E|$ and $|V|$ denote the size of $V$ and $E$, respectively. Note that $d_G \in$ [0,1].

For a graph $G$, let $A$ be the adjacency matrix of $G$. The entries in $A$ are either 0 or 1. Self-loops in $G$ are not allowed. Consequently, $A$ is symmetric and its diagonal entries are zero. Let matrix $M$ store the cosines between any two columns/rows of the adjacency matrix $A$:

$$M = \frac{\langle A(:,i), A(:,j)\rangle}{\|A(:,i)\| \cdot \|A(:,j)\|} \tag{2}$$

where $A(:,i)$ and $A(:,j)$ are the $i^{th}$ column and the $j^{th}$ column, respectively, $A$. The value of cosine signifies as a probability that the two nodes may belong to a community. The Chen's method uses $M$ for detecting subgraphs. Instead, we do the following:

1) Set $C = 0$. Let the adjacency matrix be $A$.
2) Step 1. We compute the matrix $M$ by Eq.(2). Then we compute another matrix $K$ by

$$\begin{cases} k_{ij} = m_{ij}, & \text{if S} = 1 \\ k_{ij} = m_{ij} \times (1 - \frac{S}{L}), & \text{if S} > 1 \end{cases} \tag{3}$$

where $S$ is the length of the shortest path between node $i$ and node $j$ and $L$ is the diameter of the graph $G$. Note that the diameter of a graph is defined to be the length of the longest of all the shortest paths between any two nodes in the graph. The rationale for this idea is that the similarity between any two nodes should be larger when they are closer to each other and vice versa. Diameter is involved since we want the ratio to be dependent on the structure of the graph.
3) Step 2. If $C$ equals the diameter of the graph, then $M = K$ and we are done. Otherwise, set $C = C + 1$.
4) Step 3. Replace $A$ by $A + K$, where $a + b$ results in $\max(a, b)$.
5) Step 4. Find $M$ from $A$ by Eq.(3).
6) Repeat Step 1 through Step 4.

Consider Fig. 1, in which node $a$ has three paths to walk to node $e$. If $a_{13} = 1.0$ and $k_{1,3} = 0.5$, we do not replace the value of $a13 = 1.0$. However, if $a_{13} = 0.2$ and $k_{1,3} = 0.5$, we change the value of $a13$ to 0.5, i.e., $a13 = 0.5$. Fig. 2 shows the result after adding the similarity.

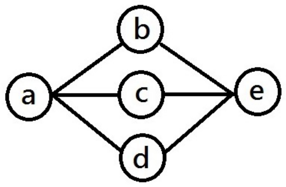

Fig. 1. A graph.

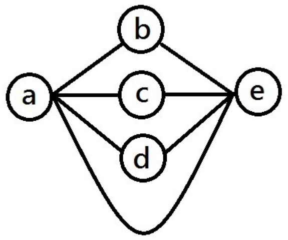

Fig. 2. Adding an edge between node $a$ and node $e$.

We use the matrix $K$ of the final iteration to construct a weighted graph $G'(V, E')$, where the edge $e'_{ij}$ in $G'$ is weighted with the corresponding element $k_{ij}$ in $K$. Then we sort $e'_{ij}$ in ascending order. In the case of $e'_{i_1 j_1} = e'_{i_2 j_2}$, we consider the degree of nodes $i_1$, $j_1$, $i_2$, and $j_2$. If the degree of $i_1$ or $j_1$ is the smallest, $e'_{i_1 j_1}$ is placed in front of $e'_{i_2 j_2}$, and vice versa. Then we delete one edge at a time successively from $G'$ until it is disconnected, i.e., separated into two disconnected components. Let $G'_1$ and $G'_2$ be the two components, and $G_1$ and $G_2$ be the subgraphs corresponding to $G'_1$ and $G'_2$, respectively, in $G$. Note that $G_1$ contains the identical nodes in $G'_1$ but the edges in $G_1$ are taken from $G$. We then check the density of $G_1$. If its density is greater than a threshold, $d_{thres}$, $G_1$ is a desired dense subgraph we are looking for. Otherwise, we proceed recursively with the above process to separate $G'_1$ into two disconnected components. When the work for $G_1$ is done, we continue with $G_2$. The algorithm of finding dense subgraphs can be described in Algorithm 1.

## III. AN EXAMPLE

We give an example here to illustrate the working of our method. Consider Fig. 3.

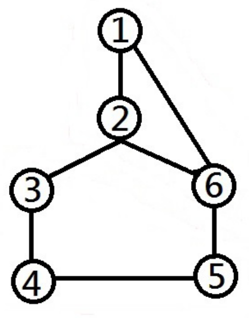

Fig. 3. A graph for illustration.

Note that the diameter of this graph is 3. Therefore, we apply Step 1 through Step 4 three times in obtaining the matrix $M$.

Firstly, the adjacency matrix $A$ is

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

**Algorithm 1** Finding Dense Subgraphs of an Undirected Graph.

```
Require: Undirected Graph $G$, density threshold $d_{thres}$;
 1: Construct the matrix $M$;
 2: Use $K$ to construct $G'$;
 3: Let $C$ be the array of tuples $(i, j, k_{ij})$, $k_{ij} > 0, i < j$;
 4: Sort $C$ in ascending order;
 5: Call FINDDENSESUBGRAPHS($G$,$G'$,$C$,$d_{thres}$);
 6: function FINDDENSESUBGRAPHS($G$,$G'$,$C$,$d_{thres}$)
 7:     $k = 0$;
 8:     while $G'$ is connected do
 9:         Delete edge $\{C[k].i, C[k].j\}$;
10:         $k = k + 1$;
11:     end while
12:     Let the connected components be $G'_1(V_1, E'_1)$ and $G'_2(V_2, E'_2)$;
13:     Let the corresponding subgraphs be $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$;
14:     Call FINDDENSESUBGRAPHS-1($G_1$,$G'_1$,$C$,$d_{thres}$);
15:     Call FINDDENSESUBGRAPHS-1($G_2$,$G'_2$,$C$,$d_{thres}$);
16: end function
17: function FINDDENSESUBGRAPHS-1($G$,$G'$,$C$,$d_{thres}$)
18:     if $d_G > d_{thres}$ then
19:         Output $G$ as a dense subgraph;
20:     elseif $|V| > 1$ then
21:         Let $C_1$ be a subarray of $C$, where $C_1[k].i \in V$ and $C_1[k].j \in V$ for all $k$;
22:         Call FINDDENSESUBGRAPHS($G$,$G'$,$C_1$,$d_{thres}$)
23:     end if
24: end function
```

The matrix $M$ by Eq.(2) is

$$M = \begin{pmatrix} 1.000 & 0 & 0.408 & 0.667 & 0.408 & 0.333 \\ 0 & 1.000 & 0.408 & 0.333 & 0.408 & 0.667 \\ 0.408 & 0.408 & 1.000 & 0.408 & 0 & 0.408 \\ 0.667 & 0.333 & 0.408 & 1.000 & 0.408 & 0 \\ 0.408 & 0.408 & 0 & 0.408 & 1.000 & 0.408 \\ 0.333 & 0.667 & 0.408 & 0 & 0.408 & 1.000 \end{pmatrix}$$

from which we have

$$K = \begin{pmatrix} 1.000 & 0 & 0.136 & 0.222 & 0.408 & 0.333 \\ 0 & 1.000 & 0.408 & 0.333 & 0.136 & 0.222 \\ 0.136 & 0.408 & 1.000 & 0.408 & 0 & 0.136 \\ 0.222 & 0.333 & 0.408 & 1.000 & 0.136 & 0 \\ 0.408 & 0.136 & 0 & 0.136 & 1.000 & 0.408 \\ 0.333 & 0.222 & 0.136 & 0 & 0.408 & 1.000 \end{pmatrix} \cdots$$

Then we compute $A = A + K$. This completes the first iteration, and the second iteration proceeds. For the third iteration, we have

$$A = \begin{pmatrix} 0 & 1.000 & 0.188 & 0.255 & 1.000 & 1.000 \\ 1.000 & 0 & 1.000 & 1.000 & 0.188 & 0.255 \\ 0.188 & 1.000 & 0 & 1.000 & 0 & 0.188 \\ 0.255 & 1.000 & 1.000 & 0 & 0.188 & 1.000 \\ 1.000 & 0.188 & 0 & 0.188 & 0 & 1.000 \\ 1.000 & 0.255 & 0.188 & 1.000 & 1.000 & 0 \end{pmatrix}$$

$$M = \begin{pmatrix} 1.000 & 0.286 & 0.569 & 0.767 & 0.488 & 0.498 \\ 0.286 & 1.000 & 0.488 & 0.498 & 0.569 & 0.767 \\ 0.569 & 0.488 & 1.000 & 0.488 & 0.364 & 0.569 \\ 0.767 & 0.498 & 1.000 & 0.488 & 0.364 & 0.569 \\ 0.488 & 0.569 & 0.488 & 1.000 & 0.569 & 0.286 \\ 0.498 & 0.767 & 0.569 & 0.286 & 0.488 & 1.000 \end{pmatrix}$$

$$K = \begin{pmatrix} 1.000 & 0.286 & 0.190 & 0.256 & 0.488 & 0.498 \\ 0.286 & 1.000 & 0.488 & 0.498 & 0.189 & 0.256 \\ 0.190 & 0.488 & 1.000 & 0.488 & 0 & 0.190 \\ 0.256 & 0.498 & 0.488 & 1.000 & 0.190 & 0.286 \\ 0.488 & 0.190 & 0 & 0.190 & 1.000 & 0.488 \\ 0.498 & 0.256 & 0.190 & 0.286 & 0.488 & 1.000 \end{pmatrix}$$

The last $K$ is used to construct $G'$. The nonzero weights of the graph are sorted as shown in Table I. Figure 4 shows

TABLE I
SORTED NONZERO WEIGHTS OF $K$

| $i$ | $j$ | $K(i,j)$ |
|---|---|---|
| 2 | 5 | 0.189 |
| 1 | 3 | 0.190 |
| 3 | 6 | 0.190 |
| 4 | 5 | 0.190 |
| 1 | 4 | 0.256 |
| 2 | 6 | 0.256 |
| 1 | 2 | 0.286 |
| 4 | 6 | 0.286 |
| 1 | 5 | 0.488 |
| 2 | 3 | 0.488 |
| 3 | 4 | 0.488 |
| 5 | 6 | 0.488 |
| 1 | 6 | 0.498 |
| 2 | 4 | 0.498 |

the result obtained by our method. Note that there are two

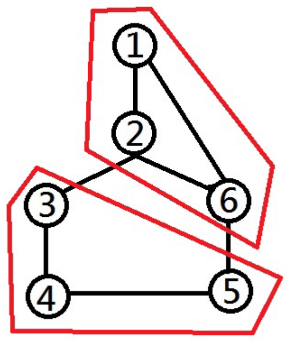

Fig. 4. The result of the example.

subgraphs identified.

## IV. Experiments and Results

In this section, we show the application of our method on some artificial and real-life data sets.

### A. *Artificial Data Set*

This section shows the results for the artificial data taken from in [8]. The characteristics of the graph randomly generated are shown in Table II. In this graph, there are three

TABLE II
CHARACTERISTICS OF THE ARTIFICIAL GRAPH

| Graph | Undirected |
|---|---|
| Whole | (100,2000) |
| Component 1 | (25,420) |
| Component 2 | (30,550) |
| Component 3 | (20,290) |

components in the graph. In this table, a pair $(v, e)$ indicates that $v$ vertices and $e$ edges are involved in an adjacency matrix. Therefore, there are $e/2$ edges in the corresponding graph or subgraph. We randomly construct the graph 30 times, by requiring that the generated graph possessing the same number of vertices and edges as shown in Table II. In this experiment, the threshold $d_{min}$ is 0.6322.

The purpose of this experiment is to show the algorithm is able to find out the dense components. Therefore, we use the $F$-score to be the evaluation measure. For each component $i$, let $V_i$ be its vertex set. We compute the $F$-score by comparing $V_i$ with the extracted result $\hat{V}$ as follows:

$$F_i = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2}{\frac{|\tilde{V}_i|}{|V_i \bigcap \tilde{V}_i|} + \frac{|V_i|}{|V_i \bigcap \tilde{V}_i|}}. \quad (4)$$

Obviously, a score close to 1 indicates that the obtained component is good. The $F$-scores obtained for this data set are shown in Table III. Note that we almost obtain the components

TABLE III
ACCURACY RESULTS

| Dense component | 1 | 2 | 3 |
|---|---|---|---|
| $F$-score | 0.9722 | 0.9883 | 0.9692 |

perfectly for this data set. The $F$-score for each obtained component is very close to 1.

### B. *Real-Life Data Sets*

In this section, we test our method on real-life data sets. Two data sets are used. The first one is a well-known network, named Zacharys karate club network [11], which is a classical social network data set from the social science literature. The data were collected from the members of a university karate club by Wayne Zachary in 1977. Each node represents a member of the club, and each edge represents a relation between two members of the club. The network is very small: it has 34 nodes and 78 undirected edges. This data set consists of two groups. The groups identified by our method in this club data set is shown in Fig. 5. According to Fig. 5, nodes 1, 2, 33, and 34 are the crucial nodes in the figure. Our

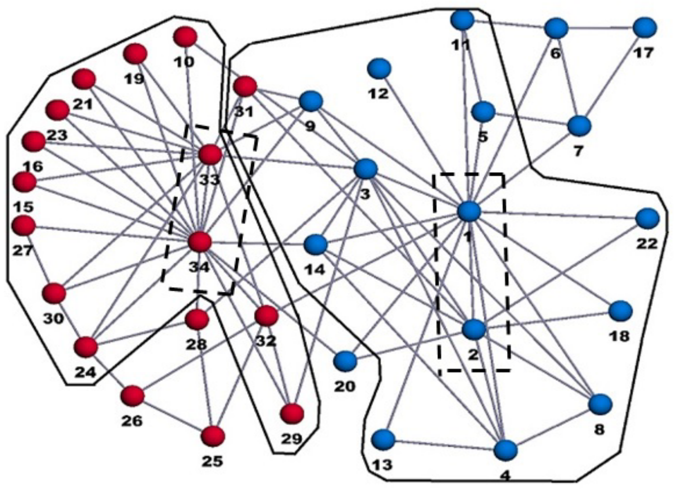

Fig. 5. The result obtained by our method in the karate club data set.

method adds additional similarity information between nodes as shown in Fig 2. Therefore, the relations among nodes can be strengthened.

The next data set is the PolBooks data set. It is about US politics, compiled by Valdis Krebs. Nodes represent books about US politics sold by the online bookseller Amazon.com, and edges mean that the ”customers who bought this book and also bought these other books” feature on Amazon. It is a directed network with 105 nodes and 441 edges. Table IV. By applying our method, two communities are identified and their $F$-scores are shown in Table IV.

TABLE IV
THE RESULTS OF POLBOOKS

| Community | 1 | 2 |
|---|---|---|
| $F$-score | 0.897989 | 0.857143 |

### C. *LFR-Benchmark Generated Data Sets*

In this section, we do experiments on the data sets with properties similar to those of the real-life data sets. Nowadays, people are interested in most of the networks, like biological networks, social networks, internet and so on. Our purpose is to try with some graphs with no noise. We use the LFR-Benchmark generator to generate networks with power-law degree distribution. By using the LFR-Benchmark generator, we create communities in high and low density, respectively. The size of the vertex set $|V|$ is 500. A high density network has the density larger or equal to 0.5, and a low density network has the density smaller than 0.5. Here, we generate 10 networks with 4 communities in high density, and 11 networks with 3 communities in low density. The threshold we set for each network is the smallest density of the communities in each network. The simulation results are shown in Table V and Table VI, respectively.

Next, we use the LFR-Benchmark generator to create a graph which has four communities. By changing the settings

TABLE V
RESULTS OBTAINED FOR HIGH DENSITY NETWORKS WITH 4 COMMUNITIES

| Community | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $F$-score | 1 | 1 | 1 | 1 |

TABLE VI
ESULTS OBTAINED FOR LOW DENSITY NETWORKS WITH 3 COMMUNITIES

| Community | 1 | 2 | 3 |
|---|---|---|---|
| $F$-score | 1 | 1 | 1 |

of the threshold and investigating the resulting scores, we can locate the best threshold that will produce the best result. Table VII shows the $F$-scores associated with each community for each specified threshold. Note that when the threshold is

TABLE VII
F-SCORES OF THE FOUR COMMUNITIES IDENTIFIED

| | Community | | | |
|---|---|---|---|---|
| Threshold | 1 | 2 | 3 | 4 |
| 0.1 | 1 | 0 | 0 | 0.6763 |
| 0.2 | 1 | 0 | 0 | 0.6763 |
| 0.3 | 1 | 1 | 0 | 0.7879 |
| 0.4 | 1 | 1 | 1 | 1 |
| 0.5 | 1 | 1 | 1 | 1 |
| 0.6 | 1 | 1 | 1 | 0.8333 |
| 0.7 | 1 | 1 | 1 | 0.4501 |
| 0.8 | 1 | 1 | 1 | 0.0995 |
| 0.9 | 1 | 1 | 0.9339 | 0.0645 |
| 1.0 | 0.2020 | 0.2912 | 0.2063 | 0.0373 |

set to be 0.4 or 0.5, we get perfect results. All the communities are perfectly identified. The average $F$-scores obtained with different thresholds are shown in Figure 6. Note that it shows the average of the $F$-scores of the four communities identified.

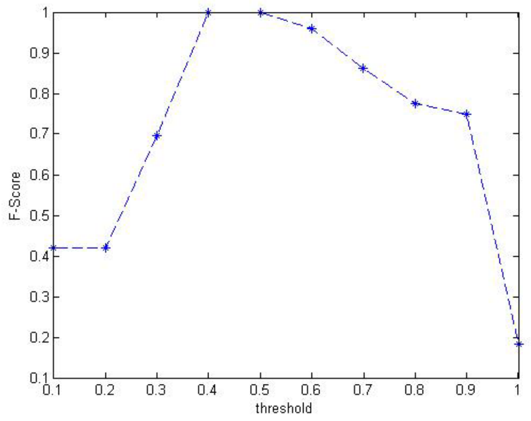

Fig. 6. Average $F$-scores obtained with different thresholds.

## V. CONCLUSION

We have proposed a method to identify meaningful dense subgraphs from a given graph. The idea of similarity transition is efficient in finding dense subgraphs. We apply punishment to the relation matrix and use it to complement the adjacency matrix several times according to the diameter of the graph. Furthermore, not every participating node in the network is required to belong to a certain community. The proposed method is demonstrated in several artificial and real-life examples. Our future work is to design a method to extract dense subgraphs in overlapping and weighted graphs. Many social networks and real-life data have overlapping structures, indicating that one node can belong to more than one community. A graph is a weighted graph if a number (weight) is assigned to each edge. We intend to modify and strengthen the method proposed in this paper in the future.

## REFERENCES

[1] D. Gibson, J. Kleinberg, and P. Raghavan, “Inferring Web Communities from Link Topology,” *Proc. Ninth ACM Conf. Hypertext and Hypermedia: Links, Objects, Time and Space (HYPERTEXT),* 1998.

[2] M.E. Newman. “Detecting Community Structure in Networks,” *European Physical J. B*, vol. 38, pp. 321-330, 2004.

[3] G.W. Flake, S. Lawrence, and C.L. Giles, “Efficient Identification of Web Communities,” *Proc. Sixth ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (KDD)*, 2000.

[4] J. Leskovec, K.J. Lang, A. Dasgupta, and M.W. Mahoney, “Statistical Properties of Community Structure in Large Social and Information Networks,” *In Proceedings of the 17th international conference on World Wide Web (WWW)*,2008.

[5] M.E.J. Newman and M. Girvan, “Finding and Evaluating Community Structure in Networks,” *Physical Review E,* vol. 69, 026113, 2004.

[6] M.E.J. Newman, “Finding Community Structure in Networks using the Eigenvectors of Matrices,” *Physical Review E,* vol. 74, No. 3, Article ID 036104, 2006.

[7] H. Shen, X. Cheng, K. Cai, and M.-B. Hu, “Detect overlapping and hierarchical community structure in networks,” *Physica A,* vol. 388, No. 8, pp. 1706-1712, 2009.

[8] J. Chen and Y. Saad, “Dense subgraph extraction with application to community detection,” *Knowledge and Data Engineering, IEEE Transactions,* vol. 24, pp. 1216-1230, 2012.

[9] P. De Meo, “Enhancing Community Detection using a Network Weighting Strategy,” *Information Sciences,* vol. 222, pp. 648-668, 2013.

[10] V. Blondel, J. Guillaume, R. Lambiotte, E. Lefebvre, “Fast Unfolding of Communities in Large Networks,” *Journal of Statistical Mechanics: Theory and Experiment,* P10008, 2008.

[11] W.W. Zachary,“An Information Flow Model for Conflict and Fission in Small Groups,” *Journal of Anthropological Research,* vol. 33, No. 4, pp. 452-473, 1977

[12] M.E.J. Newman, “Modularity and community structure in networks, *Proceedings of the National Academy of Sciences of the United States of America,* vol. 103, No. 23, pp. 8577-8582, 2006.