2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Compression Optimization Algorithm for Community Detection

Jianshe Wu, Lin Yuan, Qingliang Gong, Wenping Ma, Jingjing Ma, and Yangyang Li
The Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China
Xidian University
Xi'an, China
Email: jshwu@mail.xidian.edu.cn

***Abstract*—Community detection is important in understanding the structures and functions of complex networks. Many algorithms have been proposed. The most popular algorithms detect the communities through optimizing a criterion function known as modularity, which suffer from the resolution limit problem. Some algorithms require the number of communities as a prior. In this paper, a non-modularity based compression optimization algorithm for community detection is proposed without any prior knowledge, which is efficient and is suitable for large scale networks.**

Keywords—community detection; complex networks; compression optimazation

I. INTRODUCTION

Community detection is important in understanding the structures and functions of many real world networks [1-4]. Many efforts have been devoted to this disciplinary research field with various methods [5-9]. The most popular method for community detection is from maximizing a criterion function known as modularity (Q) [5, 6], or other modified criterion functions [7]. Along this direction, many modularity based optimization algorithms have been designed [8-12, just to name a few]. But the modularity optimization algorithms suffer from the resolution limit problem [13, 14]. In other words, small size communities cannot be detected by the modularity optimization algorithms. Detailed review of the algorithms and difficulties for community detection can be found in Ref.[3]. Thus non-modularity based algorithms are designed, e.g., the dynamics based algorithms developed in recent years [15, 16]. Some of the modularity based algorithms also require the number of communities as prior information [12], which is not always available.

In this paper, a non-modularity based fast algorithm without requiring the number of communities is provided, which can run on large scale networks with relatively low time complexity.

Our algorithm consists of two stages: compression and optimization. Thus the proposed algorithm is labeled as ComOpt for clarity.

II. PRELIMINARIES

A network is usually described by a graph $G=(V, E)$, where $V=\{v_1, v_2,\ldots, v_N\}$ is the set of nodes and E is the set of edges. Several definitions of parameters, variables, functions are introduced in this section, which are needed in description of the algorithm.

In the compression stage, the initial network is compressed by merging two or more nodes into one super-node, until the termination condition is satisfied.

Definition 1: Compression ratio: $R(G^k, G^{k+1})$. Usually, a number of steps are needed to complete the compression stage. Let $G^k=(V^k, E^k)$ denote the graph in the k-th step of compression, k=0,1, 2, … . If k=0, $G^0=G$ is the initial graph. $R(G^k, G^{k+1})$ is defined as follows,

$$R(G^k, G^{k+1}) = \frac{N(G^k) - N(G^{k+1})}{N(G^k)}, \tag{1}$$

where $N(G^k)$ is the number of nodes (include super_nodes) in G^k. Since G^{k+1} is compressed from G^k, $N(G^{k+1}) \le N(G^k)$. The value of $R(G^k, G^{k+1})$ is used to determine when to stop the compression process(see Step 3 in Section III).

Definition 2: Dense pair (v_i, v_j) [17]. Let $w(v_i, v_j)$ be the similarity between nodes v_i and v_j. If (2) holds for v_i and v_j, then $\{v_i, v_j\}$ is called a dense pair.

$$\begin{cases} w(v_i, v_j) = \max\{w(v_i, y)\}, & y \in T(v_i), \\ w(v_i, v_j) = \max\{w(v_j, x)\}, & x \in T(v_j), \end{cases} \tag{2}$$

where $T(v_i)$ is neighbor set of node v_i, similarly, $T(v_j)$ is neighbor set of node v_j.

A dense pair is a pair of nodes; the similarity between them is the largest in all the neighbors of them. Dense pair has been defined originally by Huang *et. al*. in Ref.[17] and is used to find the micro-community in the DenShrink algorithm. Here, dense pair is used to compress a pair of nodes into one super-node in ComOpt.

Definition 3: Minimum similarity threshold (*mst*). The definition of *mst* is given as follows,

$$mst = \frac{k \mid E \mid}{2 * N(G)}, \tag{3}$$

where $|E|$ is the total number of edges in the network, $N(G)$ is the total number of nodes in the network, and k is an

This work was supported by the National Natural Science Foundation of China (Nos. 61072139, 61203303, and 61272279) and the Program for New Century Excellent Talents in University (No. NCET-12-0920).

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

empirical parameter. This definition aims at giving the termination condition of the compression process. If $N(G)$=5000, k can be set to 0.2; if $N(G)$=50000, k can be set to 0.05. For a dense pair $\{v_i, v_j\}$, if $w(v_i, v_j) < mst$, they should not be compressed into a super-node.

Definition 4: Super-node. Two nodes are compressed into one super-node in our ComOpt. Given a network $G=(V, E)$, a super-node $v_i^k \in V^k$ is a connected sub-graph of G, which is composed of two nodes v_i^{k-1} and v_j^{k-1} if: 1) (v_i^{k-1}, v_j^{k-1}) is a dense pair and 2) $w(v_i^{k-1}, v_j^{k-1}) \geq mst$.

In the original graph G, each node is composed of only one node. As the compression process going on, a super-node may have lots of nodes. To some extent, one super-node represents a community. The connections in the interior of the communities are relatively dense and the connections outside the communities are relatively sparse.

Definition 5: Weight of a node, denoted as *weight* (v_i^k). *Weight* (v_i^k) is the number of the internal nodes of super-node v_i^k. Of course, the weight of an initial node is 1.

When nodes are compressed into super-nodes, in a graph G^k of the k-th step compression, the edge between two super-nodes may be merged from several edges in the initial graph G.

Definition 6: Similarity function $w(v_i^k, v_j^k)$.

$$w(v_i^k, v_j^k) = \frac{E(v_i^k, v_j^k)}{weight(v_j^k)}, \quad (4)$$

where $E(v_i^k, v_j^k)$ represents the number of edges in the initial graph G between super-nodes v_i^k and v_j^k.

In (4), super-node v_j^k is a neighbor of v_i^k. In our ComOpt, the similarity function defined by (4) is used to find the most suitable nodes to merge with super-node v_i^k.

As stated above, if $w(v_i^k, v_j^k) \geq mst$, v_i^k and v_j^k can be further merged into one super-node in the next k+1-th step of compression. By this definition, node v_i^k is prone to merge a node that $E(v_i^k, v_j^k)$ is relatively large and $weight(v_j^k)$ is relatively small. Obviously, computing of the similarity using only local information, thus the time complexity is very low.

III. Description of ComOps

As stated in Section I, ComOpt has two stages: compression and optimization.

A. Compression

The super-nodes after compression can be roughly regarded as a community. Some communities may have only one or two nodes, which are obviously not real communities. Those small communities are reassigned into other communities. Thus the partition of a network is basically completed at this stage.

Compression is the most important process in the algorithm, which directly determines the quality of the initial network partition. The main idea of this process is that two nodes with the highest similarity merge to form a super-node, the process repeats until the termination condition is satisfied.

Assume that $G^0=(V^0, E^0)$ is the initial network, here the superscript 0 means that the current network is the initial network in the 0-th layer before compression. For each node in V^0, find its dense pair that can be merged to form a new super-node, thus constitute the 1-th layer network $G^1=(V^1, E^1)$. An edge in G^1 may be an edge in G^0 or it may be merged by several edges in G^0 that connect nodes between the two super-nodes. Continue to use the lower layer network G^{k-1} to build higher layer network G^k. Detailed steps of compression are as follows:

Step 1: Choose randomly a node, e.g. v_i^0, which has not been handled.

Step 2 : Select the dense pair(v_i^0, v_j^0). At the same time, ensure that $w(v_i^0, v_j^0) \geq mst$ and v_j^0 has not been handled in this layer. Then merge v_i^0 and v_j^0 to form a new super-node v_i^1 in the next layer.

If the condition is not satisfied, then node v_i^0 remains unchanged or it can be considered to merge with itself to form a new super-node in a next layer network.

Step 3: Repeat the above steps until all nodes have been involved in constituting the super-node, then the next layer network G^1 is obtained. Similarly, G^2, G^3,…, can be obtained. The compression process of the algorithm doesn't terminate until the number of nodes in the network changes little, that is

$$R(G^i, G^{i+1}) \leq 0.05.$$

At this moment, the graph G^{end} can be considered as the initial result of the community detection. Each super-node in G^{end} is a community.

Algorithm 1 Compression

Input: Network $G^0=(V, E)$

Output: The Intermediate results of the community detection, CR=$\{C_1, C_2, \ldots\}$

```
G^0=readgrap();
while true do
  for i:=1 to number of the nodes in G^k do
  find DensePair(v_i, v_j) for node v_i.
    if w(v_i, v_j)>=mst && node v_j is not visited then
    merge nodes v_i and v_j to form a super-node v_i^{k+1},
    which belongs to the G^{k+1}
    end if
  end for
  so get the higher level graph G^{k+1}
    if R(G^k, G^{k+1})<=0.05 then
    CR= G^{k+1}
    break
    end if
  end while
  for i:=1 to number of the communities in CR
    if the number of the node in C_i<=d_min then
    delete C_i and reassign the nodes in C_i
    end if
  end for
```

In the initial result of community detection, some super-nodes have only one or two nodes. In the compression process, they may not be merged with any other node. In intuition, a community may not have only one or two nodes. The nodes in those "small communities" should be reassigned into other

communities. In this paper, the communities whose number of nodes is less than the minimum degree of the network are treated as "small communities". The nodes in those "small communities" are reassigned into other communities according to the similarity calculated by (4) to the other super-nodes.

The detail steps of reassignments are as follows:

Step1: Find the "small communities" whose number of nodes is less than the minimum degree of the network.

Step 2: Calculate the similarity by (4) of every node in the "small community" to other communities and obtain the community which has the largest similarity.

Step 3: Delete the "small communities" and reassign each node in those communities to the community which has the largest similarity.

B. Optimization

This process can be also called local search [18]. To further improve the performance of the community detection, a fast and efficient local search phase is used. The local search is modified from an efficient tabu search algorithm for graph partitioning [19]. After our modification, it also played an efficient role in improving the performance of community detection.

After a result of community detection is obtained, the local search algorithm moves appropriate nodes from a community to other communities to reduce the number of edges among communities, thus improve the performance of community partitioning.

Basically, the local search has two parts: neighborhood search and perturbation.

Algorithm 2 Optimization

Input: the Intermediate results of the community detection, CR=$\{C_1, C_2, \ldots\}$
Output: the final results of the community detection, CR=$\{C_1, C_2, \ldots\}$

```
1:   initialize the moving-gain of each boarder node to the communities
2:   for i:=1 to 0.5*N do
4:       if (i%2==0) then
5:         MoveOne();
6:       else then
7:         MoveTwo()
8:       end if
9:       if(rand()%(0.02*N)==1) then   \\ 1% of the total number of the nodes
10:      randomly choose a node v_i and move it into a different community
11:      end if
12:  end for
```

Neighborhood search. Neighborhood search tries to find a better solution from a known solution, which is composed of two moving operators, denoted as MoveOne and MoveTwo respectively. The two operators are explored in a token-ring way. That is, repeatedly apply one moving operator to the result (solution) produced by the other moving operator.

Given a community C_j of a k-community partition (k is the number of the communities). If v_i is an overlapping (border) node of C_j and v_i belongs to C_j, then moving v_i into C_j obtains a neighbor solution of the original community partition. An overlapping node means that it has at least one connection to a node in the community C_j. As the number of the overlapping nodes is very limited, so the neighborhood search is very fast.

The problem is how to evaluate the quality of the neighborhood solution. Suppose v_i is moved from C_m to C_j, the *moving-gain*(v_i, m, j) is defined for this purpose. The *moving-gain*(v_i, m, j) is the number of edges connecting v_i with C_j minus the number of the edges connecting v_i with C_m. In order to move the node into the community with the highest gain, moving-gains of all the nodes in the border set of the communities are computed. After each moving, only the relevant nodes' moving-gain is changed and should be updated.

Let $I=\{C_1, C_2,\ldots, C_k\}$ be a community k-partition, $B(C_i)$ be the set of the border nodes which are relative to the community C_i. The neighborhood search uses the following two move operators: MoveOne and MoveTwo, which are explored in a token-ring way.

MoveOne: move one highest gain node v_i.

Choose randomly a subset C_j, j=1,…, k, then select the highest gain node v_i belongs to $B(C_j)$ whose current subset is C_m. If both (5) and (6) hold, then move the selected node v_i to community C_j.

$$weight(C_m) - weight(v_i) > d_{\min}\ , \tag{5}$$

$$weight(C_j) + weight(v_i) < d_{\max}\ , \tag{6}$$

where $d_{\min}$ is the minimum degree of the network and $d_{\max}$ is the maximum degree of the network.

MoveTwo: move two highest gain nodes v_i and v_p.

MoveTwo has two steps. The first step is the same as the MoveOne. Choose randomly a community C_j and move its highest gain node v_i to C_j. The second step is as follows: choose randomly an another community C_q, $q \neq j$, then, select node v_p belongs to $B(C_q)$ whose current community is denoted as C_m, if both (7) and (8) hold, then move v_p to C_q.

$$weight(C_m) - weight(v_p) > d_{\min}\ , \tag{7}$$

$$weight(C_q) + weight(v_p) < d_{\max}\ . \tag{8}$$

MoveTwo is helpful to bring diversity into the search.

Perturbation. Since the neighborhood search move only the border nodes, it is easy to get trapped in a local optimum. Perturbation brings more diversification into the search, as far as possible to reduce the probability of trapping into local optimums.

In perturbation, we periodically move a fixed number of nodes, including non-border ones, in the following way:

Randomly choose a node v_i, whose current community is C_m. Move v_i into a randomly selected community C_j. Repeat the operator z times (z is set to 1% of the total number of the nodes in the simulations of this paper).

The perturbation process can increase the diversity of ComOpt and help to find a better solution.

IV. Experiment Results

In this section, the proposed ComOpt is evaluated by comparative experiments on the LFR benchmark datasets [20]. Three recent modularity-based algorithms are selected in the experiments: BGLL [21], DenShrink [17], and simulated annealing [22].

The LFR benchmark networks are governed by the following parameters [20].

- N: number of nodes in the network.
- $<k>$: average degree of the nodes
- k_{max}: maximum degree
- μ: mixing parameter, each node shares a fraction μ of its edges with nodes in other communities and a fraction 1-μ of its edge with intra-community nodes.
- γ: exponent for the degree distribution of nodes.
- β: exponent for the community size distribution.

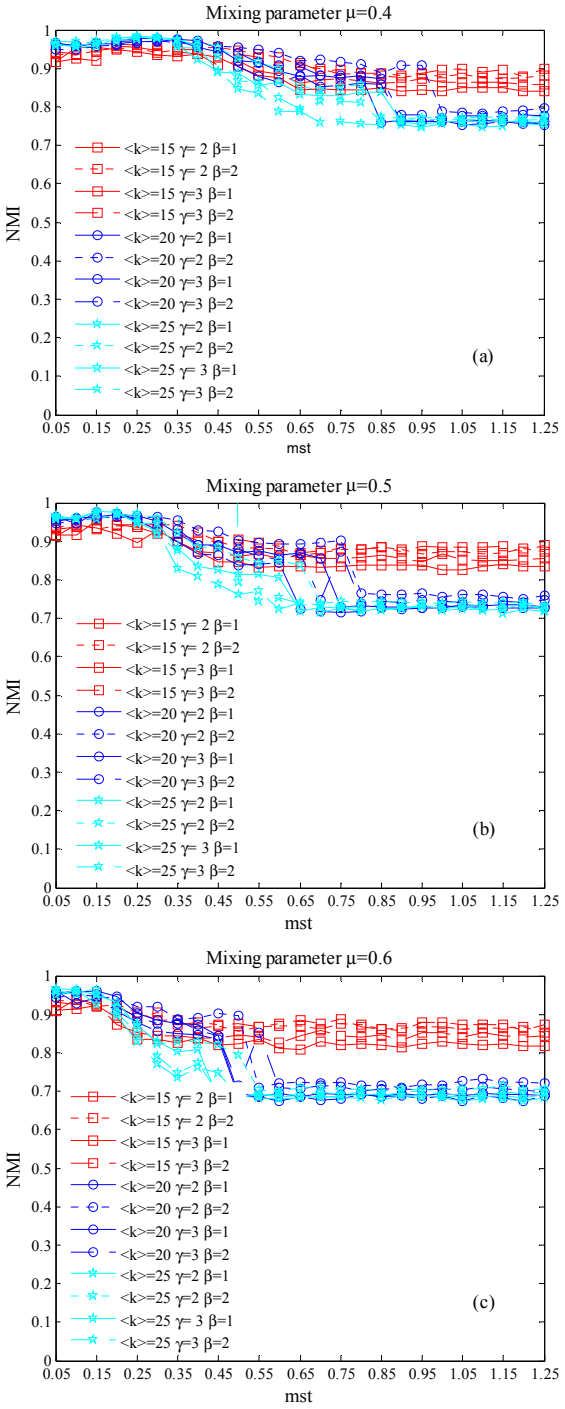

Fig. 1. The performance of ComOpt on the value of *mst*. (a) μ=0.4. (b) μ=0.5. (c) μ=0.6.

In our experiments, we adopt the Normalized Mutual Information (NMI) [10], a measurement based on the information theory, to evaluate the quality of the community detection results generated by different algorithms. NMI is widely accepted in measuring the performance of network community detection algorithms. The value of NMI is in the range from 0 to 1. NMI = 1 indicates that the network partition is perfect. In general, the higher score the NMI, the better the performance. All the experiments were conducted on a PC with a 2.2GHz, i7CPU and 8 GB of RAM.

A. Parameter sensitivity

In this subsection, experiments are done to test the performance of ComOpt on the value of *mst*, which is helpful for us to select appropriate value for *mst*.

In the experiments, N=5000 and k_{max}=50. Many networks are generated with different values of parameters. Fig. 1(a), Fig. 1(b), and Fig. 1(c) are the simulation results when μ=0.4, 0.5, 0.6, respectively. In each of the figures, three values of $<k>$ are tested (15, 20, 25), corresponding to (γ, β)= (2, 1), (γ, β)= (2, 2), (γ, β)= (3, 1), and (γ, β)= (3, 2), respectively. Thus, in each of the figures, 12 curves are shown.

When μ=0.4 (see Fig. 1(a)), in a wide range of *mst* from 0.05 to 0.45, the value of NMI is high; when μ=0.5 (see Fig. 1(b)), in the range of *mst* from 0.05 to 0.35, the value of NMI is high; when μ=0.6 (see Fig. 1(c)), in the range of *mst* from 0.05 to 0.25, the value of NMI is high. Beyond the range of *mst*, larger values of *mst* will make losing of the performance, but the effects on $<k>$=15 is smaller than that of on $<k>$=25.

From *mst*=0, as the increasing of *mst*, the values of NMI is increasing. The value of *mst* corresponding to the peak value of NMI is the most appropriate value of *mst*. Choosing an appropriate value of *mst* is dependent on μ. Smaller value of μ requires a larger value of *mst*.

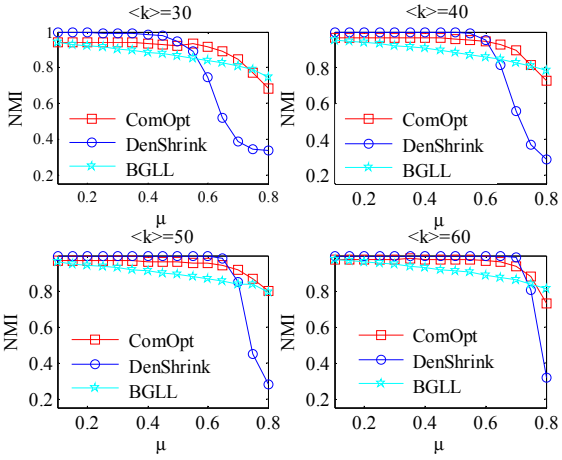

Fig. 2. Comparison of ComOpt with BGLL and DenShrink on LFR networks with N=50000, γ=2, β=1, and k_{max}=100.

B. Comparative experiments

In this subsection, two experiments are done to compare ComOpt with BGLL and DenShrink on LFR networks with N=50000 and k_{max}=100. In the first experiment, γ=2 and β=1. Four groups of networks are generated corresponding to $<k>$=30, 40, 50, and 60 respectively. The mixing parameter μ

is varying from 0.05 to 0.8 with distance 0.05. In each value of μ, three networks are generated and tested with the three algorithms. The simulation results are shown in Fig. 2.

From Fig. 2, when μ is less than 0.6, the NMI of our ComOpt is slightly lower than that of DenShrink algorithm, but is high enough and it is basically maintained at about 0.95. When μ is larger than 0.6, the NMI of DenShrink algorithm declines sharply, however, ComOpt is still able to maintain at a higher value of NMI.

In the second experiment <k>=30. Four groups of networks are generated corresponding to (γ, β)=(2, 1), (2, 2), (3, 1), and (3, 2), respectively. In each value of μ, three networks are generated and tested with the three algorithms. The simulation results are shown in Fig. 3, which indicates that the performance of ComOpt is insensitive with the variety of γ and β.

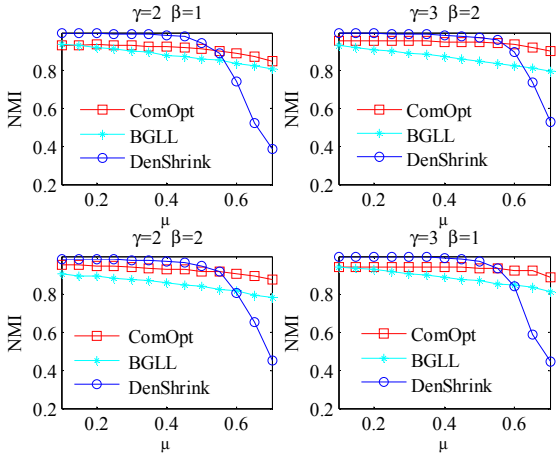

Fig. 3. Comparison of ComOpt with BGLL and DenShrink on LFR networks with N=50000, <k>=30, and k_{max}=100.

REFERENCES

[1] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Natl. Acad. Sci.* USA, vol. 99, no. 12, pp. 7821-7826, 2002.

[2] E. A. Leicht, M. E. J. Newman, "Community structure in directed networks," *Phys. Rev. Lett.* vol. 100, p.118703, 2008.

[3] S. Fortunato, "Community detection in graphs," *Phys. Rep.* vol. 486, pp. 75-174, 2010.

[4] J. Wu, X. Wang, and L. Jiao, "Synchronization on overlapping community network," *Physica A*, vol. 391, pp. 508-514, 2012.

[5] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Natl. Acad. Sci*. USA , vol. 103, p.8577, 2006.

[6] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, p.066133, 2004.

[7] Z. Li, S. Zhang, R. Wang, S. Zhang, and L. Chen, "Quantitative function for community detection," *Phys. Rev. E*, vol. 77, p.036109, 2008.

[8] Y. Li, J. Liu, and C. Liu, "A comparative analysis of evolutionary and memetic algorithms for community detection from signed social networks," *Soft Computing,* [Online]. Available: http://link.springer.com/article/10.1007/s00500-013-1060-4#.

[9] M. Gong, B. Fu, L. Jiao, and H. Du, "Memetic Algorithm for Community Detection in Networks," *Phys. Rev. E*, vol. 84, no. 5, p.056101, 2011.

[10] W. Zhan, Z. Zhang, J. Guan, and S. Zhou, "Evolutionary method for finding communities in bipartite networks," *Phys. Rev. E*, vol. 83, p. 066120, 2011.

[11] M. Gong, L. Ma, Q. Zhang, and L. Jiao, "Community Detection in Networks by Using Multiobjective Evolutionary Algorithm with Decomposition" *Physica A*, vol. 391, pp. 4050-4060, 2012.

[12] R. Shang, J. Bai, L. Jiao, and C. Jin, "Community detection based on modularity and an improved genetic algorithm," *Physica A*, vol. 392, pp. 1215-1231, 2013.

[13] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proc. Nati. Acad. Sci.* 104(1), pp. 36-41, 2007.

[14] A. Lancichinetti and S. Fortunato, "Limits of modularity maximization in community detection," *Phys. Rev. E*, vol. 84, p.066122, 2011.

[15] J. Wu, L. Jiao, C. Jin, F. Liu, M. Gong, R. Shang, W. Chen, "Overlapping community detection via network dynamics," *Phys. Rev. E*, vol. 85, p.016115, 2012.

[16] J. Wu, R. Lu, L. Jiao, F. Liu, X. Yu, D. Wang, B. Sun, "Phase transition model for community detection," *Physica A*, vol. 392, pp. 1287-1301, 2013.

[17] J. Huang, H. Sun, J. Han, and B. Feng, "Density-based shrinkage for revealing hierarchical and overlapping community structure in networks," *Physica A*, vol. 390, pp. 2160–2171, 2011.

[18] U. Benlic and J.-K. Hao, "A Multilevel Memetic Approach for Improving Graph k-Partitions," *IEEE Trans. On Evolut. Compt*., vol.15, pp. 624-642, 2011

[19] U. Benlic and J.-K. Hao, "An effective multilevel memetic algorithm for balanced graph partitioning," *in Proc. 22th Int. Conf. Tools Artif. Intell*., 2010, pp. 121–128.

[20] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E,* vol. 78, p. 046110, 2008.

[21] P. Chaturvedi, M. Dhara, and D. Arora, "Community Detection in Complex Network via BGLL Algorithm," *International Journal of Computer Applications*, vol. 48, pp. 32-42, 2012.

[22] R. Guimerà and L. A. N. Amaral, "Functional cartography of complex metabolic networks," *Nature*, vol. 433, pp. 895-900, 2005.

[23] A. Lancichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," Phys. Rev. E, vol. 80, p.056117, 2009.