IJICTR
International Journal of Information & Communication Technology Research

Volume 11- Number 4 – Autumn 2019 (48 -56)

Creating a Maximal Clique Graph to Improve Community Detection in SCoDA and OSLOM Algorithms

Sasan Sabour
School of Engineering Science
College of Engineering
University of Tehran
Tehran, Iran
sasan.sabur@ut.ac.ir

Ali Moeini*
School of Engineering Science
College of Engineering
University of Tehran
Tehran, Iran
moeini@ut.ac.ir

Received: 9 May 2019 - Accepted: 16 August 2019

Abstract—**Community detection is one of the important topics regarding complex network study. There are many community detection algorithms such as Streaming Community Detection Algorithm (SCoDA) and Order Statistics Local Optimization Method (OSLOM). However, the performance of these algorithms, in overlap communities and communities with ambiguous structure, is problematic. In community detection algorithms achieving accurate results is a challenge. In this paper, we've proposed a method based on finding maximal cliques and generating the corresponding graph in order to use as an input to SCoDA and OSLOM algorithms. Synthetic non-overlap and overlap graphs and real graphs data are used in our experiments. F1score and NM1 score functions are utilized as our evaluation criteria. We have shown that the improved version of SCoDA demonstrated better results in comparison to the original SCoDA algorithm, and the improved version of OSLOM was also superior in performance when compared with the original OSLOM algorithm.**

Keywords-Maximal clique; Maximal clique graph; OSLOM; SCoDA; Community Detection; Non-overlap community; Overlap community

I. INTRODUCTION

In the last decades, analyzing complex networks has been one of the most sought after research topics in the field of data mining. Examples of complex networks include social, biological, and technological networks [1, 2]. Community detection is one of the fundamental tasks in analyzing complex networks [1-13]. In general community structure in a network means a group of nodes with dense internal links and sparse external links as shown in Fig. 1. In this figure, a small network with three communities is shown. Community detection helps to understand and visualize the structure of a complex network. Communities in social networks like Facebook are recognized based on relations such as common friends, interest, location, etc. or in the case of Twitter based on common followers [14, 15]. Similarly, communities in protein-protein interaction networks are described as proteins with equivalent functionality inside a biological cell, also citation network communities are made by common research topics [16].

* Corresponding Author

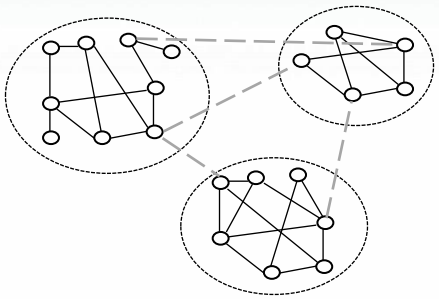

Figure 1. A small network with three communities, specified in dashed circles, in which nodes inside circles have dense internal links and external links between circles are of lower density.

Maximal clique is the densest substructure in the graph and is perhaps most commonly used as a powerful tool to find communities [6, 17]. Maximal Clique Enumeration (MCE) problem is an important part of graph theory. MCE is applied successfully in community detection and other areas such as, clustering [6, 12] and integrating different types of genome mapping data [18]. Due to the similarity between clique and community, conducting further researches on cliques has been a concern for community detection researchers [6, 12, 19].
There are two kinds of communities, non-overlap and overlap communities. If communities in the network have common members, they are overlap communities, otherwise, they are referred to as non-overlap. As an example of overlap communities, on Facebook, one person can be a member of the football team and also a university student. Recently, finding overlap communities has lately been one of the top topics of research in the field of complex networks.
[3-6, 9, 12, 20].

Accurate detection of communities is challenging. The purpose of this paper is to improve the accuracy of community detection in two well-known algorithms (SCoDA and OSLOM) through adding a pre-processing phase, which is called the maximal clique graph. By utilizing this method, the evaluation criteria demonstrated improved results for community detection in comparison with previous algorithms.

In the following, the preliminary concepts and definitions are described in Section II. In Section III related works to maximal clique and community detection can be found. Then, in Section IV, we introduce our method for community detection. Datasets, applied tools, and experimental results are presented in Section V. Section VI concludes the paper.

II. PRELIMINARY CONCEPTS AND DEFINITIONS

A network can be modeled as a graph $G = (V, E)$ which $V = \{v_1, v_2, \dots, v_n\}$ is the set of vertices and $E = \{(v_i, v_j) | v_i, v_j \in V \text{ and } i \neq j\}$ is the set of edges. The graphs we use in this paper are undirected and unweighted. A subset $C \subseteq V$ is a clique in graph G if for any pair of vertices $u, w \in C$ there exists an edge between them. A clique C is maximal if there is no vertex$u \in V - C$ that by adding it to C a larger clique can be made. Maximal clique graph is a graph that maximal cliques are vertices, and if two maximal cliques have common vertices, there would be an edge between them.

Community in a graph is a set of vertices with dense internal edges and external sparse edges. Let $C = \{c_1, c_2, \dots, c_n\}$ be a set of communities where each c_i is a community, if $c_i \cap c_j = \emptyset$ then they are non-overlap communities and if $c_i \cap c_j \neq \emptyset$ then they are overlap communities.

Average F1score is one of the evaluation metrics used in this paper, given an estimate $\acute{C}$ of the true community C, the precision and recall of this estimation are defined below:

$$Precision(\acute{C}, C) = \frac{|\acute{C} \cap C|}{|\acute{C}|}, Recall(\acute{C}, C) = \frac{|\acute{C} \cap C|}{|C|} \quad (1)$$

The F1score is defined as:

$$F1(\acute{C}, C) = 2\frac{Precision(\acute{C},C).Recall(\acute{C},C)}{Precision(\acute{C},C)+Recall(\acute{C},C)} \quad (2)$$

Consider $C = \{C_1, \dots, C_N\}$ and $\acute{C} = \{\acute{C}_1, \dots, \acute{C}_M\}$ then:

$$F1(\acute{C}, C) = \frac{1}{N}\sum_{n=1}^{N} \max_{1 \leq m \leq M} F1(\acute{C}_m, C_n) \quad (3)$$

Finally, the average F1score is defined as:

$$\overline{F1}(\acute{C}, C) = \frac{F1(\acute{C},C)+F1(C,\acute{C})}{2} \quad (4)$$

Normalized Mutual Information (NMI) is another evaluation metric used, which is based on quantifying how two covers (a cover is a collection of subsets) are similar [5]. NMI is defined as:

$$I_{norm}(X:Y) = \frac{H(X)+H(Y)-H(X,Y)}{(H(X)+H(Y))/2} \quad (5)$$

Where $H(X)$ is the entropy of the random variable X associated with the partition C', $H(Y)$ is the entropy of the random variable Y associated with the partition C'', and $H(X,Y)$ is the joint entropy. F1score and NMI are in the range [0,1] and equal 1 when two communities are exactly coincident.

III. RELATED WORK

Many algorithms are proposed for detecting communities in networks [21]. Some of these algorithms depend on optimizing an objective function that measures the quality of the detected communities [13]. Other popular algorithms use random walks [22], statistical inference [23], clique percolation [20], or spectral clustering [24].

Pala et al. [20] proposed the Clique Percolation Method (CPM) based on the concept that high density internal links in the communities are similar to cliques. Pala et al. used the concept of k-clique. K-clique is a complete sub-graph with the size of k that has edges between all k vertices. If two k-cliques share k-1 vertices, they are adjacent. The largest connected sub-graph obtained by the union of k-clique and all k-cliques connected to it is called the k-clique community. Dereny et al. [19] studied the percolation properties of k-cliques in random graphs. In [12], based on a multi-objective evolutionary approach a pruning algorithm for maximum clique is proposed. Also, in [6], for detecting overlapping communities, a pruning algorithm for maximum clique is introduced.

In the next three subsections, we first present maximal clique enumeration (MCE) algorithms. Secondly, we describe the SCoDA algorithm. And finally, OSLOM algorithm is introduced.

A. Maximal Clique Enumeration

Algorithms that attempt to solve Maximal Clique Enumeration (MCE) are categorized as sequential MCE and parallel MCE. In sequential MCE, Bron, and Kerbosch [25] present an algorithm based on depth-first-search, Kose et al. [26] use breadth-first-search, and Modany and Dey [27] utilize pruning strategies for enumerating large cliques. In Parallel MCE, Zhang et al. [28] developed an algorithm based on breadth-first-search, Due et al. [29] present a parallel algorithm based on output-sensitive algorithms, and in [17], authors mine maximal cliques using a distributed MapReduce algorithm.

B. SCoDA algorithm

Here, we describe how SCoDA [10] algorithm works, which is defined in Algorithm 1. The input of the algorithm is a list of graph edges and a parameter $D \geq 1$. Two arrays of size m are built to store the degrees of nodes and the community that the nodes belong to. These arrays in the algorithms are called d and c respectively. At the beginning of the algorithm, $d_i = 0$, and $c_i = i$ for all i. In line 5, the list of edges is shuffled. The main loop starts from line 6 and iterates over the edges, for each edge $e_j = (u, v)$, the degrees of u and v are updated. If the degree of both u and v are lower than threshold D, the node with the lower degree is added to the other nodes community. Otherwise, communities do not change.

Algorithm 1 SCoDA

Input: List of edges E between nodes {1,…,n} and parameter $D \geq 1$
Output: Detected communities $(\acute{c}_i)_{i=1,\dots,n}$
1: **For** all i=1,…, n, $d_i \leftarrow 0$ and $c_i \leftarrow i$
2: shuffle the list of edges E
3: **for** j=1,…, $|E|$ **do**
4: $(u, v) \leftarrow j^{th}$ edge of E
5: $d_u \leftarrow d_u + 1$ and $d_v \leftarrow d_v + 1$
6: **if** $d_u \leq D$ and $d_v \leq D$ **then**
7: **if** $d_u \leq d_v$ **then** $c_u \leftarrow c_v$
8: **else** $c_v \leftarrow c_u$
9: **end if**
10: **end if**
11: **end for**
12: **return** $(\acute{c}_i)_{i=1,\dots,n}$

C. OSLOM algorithm

OSLOM [8] estimates the significance of clusters with statistical tools. The statistical significance is the probability of finding a similar community (same degree sequence, size, and internal connections) in an empty model that holds no community structure. OSLOM has three phases mentioned below:

1- Searching for significant clusters, until convergence
2- Analyzing the result of phase 1, trying to detect their internal structure or possible unions of the set of clusters
3- Finally, Detecting the hierarchical structure of the clusters

In the first phase, the algorithm agglomerates neighbor nodes to create a collection of significant, probably overlapping communities, then, by removing from or adding to communities, tries to increase their significance. This process is repeated until stability is ensured. Different hierarchical levels are obtained by using the same process for the super-network where nodes represent communities.

IV. Proposed Method

In this section, we describe our proposed method. In the next three subsections, we describe the maximal clique graph method, MSCoDA, and MOSLOM algorithms, respectively.

A. Maximal clique graph method

The model diagram of the proposed method and its corresponding algorithm are illustrated in Fig. 2 and Algorithm 2, respectively. Overlapping and non-overlapping synthetic graphs and real graphs are used for comparing our method with other algorithms. These graphs all are undirected, which means if we have an edge between (u, w), there is also an edge between (w, u).

The proposed method adds a pre-processing phase before feeding the graph as an input to SCoDA and OSLOM algorithms. In the first step of our method, we find all maximal cliques of the graph. To achieve this, we used the *find_cliques* function, as shown in line 2 of Algorithm 2.

The second step begins with making the maximal clique graph. The output of the first step is a text file containing all maximal cliques of the graph. Each line in this file is a maximal clique. We numbered each line starting from 1 to *n*, where *n* is the number of maximal cliques of the graph, then saved the file as maximal clique graph vertices. In the maximal clique graph vertices file, there are two columns: id and maximal cliques. To create the maximal clique graph edges file, we compare the second column of each row with each other to find vertices that share common maximal cliques. Finally, we save the *id* of these vertices in a new file called *EdgeMaximalCliqueGraph*.

In the third step, we set the maximal clique graph edge list as input to the SCoDA and OSLOM. The output of these two algorithms is the communities list.

Finally, in the last step, we replace vertices in the communities list with the original vertices (maximal cliques). The output of the fourth step is the final communities detected by this method. Algorithm 2 shows the maximal clique graph method.

Algorithm 2 Creating maximal clique graph

Input: List of edges E between nodes
Output: a file containing the edges of maximal clique graph

```
1: G= read_edgelist
2: maximal= find_cliques(G)
3: for clique in maximal do
4:     id ← id + 1
5:     write (id,clique) as VertexMaximalCliqueGraph
6:     for vertex in clique
7:         Edges (vertex).Add (id)
8:         CliqueGraph(id).Add(vertex)
9: for i=1,…,id do
10:  for vertex in cliqueGraph(i)
11:    for edge in Edges(vertex)
12:      if edge > i
13:        write (x,edge) as EdgeMaximalCliqueGraph
14: return EdgeMaximalCliqueGraph
```

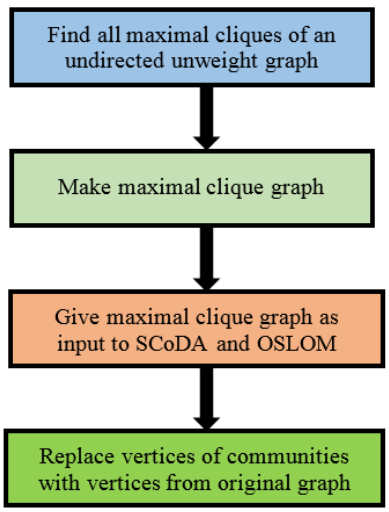

Figure 2. Model diagram of the proposed method.

B. MSCoDA algorithm

SCoDA is a community detection algorithm with short execution time. However, its weakness is when we test it out on overlap communities with complex community structure and synthetic graphs. Therefore, the pre-processing maximal clique graph method was added to the SCoDA algorithm, in order to improve community detection. The purpose is to make the structure of complex graphs easier for the algorithm.

Here, we describe how the MSCoDA algorithm works, as shown in Algorithm 3. MSCoDA uses the SCoDA algorithm. The algorithms input is a list of the graphs edges. First, the algorithm creates a maximal clique graph by using Algorithm 2. The list of maximal clique graph edges is called E_{new}. Then the SCoDA algorithm is performed on E_{new} to detect communities. Finally, detected communities' vertices should be replaced with the original graph vertices.

Algorithm 3 MSCoDA

Input: List of edges E between nodes {1,…,n}
Output: Detected communities of the original graph

```
1: create maximal clique graph by using Algorithm 2
2: E_new ← list of EdgeMaximalCliqueGraph
3: run community detection algorithm SCoDA on E_new
4: for community in communities
5:   for vertex in community
6:     replace(vertex,VertexMaximalCliqueGraph(vertex))
```

C. MOSLOM algorithm

MOSLOM uses the OSLOM algorithm. MOSLOM has three phases mentioned below:

1- Create maximal clique graph by Algorithm 2.
2- Run OSLOM algorithm on maximal clique graph.
3- Replace detected communities' vertices with original network vertices.

In the first phase, the algorithm creates a maximal clique graph. In the next phase, the output of the previous stage is given to the OSLOM algorithm as input. Finally, vertices of the detected communities will be replaced with original graph vertices. MOSLOM is more accurate compared to the OSLOM algorithm in finding overlap communities in graphs with complex community structure. However, the MOSLOM algorithm in large graphs could fail or suffer from a long execution time.

V. DATASETS AND EXPERIMENTAL RESULTS

In this section, we first introduce datasets used in this paper and then show the experimental results of our method in comparison with other algorithms.

A. Datasets

The datasets we used in this paper can be divided into two categories: synthetic graphs and real graphs. Both synthetic and real graphs in this paper include ground-truth communities that are used to measure the detection quality. The Lancichinetti-Fortunato-Radicchi (LFR) [3] is used to make synthetic graphs, and input parameters are $LFR(N, k, k_{max}, \tau_1, \tau_2, c_{min}, c_{max}, \mu, O_n, O_m)$as described in Table 1.

TABLE I. PARAMETER AND THEIR DEFINITION OF LFR.

Parameters	Definition
N	Number of nodes
K	Average node degree
k_{max}	Maximum node degree

τ_1	Power law distribution of the node degrees
τ_2	Power law distribution of the community sizes
c_{min}	Minimum size of each community
c_{max}	Maximum size of each community
μ	$\mu \in [0,1]$, controls the average ratio of the external links to the total links of each node. If $\mu = 0$, all the edges in the graph are external links. If $\mu = 1$, all the edges in the graph are internal links. In other words, a larger μ means a more ambiguous community structure.
O_n	Number of overlapping nodes. Higher value of O_n makes a more ambiguous community structure.
O_m	Number of communities that each node belongs to. By increasing the O_m value, the detection problem becomes more difficult.

We used two kinds of synthetic graphs, non-overlap, and overlap. For the non-overlap synthetic graphs, we set the parameters according to values displayed in Table 2.

TABLE II. IN THE GRAPH NAME COLUMN, S STANDS FOR SMALL AND B FOR BIG COMMUNITY RESPECTIVELY.

graph name	N	K	k_{max}	τ_1	τ_2	c_{min}	c_{max}
1000b	1000	20	50	2	1	20	100
1000s	1000	20	50	2	1	10	50
5000b	5000	20	50	2	1	20	100
5000s	5000	20	50	2	1	10	50
50000	50000	20	200	2	1	20	1000
100000	100000	20	200	2	1	20	1000

Also, parameters set for overlap synthetic graphs are presented in Table 3.

TABLE III. PARAMETERS VALUE FOR OVERLAP GRAPH.

graph name	μ	O_n	other parameters
a	0.1	0.1N	N=1000
b	0.3	0.1N	
c	0.5	0.1N	$c_{min} = 20$, $c_{max} = 50$
d	0.1	0.3N	
e	0.3	0.3N	$\tau_1 = 2$, $\tau_2 = 1$
f	0.5	0.3N	
g	0.1	0.5N	$k = 20$, $k_{max} = 50$
h	0.3	0.5N	
i	0.5	0.5N	$O_m = \{2,4,6,8\}$

The real graphs used in this article are from Stanford Social Network Analysis Project (SNAP [31]). The real graphs used are described in Table 4.

TABLE IV. REAL GRAPHS.

graph name	$\|V\|$	$\|E\|$	communities
Amazon [10]	334863	925872	311782
DBLP [10]	317080	1049866	1449666
polBooks [9]	105	441	3
polBlogs [9]	1490	16726	2

[1] https://github.com/networkx/networkx

B. Benchmark algorithms

For evaluating the MOSLOM and MSCoDAs performance, a wide range of state-of-the-art algorithms have been used:

- **Infomap:** By compressing information flow generated by random walks, splits the graph into modules [32]
- **Louvain:** based on the optimization of modularity metrics [1]
- **OSLOM:** optimizing a fitness function which measures the statistical significance of a community [8]
- **SCoDA:** a streaming algorithm based on vertex degree [10]

C. Experimental Results

The experiments were performed on the parallel processing lab of the Faculty of Mathematics of the University of Tehran with 1280 GB of RAM, 4 TB of disk space, 336 computational cores Intel Xeon with 2.3 GHz speed. For finding maximal cliques in a network, we used Networkx as our software tool [30]. Networkx is a Python language software package for analyzing graphs. With this python library, one can create, manipulate, and study the structure of complex networks. Networkx is open source software and available on GitHub[1].

The Networkx library was used for making maximal cliques. Also, we used Python 2.7.11 for coding two parts of our method: making the maximal clique graph and replacing vertices of detected communities with original vertices. We used implementation programs based on C++ provided by the authors of SCoDA [10] and OSLOM [8]. The scoring functions, NMI and F1score, are implemented in C++, and we used these implementations which were provided by their authors respectively in [5] and [2]. We tested our new method on synthetic non-overlap graphs, synthetic overlap graphs, and real graphs. Maximal SCoDA (MSCoDA) and maximal OSLOM (MOSLOM) are both tested on these graphs. We also evaluated our method with NMI and F1score. In the next three subsections, we show our experimental results.

C.1 Synthetic graphs with overlap communities

Results of comparing MSCoDA and MOSLOM with other algorithms on F1score and NMI scores are illustrated in Fig. 3 and Fig 4. As we can see in the figures, MOSLOM had better results on synthetic graphs with different combinations of μ, O_n, O_m, and N, as displayed. For example, in Fig 3, when O_n, O_m, and N are set to 0.3N, 8 and 1000, respectively, the value of F1score is 0.679 and even when μ is increased, MOSLOM performs better compared to other algorithms. Also, when μ is increased $O_n = 0.5N$ and N=1000, the improvement of MOSLOM in

comparison to other algorithms based on F1score criteria is in the range 0.047 and 0.161.

In Fig 4, when O_n, O_m, and N are set to 0.3N, 8, and 1000 the value of NMI is 0.547 and even when μ is increased MOSLOM is at least 0.015 improved. Also, when O_n and N are set to 0.5N and 1000 respectively, with an increasing μ, MOSLOMs maximum and minimum improvement are 0.286 and 0.035, respectively.

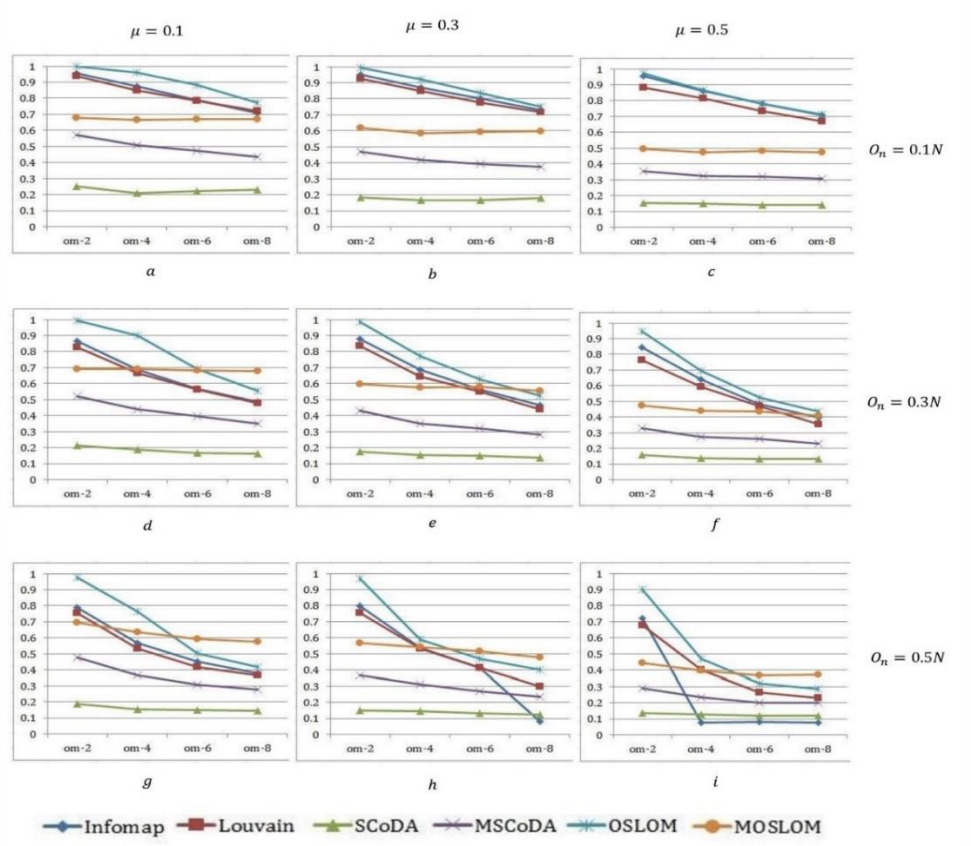

Figure 3. F1score result on graphs with overlap communities. F1score versus O_m.

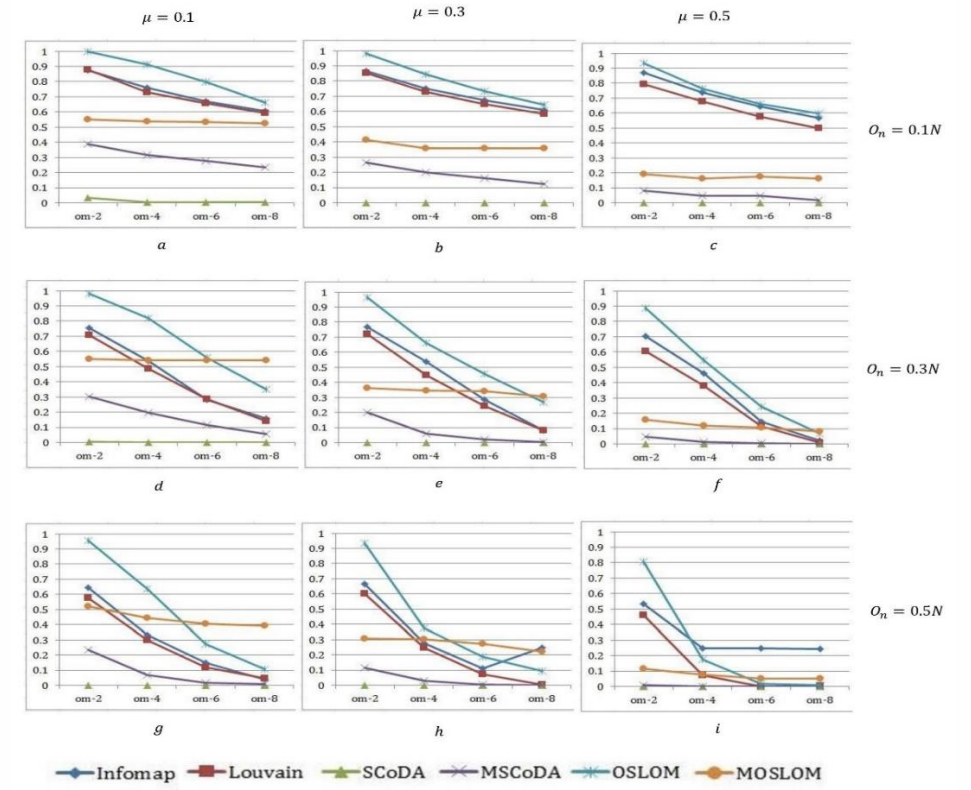

Figure 4. NMI result on graphs with overlap communities. . NMI versus O_m.

C.2 Real graphs

Results of comparing MSCoDA with other algorithms on F1score and NMI scores are illustrated in Fig. 5.

F1score and NMI of MSCoDA in the amazon graph are respectively, 0.468 and 0.188, in dblp are respectively 0.384 and 0.16. Also, MSCoDA outperformed SCoDA when the improvement of F1score and NMI values in polBooks are respectively, 0.133 and 0.023, and in polBlogs are respectively, 0.088 and 0.049.

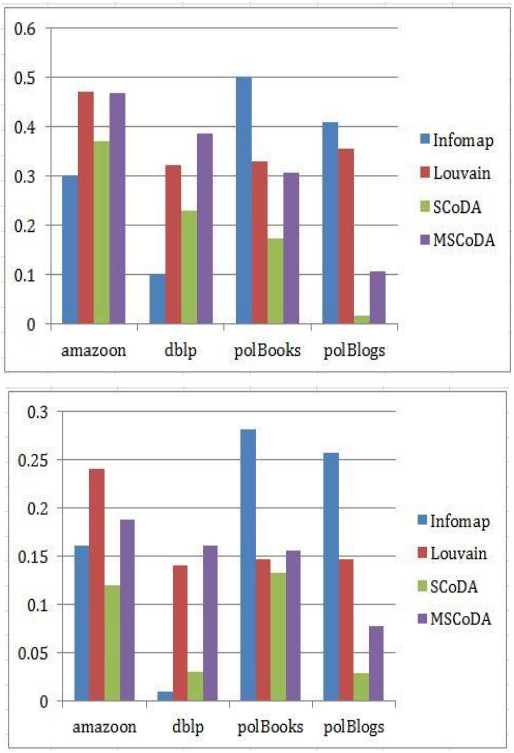

Figure 5. F1score (upper) and NMI (down) result on real graphs.

C.3 Synthetic graphs with non-overlap communities

Results of comparing MSCoDA and MOSLOM with other algorithms on F1score are illustrated in Fig 6. As seen in graph 1000b and 1000s with a threshold of $\mu = 8$, MOSLOM produces more accurate results when the graph has a higher level of structural complexity.

In Fig. 6, when $\mu = 9$ the F1score improvement of MSCoDA in 5000b is 0.094 and 0.163 in 5000s. Also, MSCoDA outperformed SCoDA, when comparing F1score values in 50000 and 10000 graphs with an increasing μ. The range of improvement in 50000 is between 0.042 and 0.131, in 100000 is 0.044 and 0.133, respectively.

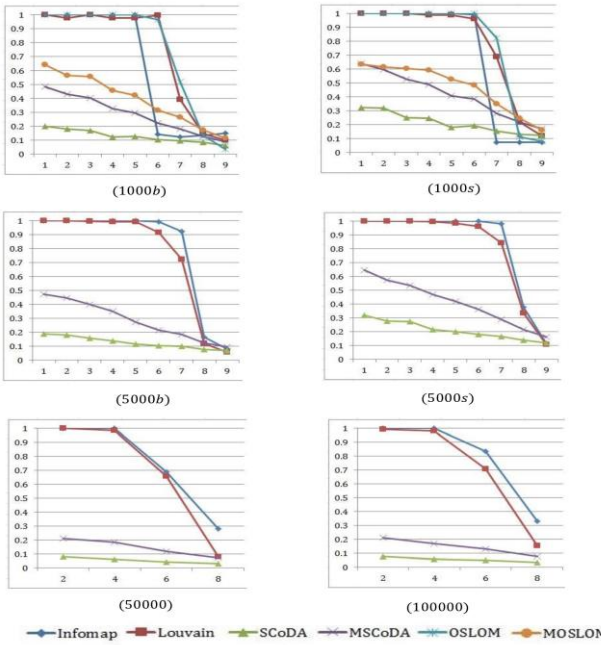

Figure 6. F1Score versus μ on graphs with non-overlap communities.

D. Complexity analysis

The complexity analysis of the proposed method contains the analysis of its four parts:

In the first part, finding maximal cliques which is an NP-Complete problem can be done by parallel, greedy and heuristic algorithms in an appropriate time for some graphs.

In the second part, creating maximal clique which is done in the main loop of the algorithm with two inner loops, vertices of the maximal clique are extracted in the first inner loop, and in the second inner loop the edges are created.

The time complexity of the first inner loop is:
$\boldsymbol{Time\ Complexity(first\ loop)}$: $maximal\ cliques \times vertices\ of\ each\ clique$.
It means that in the worst case the complexity of this part is $O(n^2)$,where n is the number of vertices of the graph. The time complexity of the second loop is:

$\boldsymbol{Time\ Complexity(second\ loop)}$:
$maximal\ cliques \times vertices\ of\ each\ clique \times edges\ of\ each\ vertex$.

It means that in the worst case, the complexity of this part is $O(n^3)$, where n is the number of vertices of the graph. It can be deduced that the time complexity of the second part depends on the time complexity of the second loop.

In the third part, there are SCoDA and OSLOM algorithms. The time complexity of the SCoDA algorithm is linear in m (number of edges). The time complexity of the OSLOM algorithm depends on the specific features of the community structure, therefore cannot be estimated exactly.

In the last part the vertices in communities are replaced by the original vertices. The complexity of this part is:

Time Complexity: *detected comunities* × *vertices of each comunity*.

The execution times of the proposed method versus O_m, with different μ and O_n are shown in Fig.7. As seen in graphs, the execution time difference between MSCoDA and SCoDA is not significant.

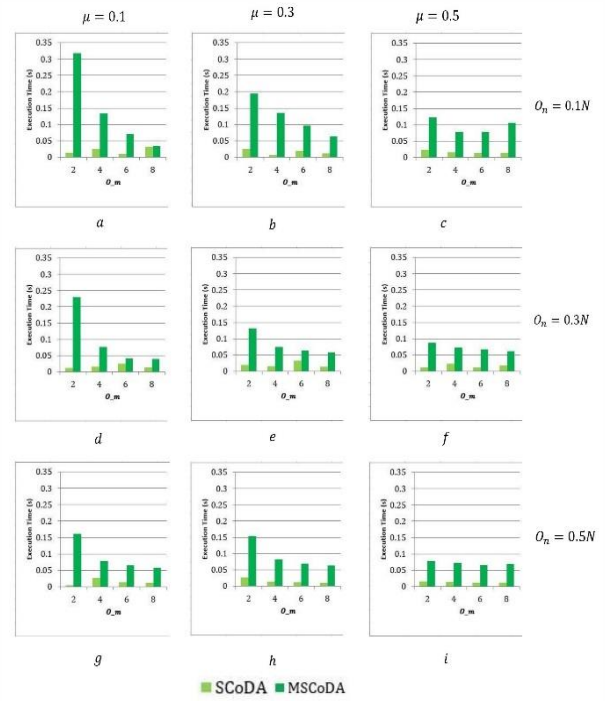

Figure 7. Execution time of SCoDA and MSCoDA algorithms.

Although not shown in the figures, we realized that the execution time of MOSLOM is longer than OSLOM. For example, if the execution time for a graph with OSLOM takes 100 seconds, the MOSLOM takes 1000 seconds for the same graph.

VI. Conclusion and Discussion

We introduced a method for detecting communities in graphs. Our method is based on detecting maximal cliques and making the maximal clique graph. We added a pre-process phase – making a maximal clique graph - to SCoDA and OSLOM algorithms and named the new algorithms: MSCoDA and MOSLOM. Our purpose of adding this pre-process phase was to improve the accuracy of finding communities in graphs with overlap communities and with complex community structure. We tested our new method on non-overlap and overlap synthetic graphs and real graphs. Our evaluation method was F1score and NMI score functions. We demonstrated that our new method, MSCoDA outperformed the SCoDA in all of the test graphs, and as ambiguity increased in graphs, MOSLOM produced better results compared to other algorithms. Also in real graphs, MSCoDA demonstrated better results than other algorithms.

Finding maximal cliques and creating their maximal clique graph made detecting communiteis easier in graphs (with ambiguous structure). This preprocessing allows edges to connect the graphs maximal cliques, which in turn decreases the complexity of the connections.

Networks where maximal cliques can be extracted in a reasonable execution time and networks with dense structure, are best suited to these algorithms. The execution time of MOSLOM is very long in comparison with OSLOM. Since finding maximal cliques in graphs is NP-Complete, MOSLOM has long execution time in huge graphs.

For future work, one can change the maximal clique graph creation algorithm. One could also use a distributed algorithm to find the maximal clique graph in order to reduce the time complexity of the propsed algorithm.

References

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[2] A. Prat-Pérez, D. Dominguez-Sal, and J.-L. Larriba-Pey, "High quality, scalable and parallel community detection for large real graphs," in *Proceedings of the 23rd international conference on World wide web*, 2014, pp. 225-236: ACM.

[3] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Physical Review E*, vol. 80, no. 1, p. 016118, 2009.

[4] H. Shen, X. Cheng, K. Cai, and M.-B. Hu, "Detect overlapping and hierarchical community structure in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 8, pp. 1706-1712, 2009.

[5] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, no. 3, p. 033015, 2009.

[6] B. Pattabiraman, M. M. A. Patwary, A. H. Gebremedhin, W.-k. Liao, and A. Choudhary, "Fast algorithms for the maximum clique problem on massive graphs with applications to overlapping community detection," *Internet Mathematics*, vol. 11, no. 4-5, pp. 421-448, 2015.

[7] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[8] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, "Finding statistically significant communities in networks," *PloS one*, vol. 6, no. 4, p. e18961, 2011.

[9] M. Hajiabadi, H. Zare, and H. Bobarshad, "IEDC: An integrated approach for overlapping and non-overlapping community detection," *Knowledge-Based Systems*, vol. 123, pp. 188-199, 2017.

[10] A. Hollocou, J. Maudet, T. Bonald, and M. Lelarge, "A linear streaming algorithm for community detection in very large networks," *arXiv preprint arXiv:1703.02955*, 2017.

[11] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *nature*, vol. 466, no. 7307, p. 761, 2010.

[12] X. Wen *et al.*, "A maximal clique based multiobjective evolutionary algorithm for overlapping community detection," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 363-377, 2017.

[13] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577-8582, 2006.

[14] H. Fani and E. Bagheri, "Community detection in social networks," *Encyclopedia with Semantic Computing and Robotic Intelligence*, vol. 1, no. 01, p. 1630001, 2017.

[15] M. Hamdaqa, L. Tahvildari, N. LaChapelle, and B. Campbell, "Cultural scene detection using reverse Louvain optimization," *Science of Computer Programming*, vol. 95, pp. 44-72, 2014.

[16] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821-7826, 2002.

[17] M. Svendsen, A. P. Mukherjee, and S. Tirthapura, "Mining maximal cliques from a large graph using mapreduce: Tackling highly uneven subproblem sizes," *Journal of Parallel and distributed computing*, vol. 79, pp. 104-114, 2015.

[18] E. Harley, A. Bonner, and N. Goodman, "Uniform integration of genome mapping data using intersection graphs," *Bioinformatics*, vol. 17, no. 6, pp. 487-494, 2001.

[19] I. Derényi, G. Palla, and T. Vicsek, "Clique percolation in random networks," *Physical review letters*, vol. 94, no. 16, p. 160202, 2005.

[20] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *nature*, vol. 435, no. 7043, p. 814, 2005.

[21] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75-174, 2010.

[22] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *International symposium on computer and information sciences*, 2005, pp. 284-293: Springer.

[23] A. Lancichinetti, F. Radicchi, and J. J. Ramasco, "Statistical significance of communities in networks," *Physical Review E*, vol. 81, no. 4, p. 046110, 2010.

[24] D. A. Spielman and S.-H. Teng, "Spectral partitioning works: Planar graphs and finite element meshes," *Linear Algebra and its Applications*, vol. 421, no. 2-3, pp. 284-305, 2007.

[25] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575-577, 1973.

[26] F. Kose, W. Weckwerth, T. Linke, and O. Fiehn, "Visualizing plant metabolomic correlation networks using clique–metabolite matrices," *Bioinformatics*, vol. 17, no. 12, pp. 1198-1208, 2001.

[27] N. Modani and K. Dey, "Large maximal cliques enumeration in sparse graphs," in *Proceedings of the 17th ACM conference on Information and knowledge management*, 2008, pp. 1377-1378: ACM.

[28] Y. Zhang, F. N. Abu-Khzam, N. E. Baldwin, E. J. Chesler, M. A. Langston, and N. F. Samatova, "Genome-scale computational approaches to memory-intensive applications in systems biology," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, 2005, p. 12: IEEE Computer Society.

[29] N. Du, B. Wu, L. Xu, B. Wang, and X. Pei, "A parallel algorithm for enumerating all maximal cliques in complex network," in *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*, 2006, pp. 320-324: IEEE.

[30] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using NetworkX," Los Alamos National Lab.(LANL), Los Alamos, NM (United States)2008.

[31] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181-213, 2015.

[32] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118-1123, 2008.

Sasan Sabour Received B.Sc. degree in Software Engineering from Razi University, Kermanshah, IRAN in 2014. He also received his M.Sc. degree from the University of Tehran, Tehran, IRAN in 2018 in Algorithms and Computations. His research interests include Data Mining, Machine Learning, Social network, and the Internet of Things.

Ali Moeini received his Ph.D. in Nonlinear Systems Analysis at the University of Sussex, UK, in 1997. He is full Professor in Department of Algorithms and Computation, School of Engineering Science, College of Engineering, University of Tehran. His research interests include Mining of Massive Datasets, Randomized Algorithms, Semantic Web, and Business Analytics.