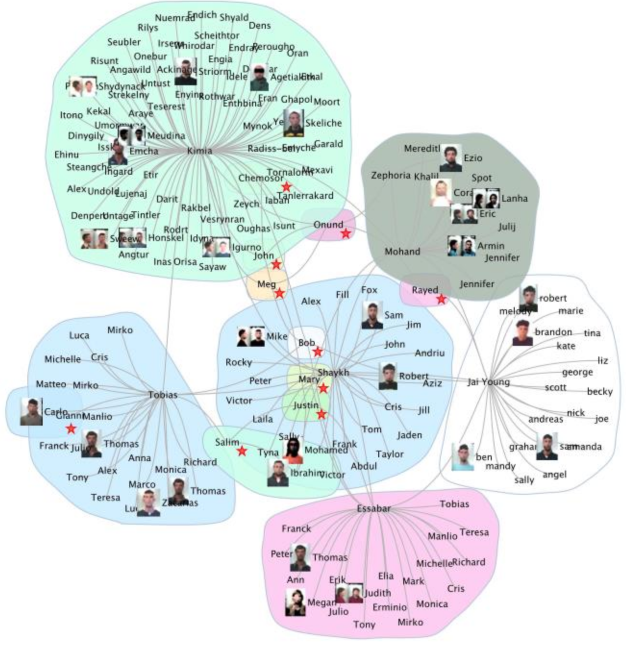


A density based link clustering algorithm for overlapping community detection in networks

by Xu Zhou

STUDENT NAME

Raghda Al-Taei ID:94231103

Teacher name:

Dr. Maryam Amir Haeri

CNA course

Contents

1. Introduction 2

2. Description of the DBLC 3

3. Edge similarity 6

4. Experimental Results 10

References 15

Table of figures

Figure 1 Algorithm one of the DBLC..4
Figure 2 Edge Expansion..5
Figure 3 Algorithm 3 updating strategy..6
Figure 4 NMI results of our implementation 10
Figure 5 NMI results mentioned in the paper 11
Figure 6 Precision results of our implementation 12
Figure 7 Precision results of the paper 12
Figure 8 Recall results of our implementation 13
Figure 9 Recall results of the paper 13
Figure 10 Fscore results of our implementation 14
Figure 11 Fscore results of the paper 14

1. Introduction

Community structure in networks tends to be overlapped (nodes which belong to multiple communities). The previous work that has been done using a link clustering method for overlapping community detection, has led to the following results which are not applicable in most cases:

- cluster bridge edge
- extreme overlapping
- assign border edge mistakenly to adjacent communities.

Due to Xu Zhou [1] This problem was solved by using a density based link clustering algorithm (DBLC) it was mostly applied in data mining because it inclines to be simple and gives good results in the case of different clustering sizes and shapes even with the existing of noise. But in the case of community detection, the density-based clustering algorithm is capable of distinguishing only non-overlapping structures. So by 'taking density based clustering algorithm to cluster links instead of nodes in networks, then the overlapping community structures could be found'.

2. Description of the DBLC

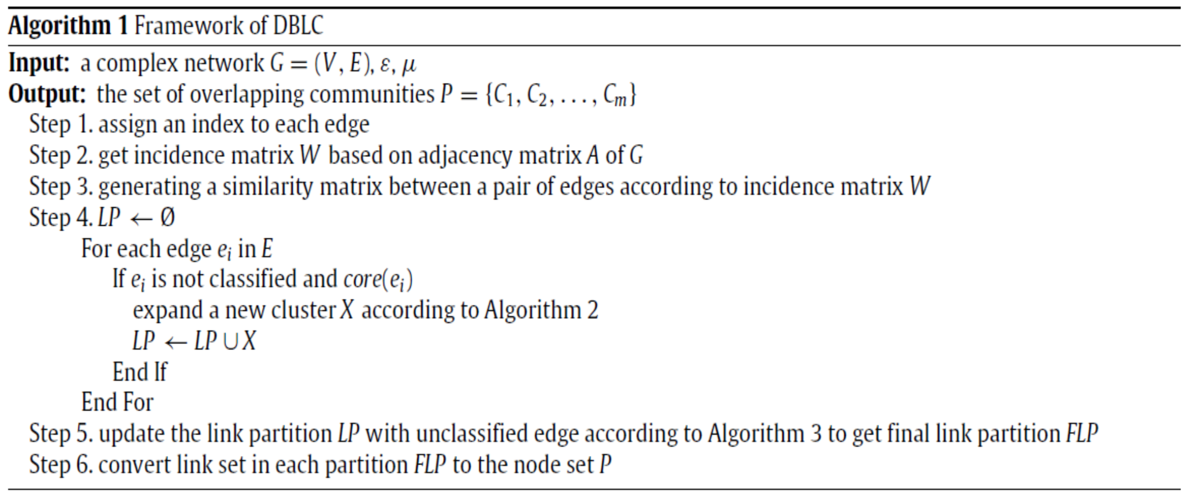
The DBLC method has the following advantages than the other older methods:

- where the links are processed it is not sensitive to the order of nodes being selected
- having a similarity computation formula used to assign similarity between links that share a common node.
- DBLC can detect overlapping community structures effectively.
- Able to identify noise edges to avoid extreme overlapping.
- performs well even when a number of communities that each overlapping node belongs to are large.
- It overcomes the matter of bridge edge and border edge incorrectly fitting to adjacent communities.

outline of DBLC includes 6 steps

1. Initialization of the input network.
2. Creating the incidence matrix W, based on the adjacency matrix A.
3. Generating the similarity matrix of all the pairs of edges, according to the incidence matrix W (will be described in coming sections).
4. Expansion step (clusters of only core edges are created at the end of this Step).

5. Updating strategy to deal with the unclassified border links (required to transfer the link cluster into a community of nodes, according to the incidence matrix).
6. Convert clusters of edges (links) to the community of nodes (each node might be connected to multiple links, so a node might belong to more than one community).



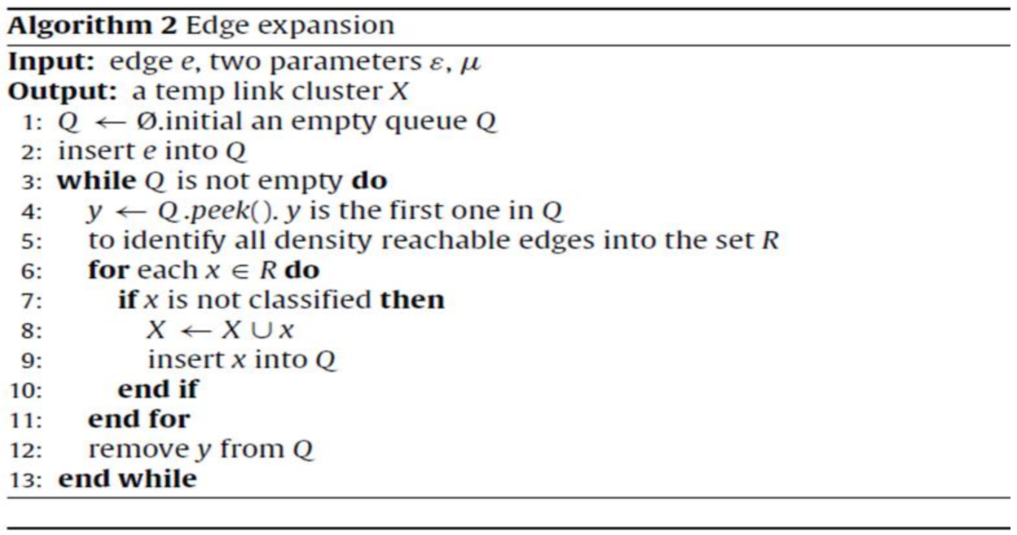
```
Algorithm 1 Framework of DBLC
Input: a complex network $G = (V, E)$, $\varepsilon$, $\mu$
Output: the set of overlapping communities $P = \{C_1, C_2, \ldots, C_m\}$
  Step 1. assign an index to each edge
  Step 2. get incidence matrix $W$ based on adjacency matrix $A$ of $G$
  Step 3. generating a similarity matrix between a pair of edges according to incidence matrix $W$
  Step 4. $LP \leftarrow \emptyset$
      For each edge $e_i$ in $E$
          If $e_i$ is not classified and $core(e_i)$
              expand a new cluster $X$ according to Algorithm 2
              $LP \leftarrow LP \cup X$
          End If
      End For
  Step 5. update the link partition $LP$ with unclassified edge according to Algorithm 3 to get final link partition $FLP$
  Step 6. convert link set in each partition $FLP$ to the node set $P$
```

Figure 1 Algorithm one of the DBLC

The problem that we may face

- Border link will be assigned to the cluster discovered first not the cluster that has most similarity to.
- The result may be different starting from different edge based on density reachable concept.

To detect clusters correctly, the edge expansion strategy for clustering based on core density reachable is shown in Algorithm 2, by creating a new cluster with a core edge and iteratively gathering core density reachable edges from the very first core edge. This will stop when no more core edges can be added to the cluster. Then it selects another unclassified core edge randomly for expanding. This process continues until some rough clusters containing only core edges are found.

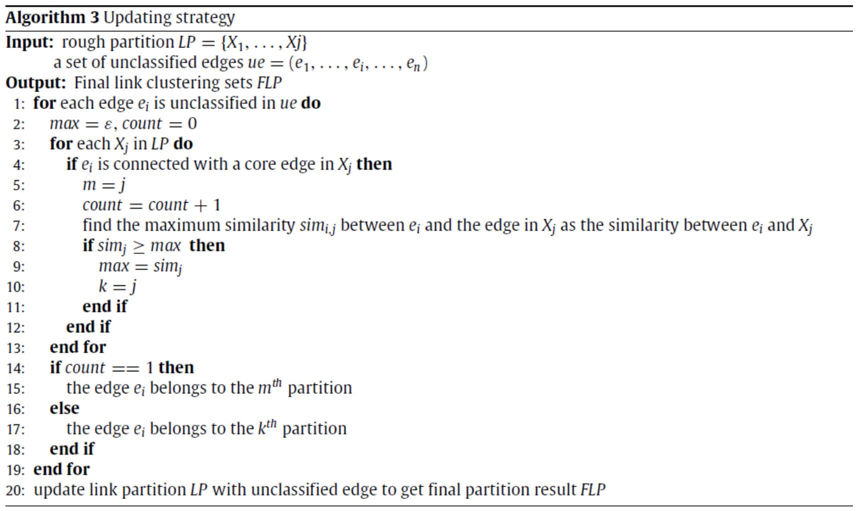


Algorithm 2 Edge expansion

```
Input: edge $e$, two parameters $\varepsilon$, $\mu$
Output: a temp link cluster $X$
Q ← Ø.initial an empty queue Q
insert e into Q
while Q is not empty do
    y ← Q.peek(). y is the first one in Q
    to identify all density reachable edges into the set R
    for each $x \in R$ do
        if x is not classified then
            $X \leftarrow X \cup x$
            insert x into Q
        end if
    end for
    remove y from Q
end while
```

Figure 2 Edge Expansion

After the expansion procedure, there will be some edges that has not been assigned to any cluster, but may be in neighborhood of some core edges that have been already assigned to a specified cluster. In this case, such border edge must be assigned to the closest cluster that contains one of its neighbor core edges with the most similarity to itself. Finally, there will be some edges that are not in neighborhood of any core edges and will be reported as noise edges. These noise edges will not be assigned to any cluster. All this procedure will be done according to Algorithm 3, depicted in figure 3.



```
Algorithm 3 Updating strategy
Input: rough partition $LP = \{X_1, \dots, Xj\}$
       a set of unclassified edges $ue = (e_1, \dots, e_i, \dots, e_n)$
Output: Final link clustering sets $FLP$
1: for each edge $e_i$ is unclassified in $ue$ do
2:   $max = \varepsilon$, $count = 0$
3:   for each $X_j$ in $LP$ do
4:     if $e_i$ is connected with a core edge in $X_j$ then
5:       $m = j$
6:       $count = count + 1$
7:       find the maximum similarity $sim_{i,j}$ between $e_i$ and the edge in $X_j$ as the similarity between $e_i$ and $X_j$
8:       if $sim_j \geq max$ then
9:         $max = sim_j$
10:        $k = j$
11:      end if
12:    end if
13:  end for
14:  if $count == 1$ then
15:    the edge $e_i$ belongs to the $m^{th}$ partition
16:  else
17:    the edge $e_i$ belongs to the $k^{th}$ partition
18:  end if
19: end for
20: update link partition $LP$ with unclassified edge to get final partition result $FLP$
```

Figure 3 Algorithm 3 updating strategy

3. Edge similarity

The similarity formula between edges $e_{v,v'}$ and $e_{w,w'}$ is shown from Eqs. (1)–(2). Here, the similarity between two edges that are common in one node, is calculated according to their uncommon nodes. For this reason, we introduce two types of similarities here. One is the known Jaccard similarity measure, that is calculated by dividing the cardinality of common neighbor set of two nodes by the cardinality of union of their neighbor set. The other type of similarity that is the contribution of this paper, is calculated by dividing the number of existing links between common neighbors divided by the number of possible links between them. The last similarity measure that has just been introduced is inspired by the intuition that if the common neighbors of two nodes are more connected to each other, then there will be more chance or possibility for them to stay connected, if we just consider the nodes as people in social networks.

The final similarity measure is a weighted sum of the two similarities. We have:

$$sim(e_{v,v'}, e_{w,w'}) = \gamma \times sim_1(e_{v,v'}, e_{w,w'}) + (1-\gamma) \times sim_2(e_{v,v'}, e_{w,w'})$$

Gamma (γ) is a number in the interval (0,1) that specifies the weight of each similarity measure.

..Eq(1)

$$sim_1(e_{v,v'}, e_{w,w'}) = \begin{cases} 0 & \text{if } v \neq w \text{ and } v' \neq w' \\ \dfrac{comm(v,w)}{n(v) \cup n(w)} & \text{if } v' = w' \\ \dfrac{comm(v',w')}{n(v') \cup n(w')} & \text{if } v = w \end{cases}$$

…Eq(2)

$$sim_2(e_{v,v'}, e_{w,w'}) = \begin{cases} 0 & \text{if } v \neq w \text{ and } v' \nvDash w' \\ \dfrac{2 \times |E(comm(v,w))|}{(|comm(v,w)|)(|comm(v,w)-1|)} & \text{if } v' = w' \\ \dfrac{2 \times |E(comm(v',w'))|}{(|comm(v',w')|)(|comm(v',w')-1|)} & \text{if } v = w \end{cases}$$

If two edges are not common in one node, then their similarity is zero.

- |comm(i, j)| signifies the number of common neighbours of vertices i and j.
- n(i) denotes neighbours of vertex I including itself.
- E(comm(i, j)) signifies the edges existing among common neighbours of vertices i and j.

- |E(comm(i, j))| signifies the cardinal number of E(comm(i, j)).

1.4- Parameter analysis

- ε is the similarity threshold of a neighbourhood for an edge
- μ is a threshold for the number of edges in this neighbourhood.

For certain μ:

When ε is too little some core edges may be considered as noise and a cluster may be divided into a number of clusters mistakenly.

When ε is too large some noise edge may be mistakenly considered as core edge and some disjointed communities may become one.

For a certain ε:

When μ is too little it will detect clusters with high density.

When μ is too large it will lead to producing a few core edges and the cluster with a few members will not be detected.

EQ is a measure to evaluate the quality of overlapping community partition

..Eq(3)

$$EQ = \sum_{l=1}^{k} EQ_l$$

$$EQ_l = \frac{1}{|M|} \times \sum_{i \in H_l, j \in H_l} \frac{1}{O_i O_j} \left[A_{ij} - \frac{n_i n_j}{2|M|} \right]$$

- EQ = 0 when all nodes fit the same community.
- A higher value of EQ specifies denser intracommunity connections of the community structure.
- M is the total number of edges in the network.
- K is the number of communities.
- H_l is the node set of the community.
- A_{ij} is the adjacency matrix. $A_{ij} = 1$ only if two nodes are adjacent.
- n(i) & n(j) represent the degree of node i and node j.
- O_i represents the number of communities that node i belong to.

4. Experimental Results

Here we just use the six synthetic datasets that were created by the specified parameters in the paper for testing the effectiveness of DBLC algorithm. We represent the final experimental results of our implementation and compare it with the results mentioned in the paper. And for the other methods that are mentioned in the paper for comparison, we have been just able to find the implementation code of only one of them, CPM that has provided its implementation code officially.

Here are the NMI results of our implementation:

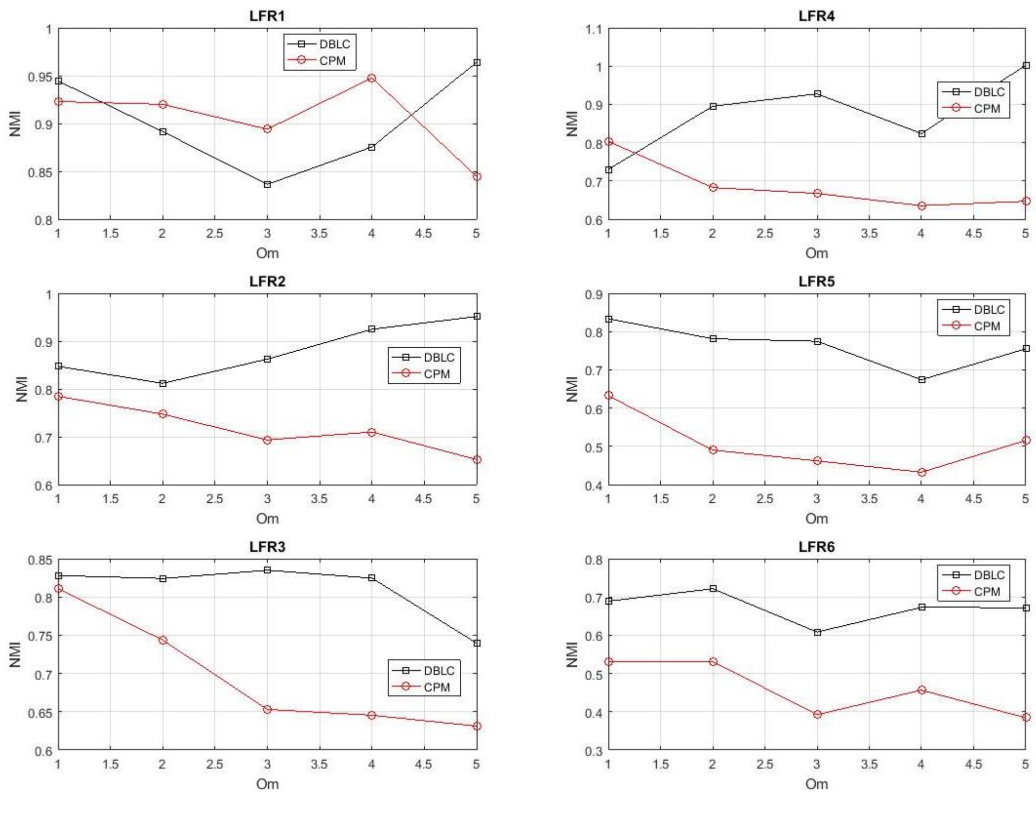


Figure 4 NMI results of our implementation

And here are the results mentioned in the paper:

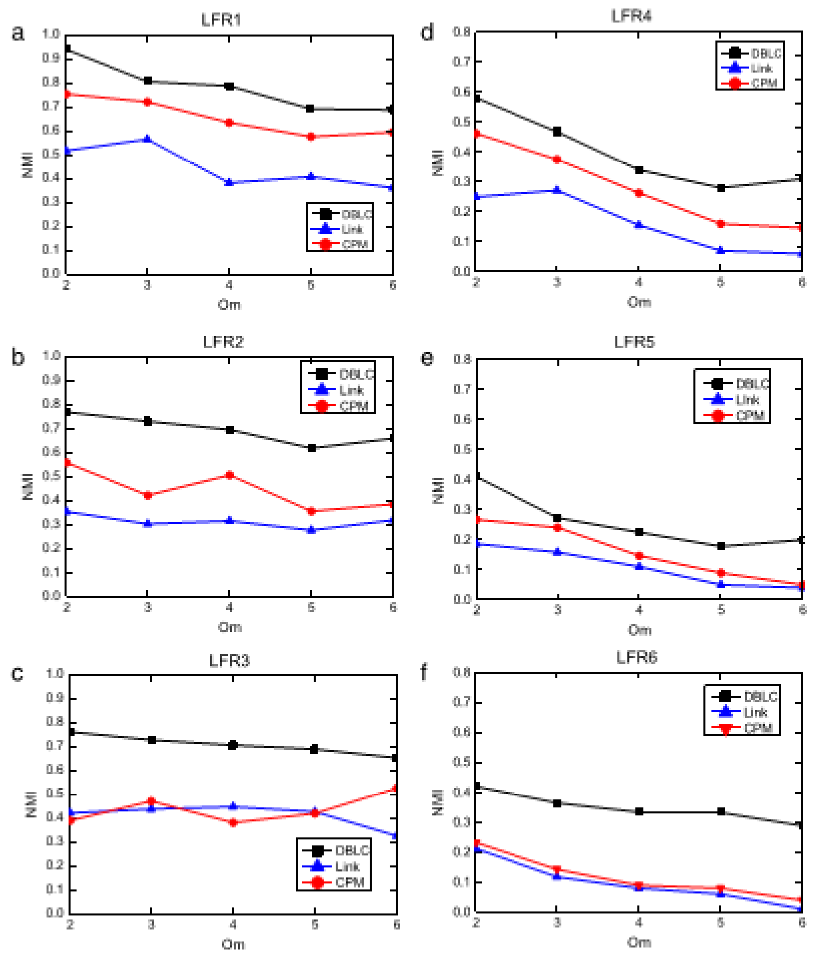


Figure 5 NMI results mentioned in the paper

As we observe, the experimental results of our implementation are close to the real ones mentioned in the paper, in the sense of variation, but not so close in values.

In the following, we represent the results of Precision, Recall and Fscore measures:

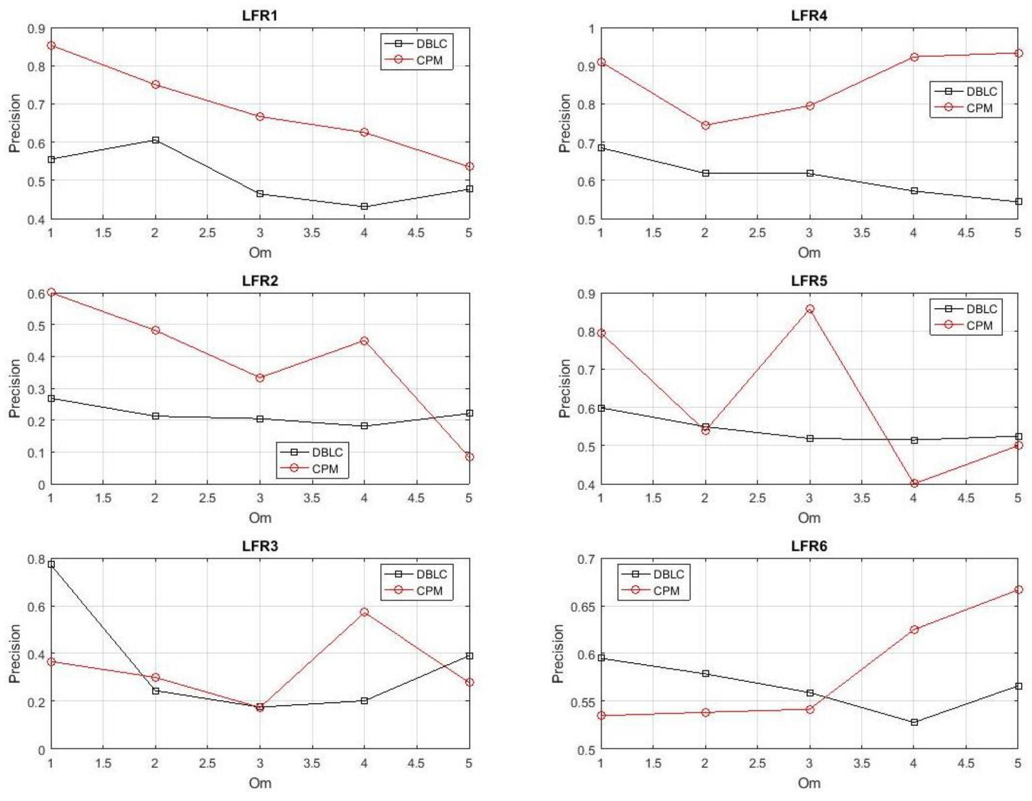


Figure 6 Precision results of our implementation

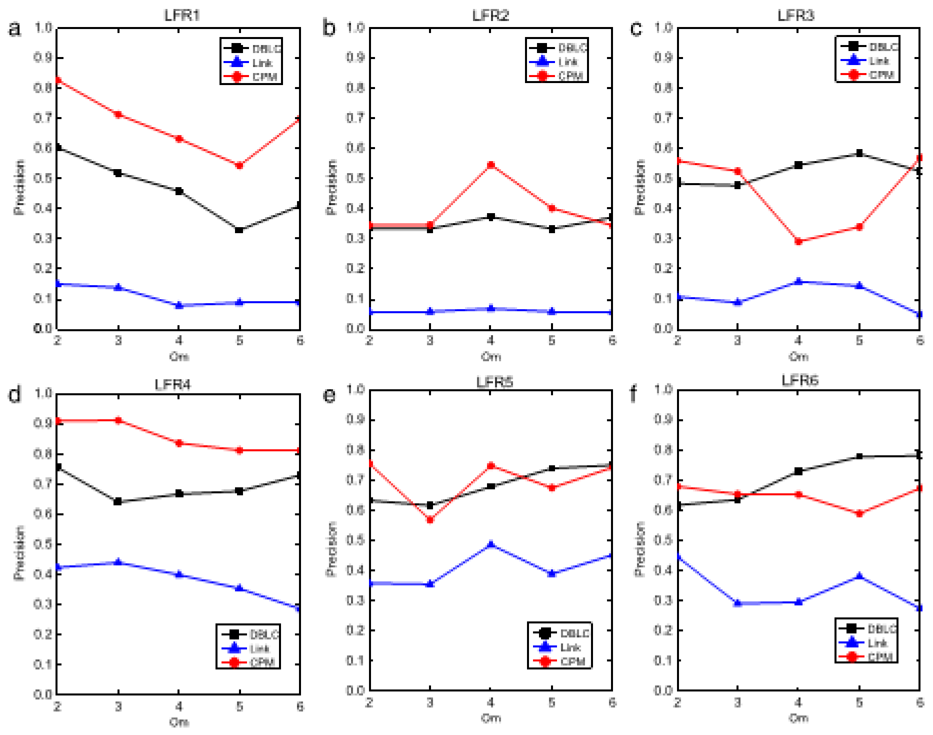


Figure 7 Precision results of the paper

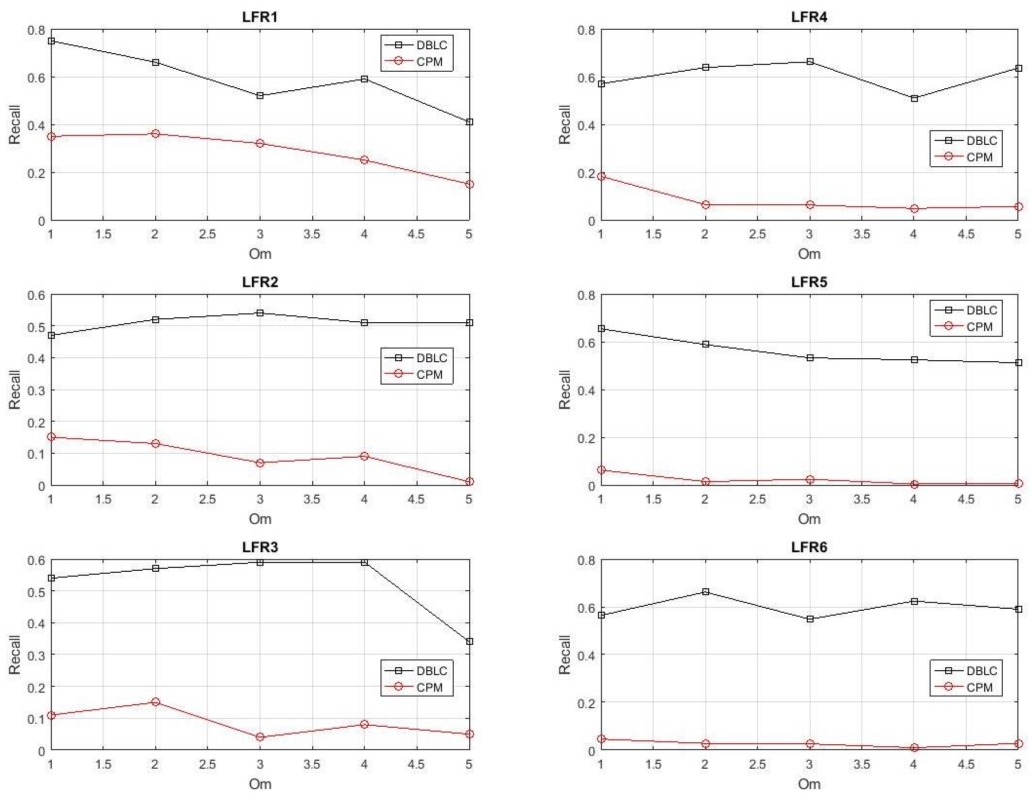


Figure 8 Recall results of our implementation

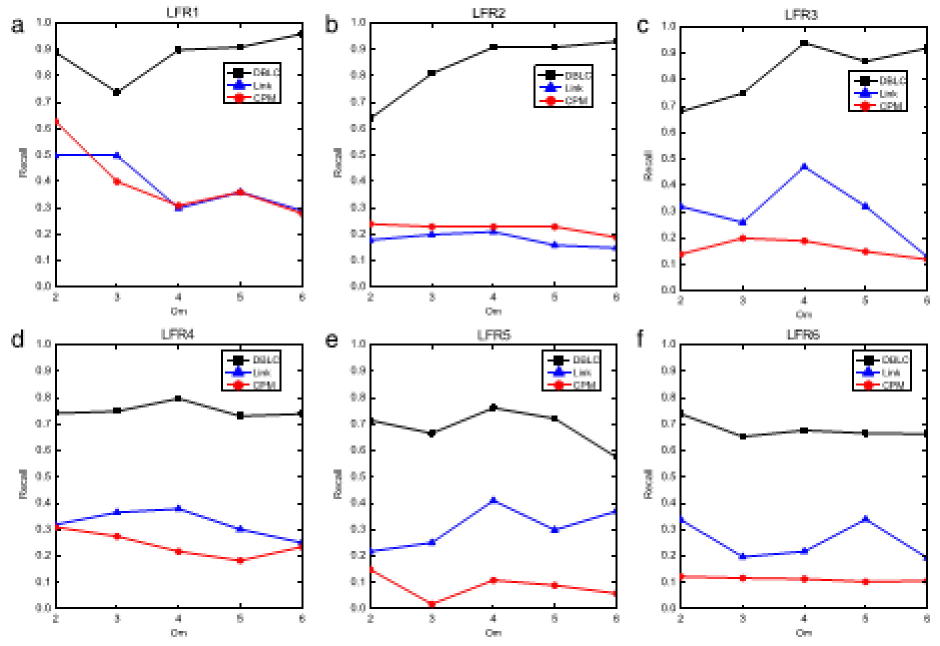


Figure 9 Recall results of the paper

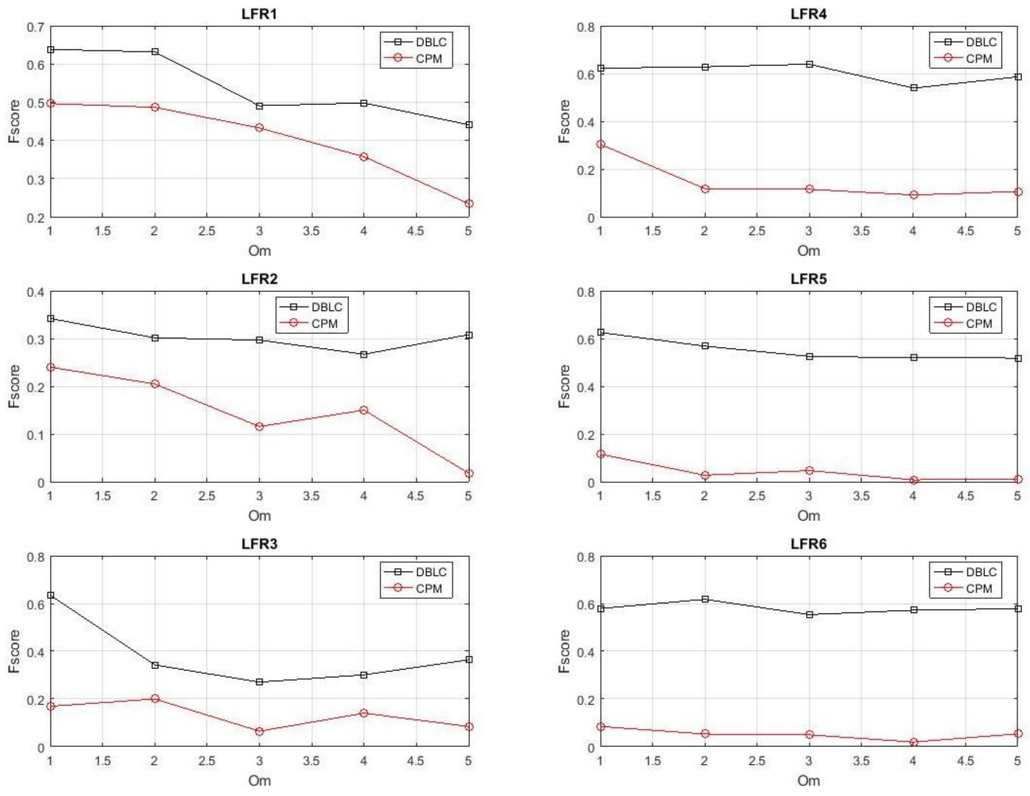


Figure 10 Fscore results of our implementation

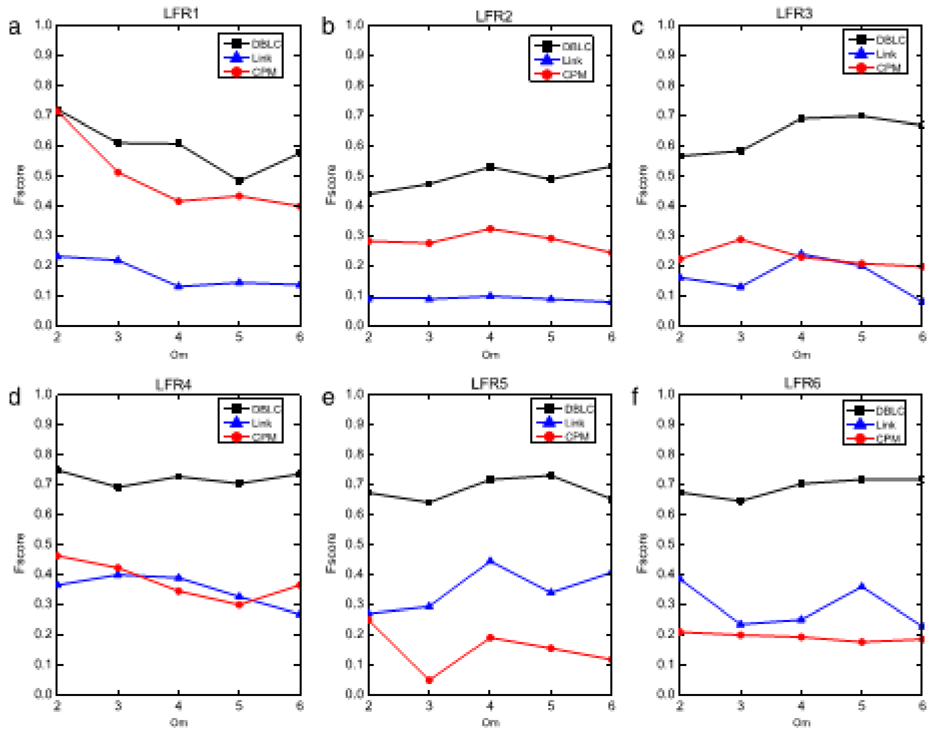


Figure 11 Fscore results of the paper

As we observe, the last results of our implementation and the ones mentioned in the paper are not so close in values, but at least for the Fscore measure that is a harmonic mean of precision and recall, we see that in our implementation results, all the Fscore values in all of six synthetic datasets are higher than the comparing method CPM with k value varying from 3 to 5.

References

[1] Mu, Zhu, Meng Fanrong, and Zhou Yong. "Density-based link clustering algorithm for overlapping community detection." Journal of Computer Research and Development 12 (2013): 006.