An MDL Approach to Efficiently Discover Communities in Bipartite Network*

Kaikuo Xu, Changjie Tang, Chuan Li, Yexi Jiang, and Rong Tang

School of Computer Science, Sichuan University, China
kaikuoxu@gmail.com, cjtang@scu.edu.cn

Abstract. Bipartite network is a branch of complex network. It is widely used in many applications such as social network analysis, collaborative filtering and information retrieval. Partitioning a bipartite network into smaller modules helps to get insight of the structure of the bipartite network. The main contributions of this paper include: (1) proposing an MDL 21 criterion for identifying a good partition of a bipartite network. (2) presenting a greedy algorithm based on combination theory, named as MDL-greedy, to approach the optimal partition of a bipartite network. The greedy algorithm automatically searches for the number of partitions, and requires no user intervention. (3) conducting experiments on synthetic datasets and the southern women dataset. The results show that our method generates higher quality results than the state-of-art methods Cross-Association and Information-theoretic co-clustering. Experiment results also show the good scalability of the proposed algorithm. The highest improvement could be up to about 14% for the precision, 40% for the ratio and 70% for the running time.

Keywords: Community Detection, Bipartite Network, Minimum Description Length, Information Theory.

1 Introduction

To understand the structure of a complex system, a common approach is to map the interconnected objects in the complex system to a complex network and study the structure of the complex network. During the mapping, the interacted objects are mapped into highly connected modules that are only weakly connected to one other. The modules are considered to embody the basic functions of the complex network. Thus people can identify the modules or communities of which the complex network is composed in order to comprehend the structure of the complex system [1-5]. One classical application is the recommendation system for E-Commence like Amazon. Let's look at an example.

Example 1. Fig 1 shows an example about books and customers. In (a), there are eleven customers and four books. To recommend books to a given customer '*C*', a

* This work was supported by NSFC Grant Number: 60773169 and 11-th Five Years Key Programs for Sci. &Tech. Development of China under grant No. 2006BAI05A01.

H. Kitagawa et al. (Eds.): DASFAA 2010, Part I, LNCS 5981, pp. 595–611, 2010.

recommendation system needs to search customers with similar tastes of '*C*' (or books having the same topic with the books bought by '*C*') from the sample data. To do so, the data is transformed into a two-mode network first, as shown in (b). The recommendation system could detect communities in the two-mode network directly. The result for community detection on (b) is shown in (c). Another choice is to project the two-mode network into one mode network and detect communities in the one-mode network. In Fig 1, (b) is projected to (d) and the result for community detection on (d) is shown in (e). Customers in the same group are considered to share the similar tastes and books in the same group are considered to share the same topic. In Fig 1, '*A*','*B*','*D*', and '*E*' are considered to share the similar tastes with '*C*' and '2' is considered to have the same topic with '1'. At last, the results are selectively applied to recommendation according to the accuracy of different methods.

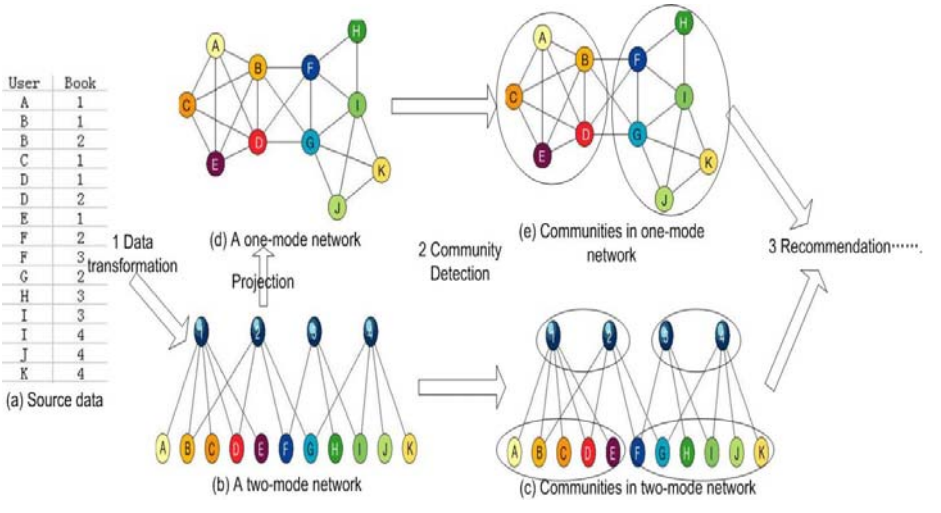

Fig. 1. The flowchart of recommendation for some sample data

A complex network is called one-mode network if it is mapped from a complex system composed of one type of objects. There are extended works on detecting communities in one-mode network. Among these works, a popular approach is to search for partitions that maximize the modularity [6, 7]. However, as a criterion of community quality, modularity is considered to have two disadvantages [8]: a resolution limit and a bias to equal-sized communities. Compression-based method that searches for partitions minimizing the description length is a recent alternation [9, 10]. Compression-based method usually makes use of the MDL principle in the field of information theory [11].

A complex network is called two-mode network or bipartite network if it is mapped from a complex system composed of two types of objects. In this paper, bipartite network refers to two-mode network. Fig. 1 (d) and (b) show examples of both one-mode network and two-mode network [12]. According to Fig.1, one-mode network has only one node set while two-mode network has two disjoint node sets. For

one-mode network, node connects to each other in the same set. We only need to analyze the connections to detect the communities in one-mode network. For two-mode network, nodes in the same set are not connected. To detect the communities of one node set, we need study its connections to the other node set and the community structure of the other node set simultaneously. Thus communities of two node sets can be detected simultaneously.

As discussed in the section of related works, the existing methods for bipartite network is still far from sound comparing with the methods for one-mode network. This paper proposes a compression-based method to resolve communities in bipartite network as: (a) defines an MDL criterion which is extended from [9]; and (b) proposes a greedy algorithm which is adapted from [13]. The criterion is based on binomial coefficient and the proposed greedy algorithm takes advantage of the properties of binomial coefficient. The proposed algorithm automatically discovers the communities of two node sets simultaneously. Experiment results show that the newly proposed method successfully finds communities that CA (cross-association) method [13] fails to find and it is more accurate than the well known ITCC (Information-theoretic co-clustering) method [14] while detecting communities.

The rest of the paper is organized as follows. Section 2 gives some related work. Section 3 introduces the basic idea of our method and the MDL criterion. Section 4 proposes a greedy algorithm that finds communities according to the proposed MDL criterion. Section 5 evaluates the proposed algorithms on four datasets. Section 6 concludes the paper.

2 Related Works

The earliest significant work on this topic is the biclustering/co-clustering [15] which simultaneously searches row/column communities. A row community and a column community together with the connections between them form a submatrix (bicluster). Cheng [16] proposed a biclustering algorithm for gene expression data analysis, using a greedy algorithm that identifies one bicluster at a time by minimizing the sum squared residue. Dhillon [14] proposed an information-theoretic co-clustering, which searches communities by minimizing the KL-divergence between the biclusters and the original matrix. According to the experiments, the results generated by biclustering/co-clustering are highly accurate. However, the row/column community number needs to be specified before running the algorithm.

Recently, Guimera [17] proposed a projection based method. It transforms the bipartite network to one-mode network and uses method for one-mode network to discover communities. However, the projection process is usually considered to cause information loss, which will leads to bad result. Barber [18] extended the modularity [6] to bipartite network and proposed a method searching communities by minimizing the modularity. Barber's method requires a serious constraint: the numbers of row community and column community need to be equal. Lehmann [19] proposed a method detecting biclique communities. However, its result highly depends on the user's specification on the relaxation of biclique community. All methods mentioned above require manual specification on some parameters.

At the same time, some parameter-free methods were proposed. Chakrabarti [13] proposed a CA(cross-association) method. It automatically discovers row/column communities simultaneously. Sun [20] extended CA to deal with time-evolving bipartite network and its ability to discover communities for static bipartite network is no better than CA. Papadimitriou[21] extended CA to search not only global communities but also local communities and its ability to discover global communities for static bipartite network is also no better than CA. Nevertheless, the result generated by CA is still unsatisfactory as shown in section 5.

3 An MDL Criterion

The MDL principle has been successfully used in many applications such as universal coding, linear regression and density estimation [11]. The basic idea of MDL principle is to use the regularity of given data to compress the data. The regularity that compresses the data the most is considered to describe the given data the best. In our method, the community structure is treated as the regularity of a bipartite network. Thus according to the MDL principle, to search a good community structure is actually to search the community structure that compresses the bipartite network the most. The sketch of the proposed method for bipartite network community detection is as follows.

1) Define a formula to compute the bits for expressing a bipartite network directly. Let l be the bit length to directly express the bipartite network;
2) Divide the process to express the bipartite network into two parts: one part to express its community structure and the other part to express the extra information describing the network given the community structure. Let l_c be the bit length. Note that if all vertices are put into one community, l_c is equal to l;
3) Search for the optimum partition that minimizes l_c. This method is called a compression-based method because l_c is expected to be less than l. l_c is called the MDL criterion since it is the optimizing target.

Table 1. Table of main symbols

Symbol	Definition
A	Binary data matrix
n, m	Dimensions of A (rows, columns)
k, e	Number of row and column communities
k^*, e^*	Optimal number of communities
(Q_r, Q_c)	A partition
$A_{i,j}$	Submatrix of A composed of rows in community i and columns in community j
a_i, b_j	Dimensions of $A_{i,j}$
$n(A_{i,j})$	Number of elements in $A_{i,j}$ $n(A_{i,j}) = a_i b_j$
$n_1(A_{i,j})$	Number of 1 in $A_{i,j}$
$C(A_{i,j})$	Code cost for $A_{i,j}$
$T(A; k, e, Q_r, Q_c)$	Total cost for A
$Ar_{i,j}$	Sub-matrix of A composed of row i and columns in community j
$n(Ar_{i,j})$	Number of elements in $Ar_{i,j}$ $n(Ar_{i,j}) = b_j$
$n_1(Ar_{i,j})$	Number of 1 in $Ar_{i,j}$

3.1 Terminologies

Let X be an unweighted and undirected bipartite network. Suppose X is composed of two disjoint node sets S_r and S_c, the size of S_r is n and the size of S_c is m, then X can be described as a $n \times m$ (n, $m \geqslant 1$) adjacency matrix A. Let us index the rows as 1, 2,…, n and columns as 1, 2,…, m. Let k denote the number of row communities and let e denote the number of column communities. Let us index the row communities by 1, 2, … , k and the column communities by 1, 2, … , e. Let

$$Q_r: \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, k\}$$

$$Q_c: \{1, 2, \dots, m\} \rightarrow \{1, 2, \dots, e\}$$

denotes the assignments of rows to row communities and columns to column communities, respectively. The pair (Q_r, Q_c) is referred as a partition. We denote submatrix of A composed of rows in community i and columns in community j as $A_{i,j}$, i = 1, 2, …, k and j = 1, 2, …, e. Let the dimensions of $A_{i,j}$ be (a_i, b_j).

Example 2. Fig 1 (c) shows a partition example for Fig1 (b). In this example, S_r = {1,2,3,4}, S_c = {A,B,C,D,E,F,G,H,I,J,K}; n = 4, m = 11; k = 2, e = 2; Q_r = {1,2,3,4}→{1,1,2,2} , Q_c ={A,B,C,D,E,F,G,H,I,J,K}→{1,1,1,1,1,2,2,2,2,2,2}.

3.2 A Two-Part Coding

We now describe a two-part coding for matrix A. The first part is the *partition complexity* that describes the partition (Q_r, Q_c). The second part is the *conditional complexity* that describes the matrix given the partition.

3.2.1 Partition Complexity

The partition complexity consists of the following terms:

1) Bits needed to describe the number of rows, columns and number of 1: n, m, and $n_1(A)$. Since this term does not vary with different partitions, it makes no sense to the result so that it can be ignored.
2) Bits needed to describe the number of communities: $\log(n)$ for k and $\log(m)$ for e. Since these two terms do not vary with different partitions, it is ignored as well.
3) Bits needed to describe the community to which the rows and columns belong: $n\log(k)$ and $m\log(e)$.
4) Bits needed to describe all the ke submatrix $A_{i,j}$, i.e. the number of 1 between pairs of row community and column community: $ke\log(n_1(A))$

3.2.2 Conditional Complexity

Example 3. Assume there is a matrix A with only 1 row community and 1 column community. Given $n(A)$ and $n_1(A)$, the left information of A is expressed as follows: (1) Each combination of the $n_1(A)$ 1's is mapped to an integer index . Thus it requires

$\binom{n(A)}{n_1(A)}$ integers to map all the combinations. (2) All the integers are encoded into bits. Consider the following matrix

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

It requires $\binom{n(A)}{n_1(A)} = \binom{16}{4} = 1820$ integers. The matrix could be encoded using log(1820) bits.

Given the partition complexity, the information to fully express A can be expressed using $\sum_{1\le i\le k}\sum_{1\le j\le e} C(A_{i,j})$ bits where

$$\sum_{1\le i\le k}\sum_{1\le j\le e} C(A_{i,j}) = \log\left[\prod_{i=1}^{k}\prod_{j=1}^{e}\binom{n(A_{i,j})}{n_1(A_{i,j})}\right] \tag{1}$$

Then the total bits to describe A with respect to a given partition is

$$\begin{aligned} T(A; k, e, Q_r, Q_c) =& n\log(k) + m\log(e) + ke\log(n_1(A)) \\ &+\log\left[\prod_{i=1}^{k}\prod_{j=1}^{e}\binom{n(A_{i,j})}{n_1(A_{i,j})}\right] \end{aligned} \tag{2}$$

3.3 Problem Formulation

By the MDL principle, smaller $T(A; k, e, Q_r, Q_c)$ leads to higher quality partition. Thus in order to resolve the communities of A, we need to find the optimal partition that minimizes $T(A; k, e, Q_r, Q_c)$. The optimal partition correspond to k^*, e^*, Q_r^*, Q_c^* and $T(A; k^*, e^*, Q_r^*, Q_c^*)$. Typically, this problem is a combination optimization problem, thus is a NP-hard problem [22]. A common approach to conquer such problem is the evolutionary algorithm [23, 24]. In order to obtain a deterministic result, we use an approximate algorithm instead.

4 A Split-Refine Greedy Algorithm

4.1 Sketch of the Algorithm

The proposed algorithm is a greedy algorithm, as shown in Algorithm 1. It is the algorithm for the case when n is bigger than or equal to m. And it has a counterpart for the case when n is smaller than m. It starts from $k = 1$ and $e = 1$. For each iteration, it performs a row and column splitting (line 6), a column splitting only (line 7), and a row splitting only (line 8). Each step above guarantees that the total cost is non-increasing, so it is considered to be a greedy algorithm. At last, it adjusts the search

step size for the three steps above. The procedure terminates if the failure continues $3*\sigma$ times, where σ is a threshold. Here σ is set to be 3.

Algorithm 1. MDL-greedy

```
Input: A
Output: k*, e*, Q_r*, Q_c*
1.   k =1; e = 1; Q_r = {1,1,…,1}; Q_c = {1,1,…,1}; T = 0; F=0, R.rowstep = 1, R.columnstep =1
                                   //start as one community;  F is the flag for no improvement;
                                   R is a counter for repeated splitting, i.e. the search step size
2.   currentcost = CostComputation (k, e, Q_r, Q_c)
3.   F = 0;
4.   if n >= m
5.     do
6.       Row-column-greedy(k, e, Q_r, Q_c, F, R);
7.       Column-greedy(k, e, Q_r, Q_c, F, R);
8.       Row-greedy(k, e, Q_r, Q_c, F, R);
9.       Adjust-searchstep(F, R);
10.    while (F < 3σ)
11.  end
12.  k* = k, e* = e, Q_r* = Q_r, Q_c*= Q_c
13.  return
Procedure CostComputation (k, e, Q_r, Q_c)
    //compute the cost of a partition according to formula 2
```

Algorithm 2 shows the row and column searching. It splits the row and column first. During the splitting, a new row community and a new column splitting are generated. After the splitting, it reassigns each row/column to a new community that has the least cost. At last, the algorithm checks whether the splitting leads to a better compression. If the compression has been improved, the new partition is assigned to the current partition. The check guarantees the non-increasing of the total cost. Algorithm 2 has its counterparts for **Column-greedy** and **Row-greedy.** There are two sub-procedures SplitRow() and ReassginRowCommunity(), which are described in Algorithm 4 and Algortihm 5, respectively. SplitColumn() and ReassginColumnCommunity() are fundamentally the same as the former two except that they are for column.

Algorithm 2. Row-column-greedy

```
Input: k, e, Q_r, Q_c, F, R
Output: k, e, Q_r, Q_c, F, R
1.   [tmpk, tmpQ_r] = SplitRow(k, Q_r, R); [tmpe, tmpQ_c] = SplitColumn(tmpe, tmpQ_c, R );
2.   [tmpk, tmpe, tmpQ_r, tmpQ_c] =  ReassginRowCommunity(tmpk, e, tmpQ_r, Q_c);
3.    [tmpk, tmpe, tmpQ_r, tmpQ_c] = ReassginColumnCommunity(tmpk, tmpe, tmpQ_r, tmpQ_c));
4.    newcost = CostComputation(tmpk, tmpe, tmpQ_r, tmpQ_c);
5.    if newcost >= currentcost
6.         F = F + 1;
7.    else
8.         currentcost = newcost;      k = tmpk; Q_r = tmpQ_r e = tmpe; Q_c = tmpQ_c; F = 0;
9.    end
```

Algorithm 1 adjusts the search size R according to the failure during the search. Algorithm 3 shows the detail. For each consecutive three times failure, the search step R will be increased. If there is no such failure, the search step size is kept as 1.

Algorithm 3. Adjust-searchstep

```
Input: F, R
Output: F, R
1.    if F%3 == 0
2.      if k > e  R.rowstep++;
3.      else  if  e > k
4.                Rcolumnstep++;
5.            else
6.                R.rowstep++;
7.                Rcolumnstep++;
8.            end
9.      end
10.   else
11.     R.rowstep=1;
12.     Rcolumnstep=1;
13.   end
```

4.2 Split the Submatrix

The algorithm splits the submatrix by splitting the row/column respectively. The detail of function SplitRow() is as follows: line 3 picks up the submatrix whose cost is the highest; line 11 judges whether the reduced cost after removing a row and regarding it as a community is higher than or equal to a threshold. In our work, the threshold is set as the maximum reduced cost. The splitting step searches for the right value for k and e.

Lemma 4.1. if $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$, then $C(A_1) + C(A_2) \leqslant C(A)$.

Proof

$$
\begin{aligned}
C(A) &= \log[\binom{n(A)}{n_1(A)}] = \log[\binom{n(A_1)+n(A_2)}{n_1(A_1)+n_1(A_2)}] \\
&= \log[\sum_{x=0}^{n_1(A)} \binom{n(A_1)}{n_1(A)-x}\binom{n(A_2)}{x}] \\
&\geq \log[\binom{n(A_1)}{n_1(A_1)}\binom{n(A_2)}{n_1(A_2)}] \\
&= \log[\binom{n(A_1)}{n_1(A_1)}] + \log[\binom{n(A_2)}{n_1(A_2)}] \\
&= C(A_1) + C(A_2)
\end{aligned}
$$

where the second equality follows the Vandermondes's Identity [25] and the inequality follows the monotonic of the log function.

Corollary 4.1. For any $k_2 \geqslant k_1$ and $e_2 \geqslant e_1$, there exists a partition such that

$$
\sum_{1\leq i\leq k_2}\sum_{1\leq j\leq e_2} C(A_{i,j}) \leq \sum_{1\leq i\leq k_1}\sum_{1\leq j\leq e_1} C(A_{i,j})
$$

Proof. This simply follows Lemma 4.1.
According to Corollary 4.1, the function SplitRow() will never increase the conditional complexity.

Algorithm 4. SplitRow

Input: k, Q_r, R
Output: k, Q_r

1. Repeat line 2~line14 *R.rowsize* times
2. $k = k + 1$;
3. $r = \arg\max_{1 \le i \le k} \sum_{1 \le j \le e} \binom{n(A_{i,j})}{n_1(A_{i,j})}$
4. for each row i ($1 \leqslant i \leqslant n$)
5. if $Q_r(i) == r$
6. $\text{costr}(i) = \sum_{1 \le j \le e} \left(\binom{n(A_{r,j})}{n_1(A_{r,j})} - \binom{n(A_{r,j}) - n(Ar_{i,j})}{n_1(A_{r,j}) - n_1(Ar_{i,j})} - \binom{n(Ar_{i,j})}{n_1(Ar_{i,j})} \right)$

   //record the reduced cost when removing row i from community r and regard it as a community
7. end
8. end
9. *threshold* = max(costr);
10. for each row i ($1 \leqslant i \leqslant n$)
11. if costr(i) >= threshold
12. $Q_r(i) == k$;
13. end
14. end
15. **return**

4.3 Reassign the Rows and Columns to Community

The reassign step is a refinement of the result from the split step. It not only refines the splitted row communities, but also refines the column communities so as to fit the change of row communities. Line 5 picks up the community to which assigning a row will add the least cost to the total. Line 7 judges whether to reassign a row will reduce the total cost. The symbols before reassignment are denoted as k^0 and $C(A_{i,j}{}^0)$, and the symbols after the reassignment are denoted as k^1 and $C(A_{i,j}{}^1)$.

Theorem 4.1. $\sum_{1 \le i \le k^0} \sum_{1 \le j \le e} C(A_{i,j}{}^0) \ge \sum_{1 \le i \le k^1} \sum_{1 \le j \le e} C(A_{i,j}{}^1)$

Proof. This simply follows the reassignment judgment.

By Theorem 4.1, the function ReassginRowCommunity() will never increase the conditional complexity.

4.4 Computational Complexity

Line 5 of Algorithm 5 computes ke binomial coefficient. The same computation is also required for the column reassignment. Thus the computational complexity of Algorithm 5 is O($I(n+m)ke$), where I is the count of the outer loop.

Line 2 of Algorithm 4 computes ke binomial coefficient, and the loop on line 3 computes ne binomial coefficient. Thus the computational complexity of Algorithm 4 is O($(k+n)e$). Here the loop time R is ignored since its maximum value is 3.

Considering the computation on columns, the complexity of each iteration in Algorithm 1 is $O(2I(n+m)ke + 2ke+ne+mk) = O(I(n+m)ke)$. Thus the complexity of Algorithm 1 is $O((k^*+e^*) I(n + m)k^*e^*)$. Since k^*, e^*, and I are small according to our experiments, Algorithm 1 is linear.

Algorithm 5. ReassignRowCommunity

Input: k, e, Q_r, Q_c
Output: k, e, Q_r, Q_c
1. do
2. currentcost = CostComputation (k, e, Q_r, Q_c);
3. $tmpQ_r = Q_r$;
4. for each row r ($1 \leqslant r \leqslant n$)
5. $i = \arg\min_{1 \le i \le k} \sum_{1 \le j \le e} \left(\binom{n(A_{i,j}) + n(Ar_{r,j})}{n_1(A_{i,j}) + n_1(Ar_{r,j})} - \binom{n(A_{i,j})}{n_1(A_{i,j})} \right)$;
6. $ipos = Q_r(r)$;
//the current community row r belong to
$$\sum_{1 \le j \le e} \left(\binom{n(A_{ipos,j})}{n_1(A_{ipos,j})} - \binom{n(A_{ipos,j}) - n(Ar_{r,j})}{n_1(A_{ipos,j}) - n_1(Ar_{r,j})} \right)$$
7. if
$$\geq \sum_{1 \le j \le e} \left(\binom{n(A_{ipos,j}) + n(Ar_{r,j})}{n_1(A_{ipos,j}) + n_1(Ar_{r,j})} - \binom{n(A_{ipos,j})}{n_1(A_{ipos,j})} \right)$$
8. $tmpQ_r(r) = i$
9. $Q_r = tmpQ_r$;
10. end
11. end
12. for each column c ($1 \leqslant c \leqslant m$)
13. do the same as done from each row
14. end
15. newcost = CostComputation (k, e, Q_r, Q_c)
16. while newcost < currentcost;
17. **return**

5 Performance Study

All experiments are conducted on an INTEL core 2DuoProcessorE2160 with 2G memory, running Windows XP. All algorithms are implemented in Matlab R2007b. The program is run on J2SE 5.0.

5.1 The Synthetic Datasets

To check the performance of the proposed algorithm, the algorithm is examined over four synthetic datasets, which parallel the datasets for one-mode network in [26].

Dataset1: The sizes of S_r and S_c for each graph are set as 192 and 192, respectively. S_r and S_c are divided into three communities of 64 vertices, denoted as $\{S_{r1}, S_{r2}, S_{r3}\}$ and $\{S_{c1}, S_{c2}, S_{c3}\}$ respectively. Edges are placed between vertex pairs of different types independently and randomly. The distribution of the edges for a graph is shown in Table 2. Here Z_{ij} is the average degree of the vertices in S_{ri}. Several constraints are

added on the generated graphs: For a given i, $\sum_{1\le j\le 3} Z_{ij} = 24$; $Z_{ii} > \sum_{j!=i} Z_{ij}$. All $\overset{i!=j}{Z_{ij}}$ are the same.

Table 2. Edge distribution for dataset 1

Graph	S_{c1}	S_{c2}	S_{c3}
S_{r1}	Z_{11}	Z_{12}	Z_{13}
S_{r2}	Z_{21}	Z_{22}	Z_{23}
S_{r3}	Z_{31}	Z_{32}	Z_{33}

Dataset 2: the number of graphs is the same as that of datasets 1. Each graph in dataset 1 has a corresponding subgraph in dataset 2, as shown in Table 3. All graphs in dataset 2 are generated by removing S_{r3} and its related edges from graphs in dataset 1.

Dataset 3: Let y be an integer and $1 \leqslant y \leqslant 21$. The sizes of S_r and S_c for each graph are set as 192y and192y, respectively. S_r and S_c are divided into three communities of 64y vertices, denoted as $\{S_{r1}, S_{r2}, S_{r3}\}$ and $\{S_{c1}, S_{c2}, S_{c3}\}$ respectively. Edges are placed between vertex pairs of different types independently and randomly. The edge distribution for a graph is shown in Table 2. Here Z_{ij} is the average degree of the vertices in S_{ri}. Two constraints are added on the generated graphs: For a given i, $\sum_{1\le j\le 3} Z_{ij} = 24y$, $\overset{i!=j}{Z_{ij}} = 2y$.

Table 3. Edge distribution for dataset 2

Graph	S_{c1}	S_{c2}	S_{c3}
S_{r1}	Z_{11}	Z_{12}	Z_{13}
S_{r2}	Z_{21}	Z_{22}	Z_{23}

Dataset 4: the number of graphs is the same as that of datasets 3. Each graph in dataset 3 has a corresponding subgraph in dataset 2, as shown in Table 3. All the graphs in dataset 4 are generated by removing S_{r3} and its related edges from graphs in dataset 3.

5.2 Evaluation and Results

There is no 'standard' criterion to measure the quality of the results. Therefore, the precision, i.e. the fraction of vertices classified correctly, is computed as in [26], and it is adapted to bipartite network. The vertices in S_r that has been correctly classified is denoted as P(S_r), the vertices in S_c that has been correctly classified is denoted as P(S_c). Then precision can be computed as follows:

$$P(V)=\frac{|S_r|}{|S_r|+|S_c|}*\frac{|P(S_r)|}{|S_r|}+\frac{|S_c|}{|S_r|+|S_c|}*\frac{|P(S_c)|}{|S_c|}$$

Another criterion is the ratio between the number of discovered communities and the correct community number. The number of correct row communities is denoted as N_r and the number of correct column communities is denoted as N_c. Then ratio can be computed as follows:

$$Ratio = \frac{N_r}{N_r + N_c} * \frac{k}{N_r} + \frac{N_c}{N_r + N_c} * \frac{l}{N_c}$$

When *Ratio* is equal to 1, the method discovers the correct community number. When *Ratio* is less than 1, the method discovers less than the correct community number. And when *Ratio* is bigger than 1, the method discovers more than the correct community number.

Table 4. Precision&Ratio for dataset 1

	$i \neq j$ Z_{ij}	MDL greedy	CA	ITCC
$P(V)$	1	0.9765±0.0525	0.9100±0.0665	0.9744±0.0529
	2	0.9944±0.0277	0.9394±0.0569	0.9746±0.0566
	3	0.9897±0.0373	0.9414±0.0583	0.9776±0.0595
	4	0.9787±0.0507	0.9262±0.0642	0.9719±0.0663
	5	0.9470±0.0577	0.8794±0.0797	0.8553±0.1233
Ratio	1	1.0567±0.1258	1.1900±0.1276	-
	2	1.0133±0.0656	1.1600±0.1296	-
	3	1.0233±0.0855	1.1450±0.1415	-
	4	1.0433±0.1127	1.1650±0.1411	-
	5	1.0733±0.1577	1.2667±0.2247	-

Table 5. Precision&Ratio for dataset 2

	$i \neq j$ Z_{ij}	MDL greedy	CA	ITCC
$P(V)$	1	0.9958±0.0181	0.8745±0.0585	0.9329±0.0166
	2	0.9978±0.0130	0.8804±0.0631	0.9350±0.0523
	3	0.9939±0.0188	0.8896±0.0693	0.9388±0.0395
	4	0.9853±0.0207	0.8922±0.0671	0.9048±0.0789
	5	0.9430±0.0431	0.8339±0.0753	0.8097±0.1009
Ratio	1	1.0100±0.0438	1.3420±0.1372	-
	2	1.0040±0.0281	1.3300±0.1738	-
	3	1.0080±0.0394	1.2780±0.1703	-
	4	1.0100±0.0438	1.3060±0.1874	-
	5	1.0540±0.1749	1.4580±0.2417	-

The algorithms MDL-greedy, CA and ITCC are tested on dataset 1 and dataset 2. Although the 'farthest' initialization is reported to be the best in [14], 'random' initialization gains better result than 'farthest' initialization for ITCC in this experiment.

Therefore, 'random' initialization is adopted for ITCC. Furthermore, the community number is specified manually as the correct number since ITCC cannot search it automatically. Therefore, there is no *Ratio* for ITCC. The results are the average precision/ratio and their standard deviation over 100 random network generations, which are shown in Table 4 and Table 5. On both datasets MDL-greedy outperforms CA and ITCC. MDL-greedy gains a higher precision and is more stable than the other two methods. MDL-greedy scarcely discovers wrong community number while CA tends to generate more communities. When $Z_{ij}^{i!=j}$ increase to 5, the performance of all methods drops. The precision for ITCC drops the most, which is even worse than CA. The precision drop of MDL-greedy is acceptable because the variation is small. Comparing with CA, the *Ratio* for MDL-greedy is hardly affected. Obviously, CA gets trapped in the noise produced by the big $Z_{ij}^{i!=j}$ when searching for the number of communities. A sample result on a specified network is shown in Fig 2 to explain this phenomenon.

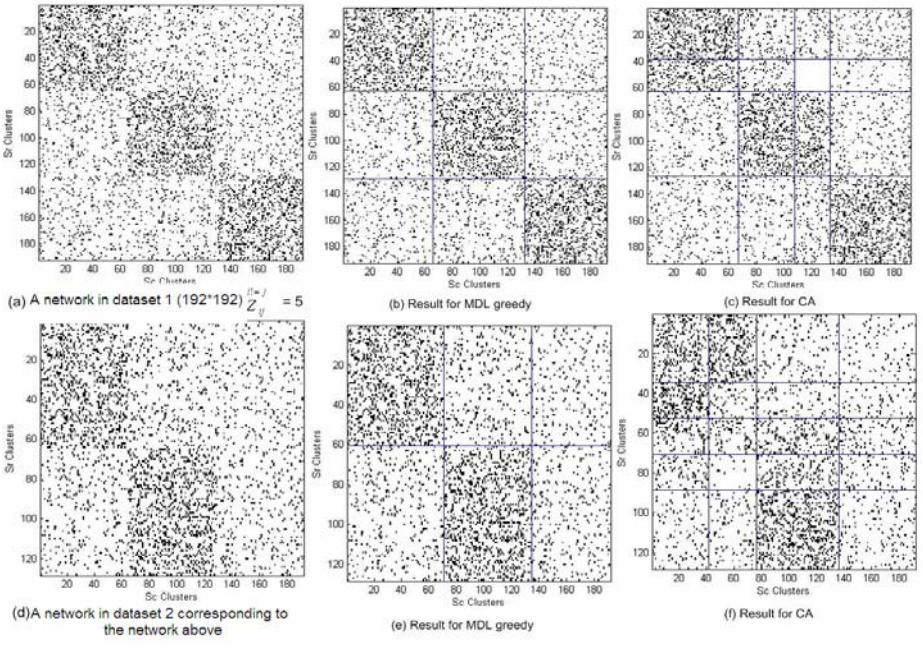

Fig. 2. A sample result for MDL-greedy & CA on dataset 1&2

5.3 Scalability

The algorithms MDL-greedy and CA are tested on dataset 3 and dataset 4. ITCC is not tested here since it requires manual specification on community number. The time cost for finding the optimum partition is recorded in our testing. The results are shown in Fig 3. The results show that the cost of MDL-greedy increases linearly along the increasing of y and is very stable. MDL-greedy outperforms CA on both datasets.

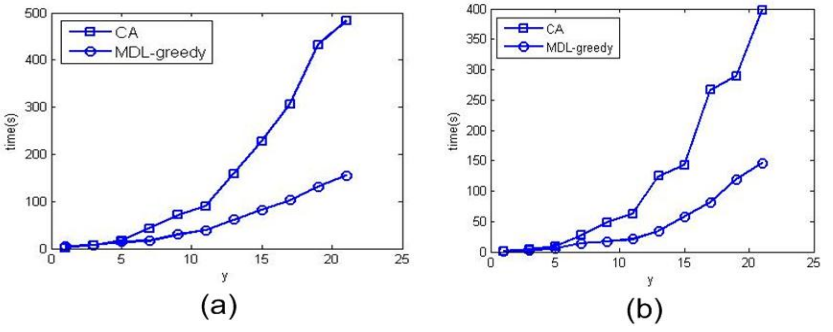

Fig. 3. Time cost for dataset 3(a)&dataset 4(b)

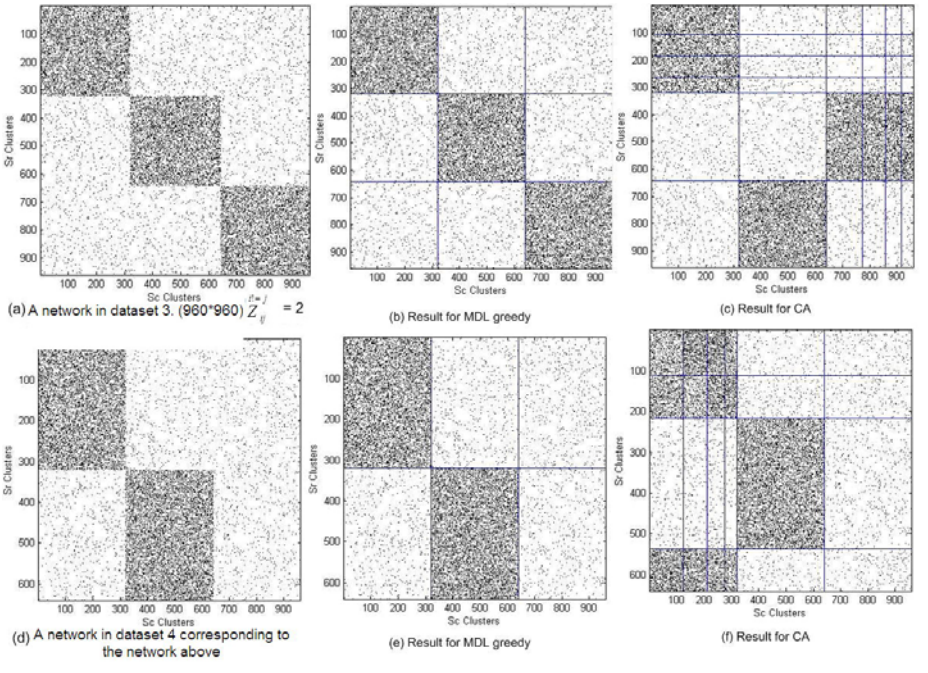

Fig. 4. A sample result for MDL-greedy & CA on dataset 3&4

When y is small, the performances of MDL-greedy and CA are close. But along the increasing of *y*, the performance difference between MDL-greedy and CA becomes larger and larger. According to the *Ratio* value in Table 4 and Table 5, the difference is attributed to the extra burden from CA's tendency to discover the community number more than the correct one. This phenomenon is very obvious in the result for dataset 4. Since the networks in dataset 4 are asymmetric, CA sometimes consumes much more time than expected. There are two jumps in Fig 3(b): from *y*=15 to *y*= 17 and from *y*=19 to *y*=21. A sample result for both methods on a specified network is shown in Fig 4. The number of communities that CA generates is much larger than the actual

number: (k_{CA}*=6, l_{CA}*=6) vs (k*=3, l*=3) in Fig 4(c) and (k_{CA}*=4, l_{CA}*=6) vs (k*=2, l*=3) in Fig 4(f). Generating more communities will decrease both the precision and scalability of CA. This may make MDL-greedy more practical than CA, since practical problems often provide large datasets.

5.4 The Southern Women Data Set

The southern women dataset is commonly used to evaluate the performance of bipartite network community detection methods [27]. It records eighteen women's attendance on fourteen events. Freeman has described it as "…a touchstone for comparing analytic methods in social network analysis[28]". MDL-greedy, CA and ITCC are run on this dataset. Again 'random' initialization is adopted for ITCC. ITCC is run twice and the community number are set as (k = 2,l = 3) and (k = 2, l = 4) respectively. The results are shown in Fig 5, where the women are labeled as 'W*' and the events are labeled as 'E*'. According to Freeman[28], the 'perfect' community for the southern women are 'W1-W9' and 'W10-W18'. There is no focus on the events. It is shown from Fig 5 that both MDL-greedy and CA detect the correct community number for the southern women. The result of MDL-greedy is exactly the 'perfect' community while the result of CA is quite different from the 'perfect' one. And no method in [28] generates the same result as CA. No method in [28] generates the same result as ITCC (k = 2, l = 3) either, which is also far from the 'perfect' one. However, the result of ITCC (k = 2, l = 4) is very close to the 'perfect' one in which only 'w8' is misplaced.

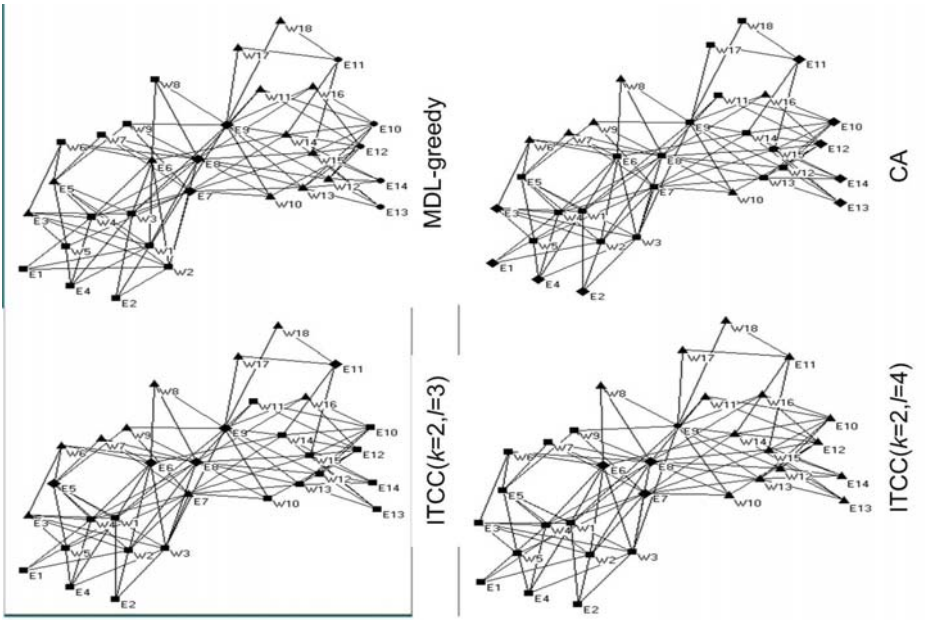

Fig. 5. Result for southern women dataset

6 Conclusions

To resolve communities in bipartite networks, the compression-based method is proposed for bipartite networks. A binomial coefficient based formula is applied as the description length of a partition. A greedy algorithm to minimizing the description length is also proposed under a split-refine framework. It successfully searches communities in automatic style for different types of nodes simultaneously. The experiment results show the high accuracy of the proposed method in the perspectives of both manually defined measures: precision and ratio. The experiment results also show its almost linear scale with the node size of the bipartite network, which fits the computational complexity analysis well.

References

[1] Newman, M.E.J.: The Structure and Function of Complex Networks. SIAM Review 45, 167–256 (2003)
[2] Guimerà, R., Amaral, L.: Functional cartography of complex metabolic networks. Nature 433, 895–900 (2005)
[3] Danon, L., Duch, J., Diaz-Guilera, A., Arenas, A.: Comparing community structure identification. J. Stat. Mech. P09008 (2005)
[4] Newman, M.E.J., Girvan, M.: Finding and Evaluating Community Structure in Networks. Physical Review E 69, 026113 (2004)
[5] Linden, G., Smith, B., York, J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing 7, 76–80 (2003)
[6] Newman, M.E.J.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences 103, 8577–8582 (2006)
[7] Pujol, J., Béjar, J., Delgado, J.: Clustering algorithm for determining community structure in large networks. Physical Review E 74, 9 (2006)
[8] Fortunato, S., Barthelemy, M.: Resolution Limit in Community Detection. Proceedings of the National Academy of Sciences 104, 36–41 (2007)
[9] Rosvall, M., Bergstrom, C.T.: An information-theoretic framework for resolving community structure in complex networks. Proc. Natl. Acad. Sci. USA 104, 7327–7331 (2007)
[10] Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. Proc. Natl. Acad. Sci. USA 105, 1118–1123 (2008)
[11] Barron, A., Rissanen, J., Yu, B.: The minimum description principle in coding and modeling. IEEE Transactions on Information Theory 44, 2743–2760 (1998)
[12] Strogatz, S.H.: Exploring complex networks. Nature 410, 268–276 (2001)
[13] Chakrabarti, D., Papadimitriou, S., Modha, D.S., Faloutsos, C.: Fully automatic cross-associations. In: Proc. Tenth ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., pp. 79–88 (2004)
[14] Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: KDD (2003)
[15] Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: a survey. IEEE/ACM TCBB 1, 24–45 (2004)
[16] Cheng, Y., Church, G.M.: Biclustering of expression data. In: ISMB (2000)
[17] Guimerà, R., Sales-Pardo, M., Lan, A.: Module identification in bipartite and directed networks. Physical Review E 76 (2007)

[18] Barber, M.J.: Modularity and community detection in bipartite network. Physical Review E 76 (2007)
[19] Lehmann, S., Schwartz, M., Hansen, L.K.: Biclique communities. Phys. Rev. E 78, 016108 (2008)
[20] Sun, J., Faloutsos, C., Papadimitriou, S., Yu, P.: GraphScope: Parameter-free mining of large time-evolving graphs. In: KDD (2007)
[21] Papadimitriou, S., Sun, J., Faloutsos, C., Yu, P.: Hierarchical, parameter-free community discovery. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 170–187. Springer, Heidelberg (2008)
[22] Sipser, M.: Introduction to the Theory of Computation. PWS Publishing Company (1997)
[23] Ashlock, D.: Evolutionary computation for modeling and optimization. Springer, New York (2005)
[24] Xu, K.K., Liu, Y.T., Tang, R.e.a.: A novel method for real parameter optimization based on Gene Expression Programming. Applies Soft Computing 9, 725–737 (2009)
[25] Rosen, K.H.: Discrete mathematics and its applications, 4th edn. WCB/McGraw-Hill, Boston (1999)
[26] Girvan, M., Neuman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA 99, 7821–7826 (2002)
[27] Davis, A., Gardner, B.B., Gardner, M.R.: Deep South. University of Chicago Press, Chicago (1941)
[28] Freeman, L.: Dynamic Social Network Modeling and Analysis. The National Academies Press, Washington (2003)