The Journal of Supercomputing
https://doi.org/10.1007/s11227-020-03151-y

Block diagonal dominance-based dynamic programming for detecting community

Xingquan Li[1] · **Cong Cao**[2] · **Tao Zhang**[3]

Abstract

Clustering or partition is a fundamental work for graph or network. Detecting communities is a typical clustering, which divides a network into several parts according to the modularity. Community detection is a critical challenge for designing scalable, adaptive and survivable trust management protocol for a community of interest-based social IoT system. Most of the existed methods on community detection suffer from a common issue that the number of communities should be prior decided. This urges us to estimate the number of communities from the data by some way. This paper concurrently considers eliminating the number of communities and detecting communities based on block diagonal dominace adjacency matrix. To construct a block diagonal dominance adjacency matrix for the input network, it first reorders the node number by the breadth-first search algorithm. For the block diagonal dominance adjacency matrix, this paper shows that the numbers of nodes in a community should be continuous adjacent. And thus, it only needs insert some breakpoints in node number sequence to decide the number of communities and the nodes in every community. In addition, a dynamic programming algorithm is designed to achieve an optimal community detection result. Experimental results on a number of real-world networks show the effectiveness of the dynamic programming approach on the community detection problem.

Keywords Social IoT system · Community of interest network · Block diagonal dominance matrix · Dynamic programming · Deep graph neural network

✉ Xingquan Li
xqli@mnnu.edu.cn

[1] School of Mathematics and Statistics, Minnan Normal University, Zhangzhou, China

[2] Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou, China

[3] Laboratory of Urban Land Resources Monitoring and Simulation, Ministry of Land and Resources, Shenzhen, China

Published online: 16 January 2020

1 Introduction

A future physical world is connected into cyberspace by the Internet of Things (IoT) system [1]. The main function of IoT technology is collecting and sharing information, and which is achieved via a trust management protocol for managing trust between IoT entities. A trust management protocol for IoT must be resilient to trust-related attacks to survive. However, existed trust management model does not pay attention to the social relationship among entities, which is important in social IoT systems. Designing and evaluating a scalable, adaptive and survivable trust management protocol for a community of interest (CoI)-based social IoT system is seriously important [2, 33]. For a CoI-based social IoT environment, nodes form into communities of interest (Fig. 1) and each node has a unique address to identify. Two nodes belonging to the same CoI have specific social interests and strong social ties, which could be manifested by more frequent interactions. It assumes that nodes in the same CoI can achieve an agreement on trust since they share the same interests. The goal of our trust management is to make sure each node trust evaluation converges to its community agreement. For this purpose, a critical challenge of detecting community for each node should be addressed, and an effective community detection algorithm is urgently needed. This paper considers the community detection problem and introduces an effective clustering method.

For graphs or networks, clustering is a crucial task where nodes are decomposed as many parts such that the number of cut edges between groups is minimal. Detecting communities is important for analyzing the system inside. Most of the existed community detection methods have a common challenge that the number of communities should be prior decided. This urges us to estimate it from the data by some way. Recently, some works have concerned the estimation of community number by using Bayesian inference applied to fits of network models to observed network data [21, 32, 35, 36]. These approaches define a generated random graph model of a network with a community structure and then fit it to the data to obtain a Bayesian posterior probability distribution and associated number of groups for possible partitioning of the network. Then, one averages over this distribution in some way to produce an estimate of the relative probability of different number of groups for the network. Complex network can be used to express the relationship between objectives

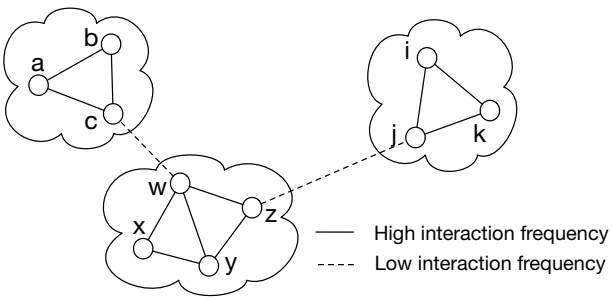

Fig. 1 Communities in social IoT

in social or information system [12, 36]. Since large networks may include billions of nodes, to deep mining the information from networks, it needs some more efficient algorithms to detect communities [22]. The methods based on decomposition are promising, where nodes are grouped into some highly inter-connected clusters [10, 11]. Detecting communities is very importance for network, which can find the function of every node and the property of structure. In addition, detected communities can be used to re-constructed the original network. Moreover, it can judge which community a new node belongs to. Community detection problem can be defined as follows: Given a network, it needs to find some communities such that the nodes in communities are dense connected and the nodes between communities are sparse connected [21, 32]. From the view of combinatorial optimization, community detection can be reduced as a minimum-k-cut problem, and the value of k also should be decided from network. As we all know, the minimum-k-cut problem is NP-hard problem [3, 13]. It is impossible to solve the problem for the large-scale network with billions of nodes and edges. Thus, the corresponding algorithms for this problem should trade-off the runtime and the solution quality [13]. Some effective approaches have been proposed to solve the problem for the large networks. Handcock et al. proposed a approach to detect several types of community by detecting inter-community edges and delete them from the network [9, 15]; Latouche et al. achieved community detection by merging similar nodes/communities recursively [5, 6, 19, 20]; McDaid et al. maximized the maximization optimization approach [23, 25]. Above these approaches are based on modularity, whose value is between -1 and 1 [34]. The common issue of these methods is that the number k of communities should be pre-decided [26, 28]. Thus if the number of communities and communities can be concurrently optimized, it has a lager probability to obtain a better result.

2 Related works

With the development, deep neural network (DNN)-based learning approaches are widely utilized to node classification and community detection [37]. In recent years, graph neural networks (GNN) have received increasing attention since their superiority on dealing with Hilbert space data. Since the graph-based data belong to Hilbert space, GNN-based methods are exactly effective to solve the learning tasks for graph [30]. Some works based on graph neural networks are proposed to handle the community detection problem [4, 7]. In [4], the authors first present the multi-scale graph neural network, whose updated process bases on stochastic block model (SBM) [8].

For a classical convolution neural network, t_{ij}^{l} is set as the feature at layer l with pixel (i, j), and which is calculated by a nonlinear transformation to $t_{i'j'}^{l}$, where (i', j') are the neighborhood of (i, j) [16]. The formal update form of 3×3 grids is shown as:

$$t_{ij}^{l+1} = m_{\mathrm{CNN}}(t_{i'j'}^{l}),$$

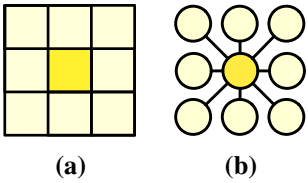

Fig. 2 **a** Convolution based on 3×3 grids; **b** Graph form of 3×3 grids

where $|i - i'| \leq 1, |j - j'| \leq 1$ is the box neighborhood of (i, j). A 3×3 box neighborhood is shown as Fig. 2a.

The 3×3 box neighborhood in 2D Euclidiean space can be described as a graph, as shown in Fig. 2b. Similarly, the feature of a vector t_i at vertex i in a graph can be calculated by following convolution neural network:

$$t_i^{l+1} = m_{\text{G-CNN}}(t_i^l, t_j^l),$$

where vertex j is in the k-edge neighborhood of i in graph. Given a graph as shown in Fig. 3a, the 1-edge neighborhood of vertex 4 is $\{1, 3, 5\}$ in Fig. 3b and the 2-edge neighborhood of vertex 4 is $\{1, 2, 3, 5, 6, 7\}$ in Fig. 3c. From the updating rule of graph neural network, it can be seen that the feature of vertex i in layer $l + 1$ is updated according to the feature of vertices j (in the neighborhood of i) of layer $l + 1$ and the feature of vertex i and vertices j (in the neighborhood of i) of layer l. Figure 4 shows an example of graph neural network.

There are many nonlinear maps for the graph convolution neural network. Reference [30] considers following nonlinear map.

$$t_i^{l+1} = m_{\text{G-CNN}}(t_i^l, t_j^l) = \text{ReLU}\left(W'^l t_i^l + W^l \sum_{j \in N(i)} t_j^l\right),$$

where ReLU is an activate function, and W'^l, W^l are the corresponding weights. To control multiple layer structures and utilize history features of vertices, graph-based long short-term memory (LSTM) method has been proposed [31].

$$t_i^{l+1} = m_{\text{G-LSTM}}(g_i^l, t_j^l) = y_{\text{G-LSTM}}\left(g_i^l, t_i^l, \sum_{j \in N(i)} t_j^l\right), d_j^l\Big)\Big),$$

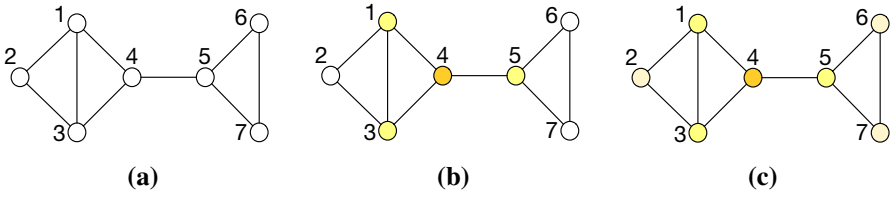

Fig. 3 **a** A graph with seven nodes; **b** 1-edge neighborhood of vertex 4; **c** 2-edge neighborhood of vertex 4

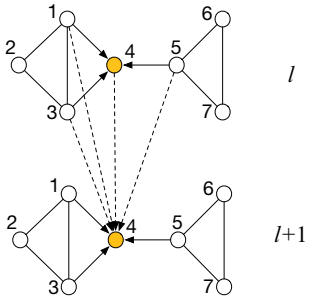

Fig. 4 Map relationship of graph convolution neural network between adjacent layer

and the detailed iterations of layer l are listed as follows. Let $\tilde{t}_i^{l,k} = \sum_{j \in N(v)} t_j^{l,k}$ for $k = 0, 1, \ldots, T$.

$$
\begin{aligned}
a_i^{l,k+1} &= \sigma(W_a'^l g_i^l + W_a^l \tilde{t}_i^{l,k}) \\
b_i^{l,k+1} &= \tanh(W_b'^l g_i^l + W_b^l \tilde{t}_i^{l,k}) \\
m_{ij}^{l,k+1} &= \sigma(W_m'^l g_i^l + W_m^l t_i^{l,k}), \quad j \in N(v) \\
d_i^{l,k+1} &= a_i^{l,k+1} \odot b_i^{l,k+1} + \sum_{j \in N(v)} \{m_{ij}^{l,k+1} \odot d_j^{l,k+1}\} \\
c_i^{l,k+1} &= \sigma(W_c'^l g_i^l + W_c^l \tilde{t}_i^{l,k}) \\
t_i^{l,k+1} &= c_i^{l,k+1} \odot \tanh(d_i^{l,k+1})
\end{aligned}
$$

From above iteration form, it can obtain $t_i^{l,T}$ at layer l, and let $t_i^{l+1} = t_i^{l,T}$ and $g_i^{l+1} = t_i^{l,T}$ for layer $l + 1$.

To achieve a better community detection result on real dataset, Chen et al. proposed a supervised learning with the help of line graph neural network instead of vertex graph [6]. The line graph $L(G) = (V^L, E^L)$ denotes the adjacency between the edges in initial graph $G = (V, E)$. For line graph $L(G)$, a vertex $ij^L \in V^L$ denotes an edge $e_{ij} \in E$, and if both two edges e_{ij} and $e_{i'j}$ in E connect to a vertex j in V, then there is an edge $e_{ij-i'j}^L \in E^L$ between two vertices ij^L and $i'j^L$ in V^L. An example of line graph is shown in Fig. 5, where seven circles and black lines are vertices and edges in V of initial graph, and nine squares and red lines are vertices and edges in V^L of line graph.

Krzakala et al. first proposed the non-backtracking operator in the context of community detection on the line graph [18], which is expressed by a matrix

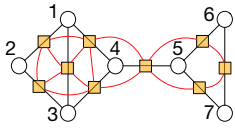

Fig. 5 Line graph

$$B_{(i\to j),(i'\to j')} = \begin{cases} 1, & \text{if } j = i' \text{ and } j' \neq i; \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The message-passing rules of back propagation (BP) can be expressed as a diffusion in the line graph $L(G)$ using this non-backtracking operator, with specific choices of activation function that turn product of beliefs into sums [37].

3 Method

3.1 Weighted modularity

For the community detection task, the common indexes of measuring the quality of communities are modularity and statistical inference, e.g., maximum entropy or maximum likelihood [25, 34]. And Newman showed that the modularity optimization and maximum likelihood are equivalent in community detection problem [25]. The modularity of a partition is a scalar value between -1 and 1 that measures the density of links inside communities as compared to links between communities [34]. In the case of weighted networks, weighted modularity is defined as in [28].

$$Q = \frac{1}{2m} \sum_{i,j} \left(a_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j), \tag{2}$$

where a_{ij} represents the weight of the edge between i and j, k_i is the sum of the weights of the edges attached to vertex i, c_i is the community to which vertex is assigned, the δ-function $\delta(u, v)$ is 1 if $u = v$ and 0 otherwise and $m = \frac{1}{2} \sum_{ij} a_{ij}$. This paper aims at achieving a network partition result with maximum modularity.

Modularization is used to compare different methods on the quality of the partition, and as the optimization objective function [17, 35]. However, it is hard to obtain a partition result with completely modular in dealing with a large network [3]; it is necessary to design an approximate algorithm. The fastest approximation optimization algorithm is proposed in [23]; this method optimizes modular production and circulation of communities. Unfortunately, the used greedy algorithm may produce values obviously lower than that of modular can be found by using, for example, simulated annealing [14]. In addition, the method in [23] tends to contain a large part of nodes, even if there is no significant synthetic network community structure among them. This artifact make it shall not apply to the network of more than one million nodes. A new approach was proposed in [27] to merge communities with balanced size, further speeding up the operation time and enabling them to handle networks with millions of nodes.

3.2 Adjacent matrix diagonal concentrate

Given a graph $G(V, E)$ with node number, whose corresponding adjacent matrix is $A = {a_{ij}}^{n\times n}$, where $a_{ij} = 1$, if there is an edge between i and j; otherwise $a_{ij} = 0$. n is the number of nodes. The adjacent matrix of graph in Fig. 6a is A_1.

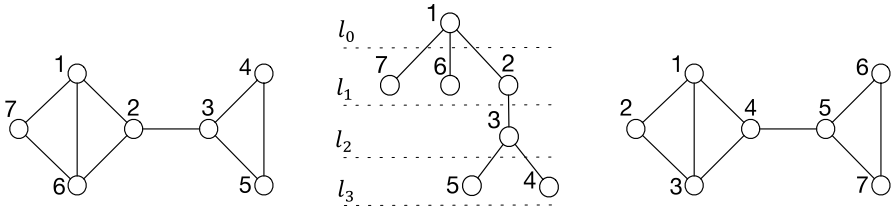

Fig. 6 Graphs with different labels. **a** A graph with seven nodes; **b** A span tree of graph in (**a**) by the BFS; **c** Relabel of nodes

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

and

$$A_2 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

In this paper, concentration degree is introduced to describe the concentration of adjacent matrix A as follow. Then, the concentration degree of A_1 is 1.0472.

$$C_d = \frac{1}{n-1} \sum_{k=1}^{n-1} \sum_{|i-j|=k} \frac{1}{a_{ij}}. \tag{3}$$

To obtain the largest concentration-degree adjacent matrix, it needs to re-number the input graph. This paper introduces a breadth-first-search (BFS)-based graph re-number algorithm. By BFS algorithm, it can obtain a span tree of the input graph; then, it visits all node according to its depth to root node. Suppose the maximum depth of the tree is K, then it obtains the sequence as $\{(S_K), (S_{K-1}), \ldots, (S_2), (S_1)\}$, where (S_k) is the best order of nodes with largest concentration degree in layer k.

For the graph in Fig. 6a, it first performs BFS algorithm to obtain a span tree shown in Fig. 6b, where node 1 is the root. Then, it reorder all node numbers by above method as $\{(5,4),(3),(2,6,7),(1)\}$, and it inverts the sequence as $\{1,7,6,2,3,4,5\}$, i.e., $1 \to 1$, $7 \to 2$, $6 \to 3$, $2 \to 4$, $3 \to 5$, $4 \to 6$, $5 \to 7$. After that, it achieves a

new nodes number of graph as shown in Fig. 6c, and its adjacent matrix is A_2. And the concentration degree of A_2 is 1.2222. That is the concentration degree of A_2 is greater than the concentration degrees of A_1. Actually, among the all adjacent matrixes of isomorphic graphs of the graph in Fig. 6a, A_2 has the maximum concentration-degree.

3.3 Dynamic programming

After obtaining the largest concentration-degree adjacent sequence and matrix, it tries to divide it into several parts such that the modularity is maximum. First, to achieve a better partition, a new binary encoding solution expression is introduced. Given a sequence of nodes as $\{1, 2, 3, \ldots, n\}$, if it is divided as k parts, then, it needs to insert k plates into this sequence, e.x., $\{1, 2, 3, i_1 \mid i_1 + 1, \ldots, i_2 \mid i_2 + 1, \ldots, i_{k-1} \mid i_{k-1} + 1, \ldots, n\}$. To express solution conveniently, this paper introduces binary variable as

$$x = \{1, 0, 0, \underbrace{0}_{i_1}, \underbrace{1}_{i_1+1}, \ldots, \underbrace{0}_{i_2}, \underbrace{1}_{i_2+1}, \ldots, \underbrace{0}_{i_{k-1}}, \underbrace{1}_{i_{k-1}+1}, \ldots, \underbrace{0}_{n}\}$$

From the above n-dimension binary variable, it can obtain the result of the number of communities k and corresponding communities: $c_1 = \{1,\ 2,\ 3, \ldots,\ i_1\}$, $c_2 = \{i_1 + 1,\ i_1 + 2,\ i_1 + 3, \ldots,\ i_2\}$, $\cdots$, $c_k = \{i_{k-1} + 1,\ i_{k-1} + 2,\ i_{k-1} + 3, \ldots,\ n\}$. Then, it can further evaluate the modularity of the result.

To obtain the optimal community detection result with larger modularity, a dynamic programming is proposed to solve the sequence partition problem. The algorithm visits nodes from node 1 to node n, and update solution by the current optimal modularity as follow update formulation, where $x[j]$ is an ordered set:

$$\begin{cases} x[1] = \{\underbrace{1}_{1}\}; \\ x[j] = argmax_{\{1 \le k \le j-1\}}\ Q(x[k] \cup \{\underbrace{1}_{k+1}, 0, 0, \ldots, \underbrace{0}_{j}\}),\ j = 2, 3, \ldots, n \end{cases} \tag{4}$$

The update steps of j-th node are listed as follows, where $x_k, (k = 1, 2, \ldots, j-1)$ denotes the best solution of k-th node:

$$\begin{array}{c} \underbrace{1,\ 2,\ 3, \ldots,\ j-1}_{x_{j-1}} \mid j \\ \underbrace{1,\ 2,\ 3, \ldots,\ j-2}_{x_{j-2}} \mid j-1,\ j \\ \vdots \\ \underbrace{1}_{x_1} \mid 2,\ 3, \ldots,\ j-1,\ j \\ 1,\ 2,\ 3, \ldots,\ j-1,\ j \end{array}$$

The details of dynamic programming are shown in Algorithm 1. According to our dynamic algorithm, it can obtain the maximum modularity result for every node that is following claim holds:

Algorithm 1 Dynamic Programming

Input: Adjacent matrix with largest concentration degree;
Output: Solution $x^* \in \{0,\ 1\}^n$ with maximum modularity;
1: Initialize $x[1] \leftarrow \{1\}$;
2: **for** $j = 2 : n$ **do**
3: $\quad Q(x[j]) \leftarrow -1$;
4: $\quad$ **for** $k = 1 : j-1$ **do**
5: $\quad\quad$ **if** $Q(x[j]) < Q(x[k] \cup \{\underbrace{1}_{k+1}, 0, 0, \ldots, \underbrace{0}_{j}\})$ **then**
6: $\quad\quad\quad x[j] \leftarrow x[k] \cup \{\underbrace{1}_{k+1}, 0, 0, \ldots, \underbrace{0}_{j}\}$
7: $\quad\quad$ **end if**
8: $\quad$ **end for**
9: $\quad$ Save $x[j]$;
10: **end for**
11: $x^* = x[n]$.

Claim 1 *Given an adjacent matrix with node ordering of maximum concentration-degree, our dynamic programming algorithm 1 can achieve the optimal partition of sequence with maximum modularity.*

For the graph in Fig. 6c with adjacent matrix A_2, the each optimal state of dynamic programming is listed in Table 1, where $x[k]$ and $Q(x[k])$ are the current best solution and the best modularity. From Table 1, it can be seen that nodes 1, 2, 3 and 4 are assigned to a community, and nodes 5, 6 and 7 are assigned to another community. At last, with the information of node reordering, it can obtain the community detection result for the graph in Fig. 6a.

4 Results

To evaluate our method, our algorithm is implemented for the community detection problem using Python programming language and ran on a personal computer with 2.7 GHz CPU, 8 GB memory and Unix operating system. To test our method under real-world conditions, four observed real-world networks "Karate club," "Les Mis'erables," "Polbooks" and "Word adjacency" are chosen to our benchmarks,

Table 1 Current best partition in every iterator of matrix A_2

k	1	2	3	4	5	6	7
$x[k]$	{1}	{1, 0}	{1, 0, 0}	{1, 0, 0, 0}	{1, 0, 0, 0, 1}	{1, 0, 0, 0, 1, 0}	{1, 0, 0, 0, 1, 0, 0}
$Q(x[k])$	−1	0.1577	0.3475	0.4429	0.3073	0.2836	0.4429

Table 2 Results for real networks

Networks	$\|V\|$	$\|E\|$	AD	$\|C\|$	C
Karate club	34	78	4.588	2	{{1–17}, {17–34}}
Les Mis'erables	77	254	6.597	6	{{1–16}, {17–24}, {25–46}, {47–68}, {69–76}, {77}}
Polbooks	105	441	8.400	2	{{1–59}, {60–105}}
Word adjacency	112	425	7.589	4	{{1–53}, {54–79}, {80–111}, {112}}

which are provided from Ref. [29]. For the real-world networks, our method can estimate correctly the number of communities and detect communities.

The experimental results are listed as follow Table 2, columns $|V|$, $|E|$ denote the number of nodes and edges of networks, column AD is the average degree of networks, column C denotes the community detection result, and $|C|$ is the number of communities. In this work, modularity is used to measure the quality of community detection result, which range from -1 to 1. Our objective is to achieve a community detection result with maximum modularity. To show the effectiveness of our dynamic programming-based method, it compares the results generated by our method with the Monte Carlo algorithm in Ref. [29]. In Ref. [29], the number of community is decided by the probability density with maximum entropy. And in Ref. [24], Newman showed that the modularity optimization and maximum likelihood are equivalent in community detection problem. For the fair comparison, it further calculates the modularities according to the community detection results in Ref. [29], which are listed in row "Mento Carlo" of Table 3. And the modularities of our results are listed into row "Dynamic Programming" of Table 3. From Table 3, it can be seen that our algorithm achieves greater modularity for networks "Karate club," "Les Mis'erables," "Polbooks" and "Word adjacency." This comparison shows the effectiveness of our dynamic programming method. For network "Word adjacency," both approaches achieve small modularities; the reason is that the adjacent matrix of network "Word adjacency" has lower concentration.

Figure 7 shows the adjacent matrix after concentrating. From Fig. 7, it can be seen that the detected communities cover the densest region in adjacent matrix for these networks. To further describe the effectiveness of our dynamic programming method, the graphs after community detection are plotted in Fig. 8. In Fig. 8a, the network is divided two communities (red and green), it can be seen that the cut edges between two communities are much less than the edges inside

Table 3 Comparisons on modularity with Ref. [29]

Networks	Karate club	Les Mis'erables	Polbooks	Word adjacency
Mento Carlo	0.321	0.418	0.409	0.153
Dynamic programming	0.354	0.461	0.444	0.156

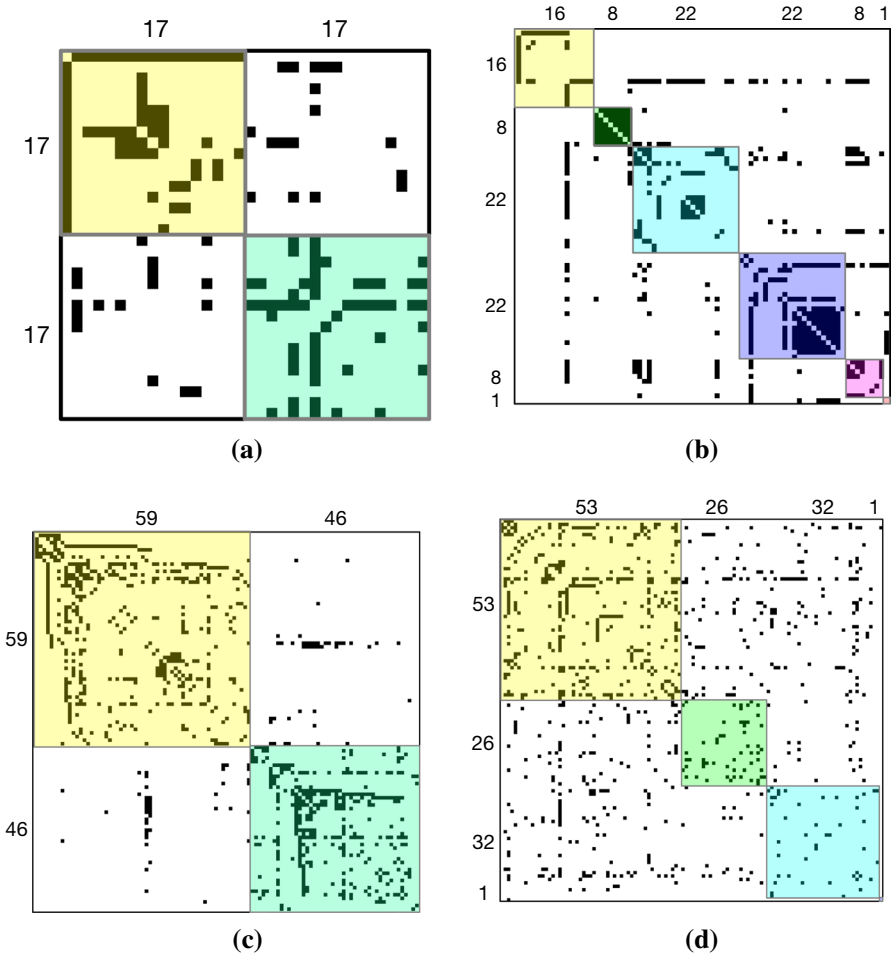

Fig. 7 Results on adjacent matrixes of communities detection. **a** Karate club; **b** Les Mis'erables; **c** Polbooks; **d** Word adjacency

communities. The same for other three graph. These graphs show the effectiveness of our dynamic programming method.

5 Conclusions

In this paper, a dynamic programming algorithm is proposed to concurrently determinate the number of communities and detect communities in a network. This approach is based on a block diagonal dominance matrix. This paper shows the relationship between communities and block diagonal dominance matrix. To obtain a desired community result, it formulates the community detection problem as an integer linear programming, and solves it by designing a simulated annealing algorithm.

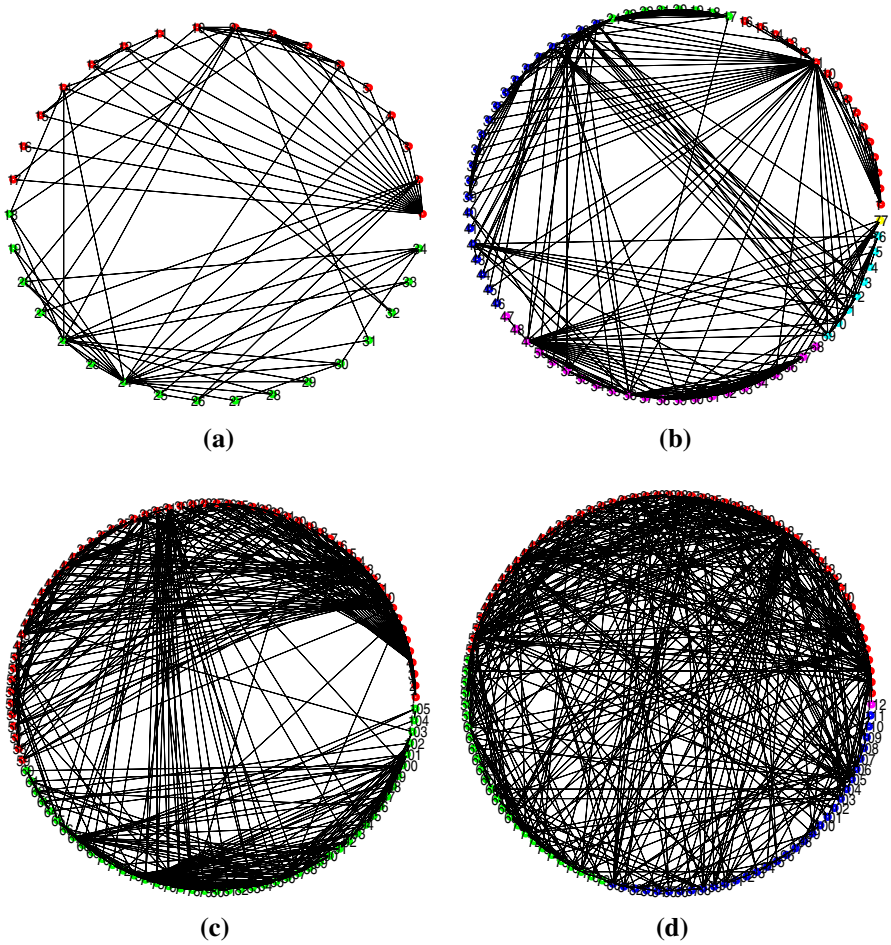

Fig. 8 Results on graphs of communities detection. **a** Karate club; **b** Les Mis'erables; **c** Polbooks; **d** Word adjacency

The tests on a number of real-world networks show the effectiveness of our dynamic programming-based community detection method. Effective community detection algorithm is helpful for designing a scalable, adaptive and survivable trust management protocol for social IoT system.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants (No. 61907024), and the Starting Research Fund from Minnan Normal University (No. KJ18009), and the Open Fund of Key Laboratory of Urban Land Resources Monitoring and Simulation, Ministry of Land and Resources (No. KF201803065).

References

1. Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw 54(15):2787–2805
2. Atzori L, Iera A, Morabito G (2011) SIoT: giving a social structure to the internet of things. IEEE Commun Lett 15(11):1193–1195
3. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech: Theory Exp 10:P10008
4. Bruna J, Li X (2017) Community detection with graph neural networks. arXiv preprint, arXiv :1705.08415
5. Chen T, Singh P, Bassler KE (2018) Network community detection using modularity density measures. J Stat Mech Theory Exp 053406
6. Chen Z, Li L, Bruna J (2019) Supervised community detection with hierarchical graph neural networks. In International Conference on Learning Representations (ICLR)
7. Chen C-M, Xiang B, Liu Y, Wang K-H (2019) A secure authentication protocol for internet of vehicles. IEEE Access 7(1):12047–12057
8. Come E, Latouche P (2015) Model selection and clustering in stochastic block models based on the exact integrated complete data likelihood. Stat Model 15:564–589
9. Daudin JJ, Picard F, Robin S (2008) A mixture model for random graphs. Stat Comput 18:173–183
10. Fortunato S (2010) Community detection in graphs. Phys Rep 486:75–174
11. Fortunato S, Hric D (2016) Community detection in networks: a user guide. Phys Rep 659:1–44
12. Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci USA 99:7821–7826
13. Guo K, Guo W, Chen Y, Qiu Q, Zhang Q (2015) Community discovery by propagating local and global information based on the MapReduce model. Inf Sci 323:73–93
14. Guimera R, Sales-Pardo M (2009) Missing and spurious interactions and the reconstruction of complex networks. Proc Natl Acad Sci USA 106:22073–22078
15. Handcock MS, Raftery AE, Tantrum JM (2007) Model-based clustering for social networks. J R Stat Soc A 170:301–354
16. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. Comput Vis Pattern Recogn, pp 770–778
17. Karrer B, Newman MEJ (2011) Stochastic block models and community structure in networks. Phys Rev E 83:016107
18. Krzakala F, Moore C, Mossel E, Neeman J, Sly A, Zdeborov L, Zhang P (2013) Spectral redemption in clustering sparse networks. Proc Nat Acad Sci 110(52):20935–20940
19. Latouche P, Birmel'e E, Ambroise C (2009) Bayesian methods for graph clustering. In Advances in Data Analysis, Data Handling, and Business Intelligence. Springer, Berlin, pp 229–239
20. Latouche P, Birmele E, Ambroise C (2012) Variational Bayesian inference and complexity control for stochastic block models. Stat Model 12:93–115
21. Luo F, Guo W, Yu Y, Chen G (2016) A multi-label classification algorithm based on kernel extreme learning machine. Neurocomputing 260:313–320
22. Massouli'e L (2014) Community detection thresholds and the weak Ramanujan property. In Proceedings of the 46th Annual ACM Symposium on the Theory of Computing, pp 694–703, Association of Computing Machinery, New York
23. McDaid AF, Murphy TB, Friel N, Hurley N (2013) Improved Bayesian inference for the stochastic block model with application to large networks. Comput Stat Data Anal 60:12–31
24. Newman MEJ, Reinert G (2016) Estimating the number of communities in a network. Phys Rev Lett 117:078301
25. Newman M E J (2016) Community detection in networks: Modularity optimization and maximum likelihood are equivalent. arXiv preprint arXiv:1606.02319
26. Pan J-S, Lee C-Y, Sghaier A, Zeghid M, Xie J (2019) Novel systolization of subquadratic space complexity multipliers based on toeplitz matrix-vector product approach. IEEE Trans Very Large Scale Int Syst 27(7):1614–1622
27. Peixoto TP (2014) Hierarchical block structures and high-resolution model selection in large networks. Phys Rev X 4:011047
28. Peixoto TP (2017) Nonparametric Bayesian inference of the microcanonical stochastic block model. Phys Rev E 95:012317

29. Riolo M A, Cantwell G T, Reinert G, Newman M E J (2017) Efficient method for estimating the number of communities in a network. arXiv:1706.02324v1
30. Sukhbaatar S, Szlam A, Fergus R (2016) Learning multiagent communication with backpropagation. Adv Neural Inf Process Syst, pp 2244–2252
31. Tai K, Socher R, Manning C (2015) Improved semantic representations from tree-structured long short-term memory networks. Association for Computational Linguistics (ACL), pp 1556–1566
32. Wang S, Guo W (2017) Sparse multi-graph embedding for multimodal feature representation. IEEE Trans Multimedia 19(7):1454–1466
33. Wu T-Y, Chen C-M, Wang K-H, Meng C, Wang EK (2019) A provably secure certificateless public key encryption with keyword search. J Chinese Inst Eng 42(1):20–28
34. Yan X (2016) Bayesian model selection of stochastic block models. In Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp 323–328
35. Yang J, Leskovec J (2015) Defining and evaluating network communities based on ground-truth. Knowl Inf Syst 42:181–213
36. Yang Y, Liu X, Zheng X, Rong C, Guo W (2018) Efficient traceable authorization search system for secure cloud storage. IEEE Trans Cloud Comput. https://doi.org/10.1109/TCC.2018.2820714
37. Ye F, Chen C, Wen Z, Zheng Z, Chen W, Zhou Y (2019) Homophily preserving community detection. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2019.2933850