Information Sciences 551 (2021) 358–372

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A community detection algorithm based on graph compression for large-scale social networks

Xingwang Zhao [a], Jiye Liang [a,*], Jie Wang [a,b]

[a] *Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan, 030006 Shanxi, China*
[b] *School of Information, Shanxi University of Finance and Economics, Taiyuan, 030006 Shanxi, China*

ARTICLE INFO

Article history:
Received 24 January 2020
Received in revised form 21 October 2020
Accepted 24 October 2020
Available online 13 November 2020

Keywords:
Social networks
Community detection
Graph clustering
Graph compression
Community seeds

ABSTRACT

Uncovering the underlying community structure of a social network is an important task in social network analysis. To solve this problem, many community detection algorithms for the full topology of an original social network have been proposed. However, these algorithms are not very effective at analyzing large-scale social networks. To overcome this deficiency, this paper proposes a community detection algorithm based on graph compression. Specifically, a compressed graph is first obtained by iteratively merging vertices with a degree of 1 or 2 into their neighbors with a higher degree. Then, two indices, i.e., the density and quality of vertices, are defined to evaluate the probability of vertices as community seeds. By considering these two measures together, in a compressed social network, the number of communities and the corresponding initial community seeds are determined simultaneously. After obtaining the community structure of the compressed social network via seed expansion, the community results are propagated to the original social network. Extensive experiments conducted on various social networks have demonstrated the superiority of our proposal compared to several existing state-of-the-art community detection algorithms.

1. Introduction

Community detection, also called graph clustering, is one of the most fundamental and vital complex network analysis techniques used to illustrate the structure of the relationship of network nodes. It involves identifying the number of communities in a complex network and the membership of each node, with more interactions among the same community than between its community nodes and the remainder of the network. Some of the practical applications of community detection range from the analysis of social networks to the analysis of Protein–Protein Interaction (PPI) networks. A growing number of community detection methods for different types of networks have recently been developed in the literature. We refer the reader to surveys of community detection or graph clustering algorithms and their applications [8,14,23,40].

With the development of information technology, the use of social networks in our society is exponentially growing. One consequence is a deep change in how people react to events and interact with each other. The community structure is a significant property of social networks [8]. It often represents specific organized groups of users with similar attributes, hobbies

* Corresponding author.
E-mail addresses: zhaoxw84@163.com (X. Zhao), ljy@sxu.edu.cn (J. Liang), xhcwj@sina.com (J. Wang).

https://doi.org/10.1016/j.ins.2020.10.057
0020-0255/© 2020 Elsevier Inc. All rights reserved.

or closer relationships. Therefore, detecting the underlying community structure of social networks is an important task for their analysis.

Recently, the community detection of social networks has attracted immense attention in both academia and industry. However, as data have accumulated, the scale of social networks is gradually growing and becoming huge. This leads to the problem that the community detection of large-scale social networks cannot be resolved with traditional algorithms in terms of time and spatial complexity. That is, most existing algorithms cannot be scaled to the massive size of today's social networks. The scalability of community detection algorithms is a critical issue.

To address this challenge, a considerable amount of research has been conducted in the literature [4,5,32,41,44]. These methods can be divided into parallel community detection algorithms, local community discovery algorithms and network scale reduction-based community detection algorithms. Note that, local community detection is also called community search [22,7,28]. It refers to the identification of a community that is specific to a query node and relies on limited information about the network structure. In the third category of methods, sampling is one of the most intuitive technologies used to reduce the network scale. However, the final results are greatly influenced by the representativeness of the subgraphs selected by the sampling technique. Therefore, some researchers have developed other methods to reduce the size of the network to speed up community detection in recent years. For example, Satuluri et al. [39] developed a graph sparsification technique to speed up existing graph clustering algorithms. In this method, by removing some edges that are likely to be between communities, they reduced the number of edges in the graph while retaining the community structure of the network. Macropol et al. [30] proposed a new technique, Top Graph Clusters (TGCs), to search for the best communities for a large network. This method only found a subset of best clusters, not all clusters in the entire graph, and save timed and memory by pruning the search space. Using the multilevel graph clustering framework, Abou-Rjeili et al. [1] proposed new coarsening strategies, which allow arbitrary sized sets of vertices to be collapsed together, to resolve the graph clustering problem. The above algorithms have made some achievements in solving large-scale networks. However, the core process of most existing community detection algorithms is based on the full network topology. These algorithms still have high computational complexity. Furthermore, in the community detection process, these methods do not make full use of the structural characteristics of social networks. Hence, it is necessary to propose a new network scale reduction method for community detection that is related to social network characteristics.

From the edge connectivity perspective, it is well known that the degree distribution in many social networks follows a power-law distribution in many social networks [3,47]. This means that a small set of nodes has a higher influence than the others. A reasonable assumption based on previous research is that certain communities may have some important nodes with higher influence. A user's community is often the same as its neighbors' in most social networks. Intuitively, the above assumptions suggest that we can identify the community structure of a large-scale social network using a graph compression strategy by merging the vertices with lower degrees to one of its neighbors with a higher degree.

Based on the above motivation, this paper proposes a community detection algorithm based on graph compression for large-scale social networks. First, by iteratively merging vertices with a degree of 1 or 2 into their neighbors with a higher degree, a compressed graph is obtained. Then, two indices, namely, the density and quality of vertices, are defined to evaluate the probability of a vertex as a community seed. In the compressed social network, vertices with a high degree or a high quality are automatically selected as the initial community seeds and the number of communities is determined simultaneously. After obtaining the community structure of the compressed social network, we propagate the community result to the rest of the social network. The experimental results show that the developed algorithm produces high quality communities comparable to or better than existing state-of-the-art community detection algorithms while being much faster.

The main contributions of the present paper are as follows:

(1) A simple, efficient and lossless graph compression method for social networks is proposed. This method can greatly reduce the storage space and computing time for community detection. On the experimental data set, the compression ratio can reach 40% on average.
(2) A community detection mechanism, which can automatically detect the number of communities and find initial community seeds, is developed based on the density and quality of vertices.

The rest of this paper is organized as follows: Section 2 reviews the related work on community detection for social networks. The proposed community detection algorithm is developed in Section 3. Then, Section 4 evaluates the proposed algorithm compared to other community detection algorithms applied to social networks. Section 5 concludes this paper and discusses the future work briefly.

2. Related work

Due to the importance of community structures and their enormous applicability in different domains, many community detection methods have been developed from various perspectives [8,23]. These methods can be broadly categorized into modularity optimization, spectral clustering, label propagation, nonnegative matrix factorization, block model approximation, and latent space models; and they are described below.

- **Modularity optimization methods.** Among the existing community detection algorithms, this is the most commonly used type of method. It is based on the modularity measure [34], a commonly used criterion for community detection. It measures the density of the connections within modules (communities) compared to the density of the connections outside of those modules. These algorithms seek to find communities with a high modularity value using heuristics. These algorithms mainly apply different hierarchical clustering strategies to partition networks, which is very time-consuming. One of the most commonly used modularity optimization-based algorithms for community detection, was proposed by Girvan et al. [16].
- **Spectral clustering methods.** These methods are derived from graph partitioning problems and have become one of the most popular community detection algorithms in recent years [42,43]. In order to detect communities, spectral clustering does not use the geometric information of the graph and does not operate on the graph itself, but rather it operates on a mathematical representation of the graph. One advantageous feature of spectral clustering is that the decision about each vertex is made based on a more global view of the problem.
- **Label propagation methods.** In these methods, each node attains its label according to the label information of its neighborhood nodes. The first label propagation algorithm (LPA) was proposed by Raghavan et al. [36,15], in which each node tries to achieve a label from the highest number of labels possessed by its neighbors. Due to its nearly linear time complexity in determining the community structure, it has received more attention recently. However, the convergence speed and clustering effectiveness of the algorithm are very sensitive to the update order of the label information. Recently, LPA was further improved in [6,20].
- **Nonnegative matrix factorization methods.** Nonnegative matrix factorization (NMF) is an unsupervised machine learning method. More recently, it has been extensively used and extended for different variations of community detection problems [26,48]. Inspired by the generative process of networks, NMF-based methods assume that one network can be divided into a number of low-dimensional subspaces. The coefficient vectors in the new space are soft membership vectors that determine the relationships between all pairs of nodes and communities.
- **Block model approximation methods.** This method approximates a given network via a block structure by reordering the index of each node according to their community membership [13]. Then, each block represents a community.
- **Latent space model methods.** These methods map the nodes in a network into a low-dimensional Euclidean space such that the proximity between nodes based on network connectivity is kept in the new space [19]. Then, the nodes are clustered in the low-dimensional space using traditional clustering methods such as the *k*-means algorithm. In recent years, network embedding methods, e.g., node2vec [17], LINE [46],etc., have become popular for the community detection of networks.

In recent years, the increasing size and complexity of networks have made most of the above mentioned community detection algorithms unworkable. In order to address these problems, various novel community detection technologies have emerged. For example, parallel methods, divide-and-conquer methods, local structure-based methods and condensation-based methods have been proposed to resolve the scalability of community detection algorithms for large-scale networks. As a kind of complex network, an attributed graph includes both a topological structure and node attribute information. Recently, some game theory-based methods and multiagent-based methods have been used to detect communities in attributed graphs [12,11,10].

3. The proposed community detection algorithm

Social networks can be modeled as graphs, where the nodes and the edges represent the individual users and the social relationships between them, respectively. Formally, a graph G is represented as $G = (V, E, W)$, where $V = \{v_1, \ldots, v_n\}$ is a set of n vertices (or nodes), $E \subseteq V \times V$ is a set of m edges, and $W : E \to R_+$ is the (nonnegative) weight function over the edge set. The vertices in G correspond to data points, edges represent pairwise relations, and edge weights reflect the strength of pairwise relations. Specifically, for an unweighted graph, if there is an edge between two vertices, the corresponding weight is 1. The edge between vertices v_i and $v_j(v_i, v_j \in V)$ can be expressed as an unordered pair (v_i, v_j) or $e_{i,j}$. As is customary, we represent a graph G with the corresponding weighted adjacency matrix A, which is also called the affinity matrix. More specifically, for an undirected graph, A is an $n \times n$ symmetric matrix, where $A_{ij} = A_{ji} = W(v_i, v_j)$ if $(v_i, v_j) \in E$, and $A_{ij} = 0$ otherwise. Clearly, if there are no self-loops, all the diagonal elements of A are zeros. In this paper, we only consider undirected graphs with no self-loops. $N(v_i)$ and $d(v_i)$ represent a set of neighbors and degrees of vertex $v_i \in V$, respectively. That is, $N(v_i) = \{v_j | a_{i,j} \neq 0, v_j \in V\}, d(v_i) = |N(v_i)|$. Formally, given a graph $G = (V, E, W)$, the goal of traditional, exhaustive community detection is to partition graph G into k disjoint communities $\mathscr{C} = \{C_1, C_2, \ldots, C_k\}$ such that $C_1 \cup C_2, \ldots, \cup C_k = V$, where there are no interactions between different communities. While nonoverlapping community detection traditionally finds exhaustive and disjoint communities, the overlapping community detection algorithm finds overlapping communities that are not necessarily exhaustive. Formally, we seek k overlapping communities $\mathscr{C} = \{C_1, C_2, \ldots, C_k\}$, such that $C_1 \cup C_2, \ldots, \cup C_k \subseteq V$, where a vertex may belong to more than one community [47]. This paper mainly focuses on traditional nonoverlapping community detection.

An overview of the proposed algorithm is shown in Fig. 1. It consists of four steps: graph compression, community seed determination, seed expansion, and community structure propagation. The algorithm is abbreviated as CDEP (graph Com-

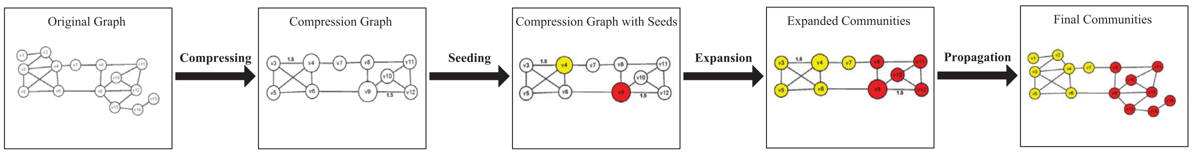

Fig. 1. Overview of the proposed algorithm.

pression, community seed Determination, seed Expansion, and community structure Propagation). In the graph compression step, a compressed graph is obtained by iteratively merging vertices with a degree of 1 or 2 into their neighbors with higher degrees. In the community seed determination step, the number of communities and initial seeds of communities in the compressed graph are determined automatically. Then, the community structure of the compressed graph is obtained via seed expansion. Finally, in the community structure propagation step, we further expand the communities to the hidden vertices that were compressed in the first step.

3.1. Graph compression

As technology advances, the scale of the social network data we collect and archive increases continuously. In order to process enormous social networks, efficient computational methods for compressing and simplifying data by reducing the amount of I/O accesses and the required storage are becoming vital. Recently, some graph compression techniques for social networks have been studied in the literature [27]. First, most of these methods reuse the compression schemes developed for web graphs. The works on web graph compression often use the URL lexicographic order and various encoding techniques. However, these methods are not necessarily suitable for social networks because of the different characteristics of web graphs and social networks. Second, existing compression methods for social networks for query operations on graphs have been proposed. Different operations on graphs have different characteristics and great differences. For different operations, we usually need to design specific compression methods to achieve better results. Thus, designing a proper compression method for community detection is necessary.

In social networks, a power-law degree distribution indicates that a small set of high degree vertices covers a large portion of the edges in the network. Most of the low degree vertices in the network tend to be attached to high degree vertices. Inspired by this, this step will compress the original graph based on the characteristics of a social network. The goal of this step is to obtain a compressed graph, which is much smaller than the original graph. This step is divided into two phases that are repeated iteratively. In the first phase, we iteratively randomly visit a vertex with a degree of 1 and merge this vertex into its neighbor. The edge between them is removed, and the resulting vertex is considered a supernode that has more than one object. In the second phase, the vertices whose degree is 2 are merged into one of their neighbors with a larger degree. Then, an edge may be added or the edge weight between two neighboring vertices of this vertex may be updated. If the two neighbors of the vertex are not connected, then there will be a new edge between them after compression. If there is an edge before compression, the edge weight between these two vertices will be strengthened. Suppose that the neighboring vertices of vertex $v_i \in V$ are v_j and v_k $(v_j, v_k \in V)$. The new weight $W'(v_j, v_k)$ between v_j and v_k after compressing is defined as

$$W'(v_j, v_k) = W(v_j, v_k) + \frac{1}{2} \cdot W(v_i, v_j) \cdot W(v_i, v_k). \tag{1}$$

However, some vertices with a degree of 2 may be the key nodes connecting different communities, which cannot be compressed. These nodes are called “bridge nodes”, which are defined as follows. For vertex v_i, where $d(v_i) = 2$, suppose that its neighbors are vertices v_j and v_k, i.e., $(v_i, v_j), (v_i, v_k) \in E$. If $(v_j, v_k) \notin E$, vertex $v_i \in V$ is a bridge node and cannot be compressed.

In the following, taking Fig. 2 as an example, we briefly describe the operation of the compression step. For the original graph in Fig. 2(a), there are 15 vertices and 23 edges. In the first phase, the vertexes with a degree of 1 are compressed. First, vertices v_1 and v_{15} are merged into their neighbors v_2 and v_{14}, respectively. Next, the corresponding edges are removed. Then vertex v_{14} and included vertex v_{15} are merged into v_{13}. The edge (v_{13}, v_{14}) is removed. In the second phase, some vertices $v = \{v' | d(v') = 2, v' \in V\}$ are compressed. Vertex v_2 and included vertex v_1 are merged into v_4, whose degree is larger than the degree of vertex v_3. The weight of the edge (v_3, v_4) is updated to 1.5 according to Eq. (1). The neighbors of vertex v_{13} are vertices v_9 and v_{12}. Because the degree of vertex v_9 is larger than the degree of vertex v_{12}, vertex v_{13} and included vertex set $\{v_{14}, v_{15}\}$ are merged with vertex v_9. Similarly, $W(v_9, v_{12}) = 1.5$ according to Eq. (1). For vertex v_7, because it is a bridge node, it cannot be compressed. The compressed graph is shown in Fig. 2(b). There are 10 vertices and 16 edges in the compressed graph. The edge weights $W(v_3, v_4)$ and $W(v_9, v_{12})$ are 1.5. The other edge weights are 1. The sets of the included vertices for vertices v_4 and v_9 are $\{v_1, v_2, v_4\}$ and $\{v_9, v_{13}, v_{14}, v_{15}\}$, respectively.

As an illustration of the step, we further apply the graph compression step on Zachary’s Karate Club network, which is a classical example for testing community detection algorithms. This network represents the friendship relationships between 34 members in a karate club at a US university in 1970. In this network, as shown in Fig. 3(a), vertex 1 and vertex 34 are

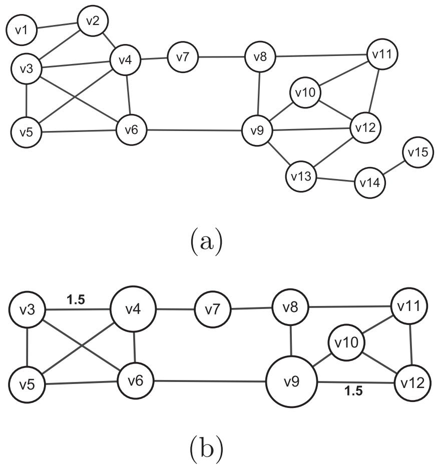

Fig. 2. Comparison of the original graph and the compressed graph: (a) the original graph; (b) the compressed graph.

known to be the instructor and the student founder of the club, respectively. These two vertices are central in the network forming two natural clusters around them. When we run this step on the original graph, some vertices, including $10, 12, 13, 15, 16, 17, 18, 19, 21, 22, 23, 27$, are compressed. Fig. 3(b) shows the compressed graph of Zachary's Karate Club network. We can see that there are 22 vertices and 56 edges in the compressed graph. The edge weights of $e_{1,2}, e_{1,4}, e_{6,7}, e_{3,34}, e_{33,34}$ and $e_{30,33}$ are $2, 1.5, 1.5, 0.5, 3.5$ and 1.5, respectively. The other weights are 1.

The detailed algorithmic description of this step is shown in Algorithm 1.

Algorithm 1. The Graph Compression Algorithm

```
Input: Graph $G = (V, E, W)$.
Output: Compressed graph $G^c = (V^c, E^c, W^c)$.
1: Compute degree for all vertices $v_i \in V$, i.e., $d(v_i) = |\{v_j | (v_i, v_j) \in E, v_j \in V\}|$;
2: Initialize the sets of including vertices for each vertex $v_i \in V$ after compressing, i.e., $IV(v_i) = \{v_i\}$;
3: Initialize sets of vertices with a degree of 1 and 2, respectively, i.e.,
   $D1 = \{v_i | d(v_i) = 1, v_i \in V\}, D2 = \{v_i | d(v_i) = 2, v_i \in V\}$.
4: Initialize the compressed graph $G^c = G$;
5: repeat
6:   for each $v_i \in D1$
7:     Update $V^c = V^c - \{v_i\}, E^c = E^c - \{(v_i, v_j)\}$, where $(v_i, v_j) \in E^c$;
8:     Update $IV(v_j) = IV(v_j) \bigcup \{v_i\}, d(v_i) = 0, d(v_j) = d(v_j) - 1$;
9:     Add the vertex $v_j$ into $D1$ or $D2$ according to its degree;
10:    Update $D1 = D1 - \{v_i\}$;
11:  end for
12:  for every $v_i \in D2$
13:    if $v_i$ is not a bridge node
14:      Update $V^c = V^c - \{v_i\}, E^c = E^c - \{(v_i, v_j), (v_i, v_k)\}, E^c = E^c \bigcup \{v_j, v_k\}$, where $(v_i, v_j), (v_i, v_k) \in E^c$;
15:      if $W^c(v_j, v_k) > 0$
16:        $d(v_j) = d(v_j) - 1, d(v_k) = d(v_k) - 1$;
17:      end if
18:      Compute new weight $W^c(v_j, v_k)$ between the vertex $v_j$ and $v_k$ according to Eq. (1);
19:      Update $d(v_i) = 0$ and $IV(v_j) = IV(v_j) \bigcup \{v_i\}$, where $d(v_j) \geqslant d(v_k), (v_i, v_j), (v_i, v_k) \in E^c$;
20:      Add the vertices $v_j$ and $v_k$ into $D1$ or $D2$ according to their degree;
21:    end if
22:    Update $D2 = D2 - \{v_i\}$;
23:  end for
24: until $D1 = \varnothing$ and $D2 = \varnothing$;
25: return Compressed graph $G^c = (V^c, E^c, W^c)$.
```

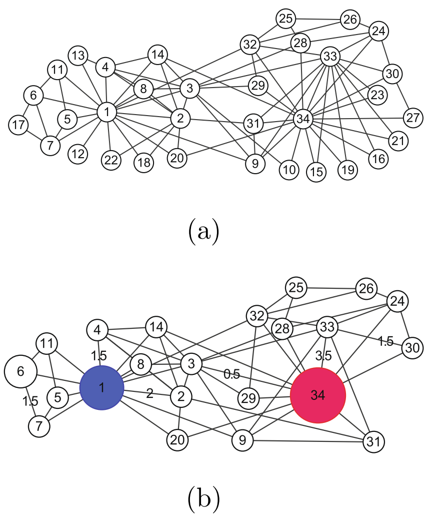

Fig. 3. The illustration of graph compression on Karate Club network: (a) the original graph of Karate Club network; (b) the compressed graph of Karate Club network.

3.2. Community seed determination

The goal of this step is to determine the number of communities and select a set of seeds in the compressed graph. We extend the automatic density peaks clustering algorithm [38], which relies on the idea that cluster centers are characterized by a higher density than their neighbors and by a relatively large distance from objects with higher densities. The objects with high density and distance values are also called peaks. Then, a decision graph is introduced to help the user to determine the cluster centers. Inspired by the idea of this method, an automatic community seed determination algorithm is proposed in this section. In the proposed method, the initial community seeds and the number of communities are determined simultaneously. First, two indices, i.e., the density and quality of vertices, are defined to evaluate the probability of a vertex as a community center. Here, we use the degree of a vertex to reflect its density. The quality of a vertex is defined as the number of vertices contained in the vertex. The higher a vertex's degree is, the higher its density. The higher a vertex's quality is, the more vertices it contains. Therefore, the density and quality of a vertex $v_i \in V^c$ in a compressed graph $G^c = (V^c, E^c, W^c)$ are defined as $\rho(v_i) = d(v_i)$ and $\mu(v_i) = |IV(v_i)|$, respectively. Specifically, for a vertex $v_i \in V^c$, if $\rho(v_i) = 0$ and $\mu(v_i) > 1$, then the density of this vertex is redefined as $\rho(v_i) = \frac{\sum_{v \in N(v_i)} d(v)}{|N(v_i)|}$, where $N(v_i)$ is a set of neighbors of v_i in the original graph. The vertices with a high $\rho(v_i)$ and $\mu(v_i)$ are more likely to be community seeds. Note that the ranges of these two indices are different. In order to consistently measure the probability, the density and quality for a vertex $v_i \in V^c$ are normalized to $[0, 1]$ using the following equations:

$$\rho'(v_i) = \frac{\rho(v_i)}{\max\limits_{v \in V^c} \rho(v)} \tag{2}$$

and

$$\mu'(v_i) = \frac{\mu(v_i)}{\max\limits_{v \in V^c} \mu(v)}. \tag{3}$$

Similar to the method in [38], the decision graph of the density and quality for the compressed graph is shown in Fig. 4. We find that the density and quality maxima are at vertices v_4 and v_9, which are identified as community seeds. Hence, as anticipated, the only vertices with a high ρ' and μ' are the community seeds. Unfortunately, in the case of a large graph with large vertices, the decision graph is not sufficient to identify the community seeds, and the number of communities and community seeds still must be determined automatically.

With these two indices ρ' and μ', we measure the possibility of a vertex being a community seed using a new centrality index γ, which is defined as

$$\gamma(v_i) = \rho'(v_i) * \mu'(v_i), v_i \in V^c. \tag{4}$$

When $\rho'(v_i)$ or $\mu'(v_i)$ is larger, the value of $\gamma(v_i)$ will be larger. Therefore, the vertices with a large γ value are recognized as the community seeds.

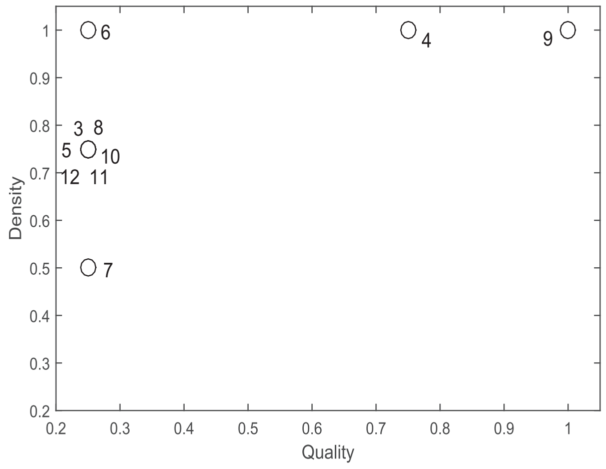

Fig. 4. Decision graph for the compressed graph in Fig. 2(b).

With this in mind, then, by analyzing the distribution of this index, we transform the community seed identification problem into an extreme-value detection problem using the second order difference method to identify the community seeds automatically. We first construct a decreasing sequence $g_1 \geqslant g_2 \geqslant \cdots \geqslant g_{n_c}$, where n_c is the number of vertices of the compressed graph, g_1 is the highest value in $\{\gamma(v_i), v_i \in V^c\}, g_2$ is the second highest value, and so on. The second order difference of the decreasing sequence is defined as

$$h_i = |(g_i - g_{i+1}) - (g_{i+1} - g_{i+2})|, \quad i = 1, \ldots, n_c - 2. \tag{5}$$

The second order difference curve with the number of community seeds for the compressed graph of Fig. 2(b) is given in Fig. 5. As seen in this figure, when the horizontal axis is 2, there is a knee point. Thus, the knee point of the second order difference sequence can be defined as

$$kp = \underset{i=n_c-2,\ldots,1}{argmax}\, h_i. \tag{6}$$

This means that the difference between $g_{1,\ldots,kp}$ and $g_{kp+1,\ldots,n_c}$ is larger. Note that there may be multiple highest values in the second order difference sequence. In order to identify more community seeds, the knee point (the highest value) is found from large to small according to the subscript value in the sequence. Therefore, the potential community seeds CS are determined by

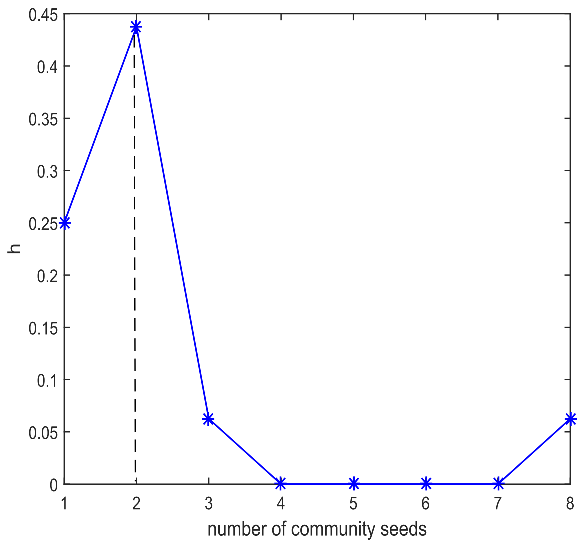

Fig. 5. Second order difference curve with the number of community for the compressed graph.

$$CS = \{v_i | \gamma(v_i) \geqslant g_{kp}, v_i \in V^c\}. \tag{7}$$

If two different potential community seeds are connected by an edge, they will be merged. From the calculation, the community seeds for the compressed graph of Fig. 2(b) are vertices v_4 and v_9. Similarly, for Zachary's Karate Club network shown in Fig. 3(b), the seeds of the communities are vertices 1 and 34, which is consistent with the ground truth.

In the following, Algorithm 2 shows the pseudocode description of this step.

Algorithm 2. The Community Seed Determination Algorithm

```
Input: Compressed graph $G^c = (V^c, E^c, W^c)$ with $n_c$ vertices.
Output: Community seeds $CS$.
1: Compute the density and quality of vertices $v_i \in V^c$ and normalize them using Eqs. (2) and (3), respectively;
2: Compute the index $\gamma(v_i), v_i \in V^c$ according to Eq. (4), and construct a decreasing sequence $g_1 \geqslant g_2 \geqslant \cdots \geqslant g_{n_c}$;
3: Compute the second order difference sequence $h$ according to Eq. (5);
4: Find the knee point of the second order difference sequence according to Eq. (6);
5: Determine the potential community seeds $CS$ according to Eq. (7);
6: if $W^c(v_i, v_j) \neq 0, v_i, v_j \in CS$
7: $CS = CS - \{v_i\}$.
8: end if
9: return Community seeds $CS$.
```

3.3. Seed expansion

After determining a set of community seeds $CS = \{cs_1, cs_2, \cdots, cs_k\}$, where k is the number of communities, we wish to expand them to obtain the corresponding community structure $\mathscr{C}^c = \{C_1, C_2, \ldots, C_k\}$. A simple and effective technique for this task is iteratively assigning labels to unlabeled vertices with the label information of their neighbors according to some criterion. A key concept in seed expansion is the similarity measure between a vertex and a community, which to a large extent determines the community structure. Existing similarity measures typically include the weight or the number of common neighbors between two vertices. However, one limitation of these similarity measures is that they usually do not take into account the fine local topological structures of the network, such as the connection vertexes among the neighbors of a vertex, and the connections among the important vertices. This information is crucial in determining the right community structures.

Adamic and Adar [2] proposed a measure to evaluate the association between personal home pages. In the proposed measure, the features of the compared pages p_a and p_b are used to estimate their similarity score, $score(p_a, p_b)$, as follows:

$$score(p_a, p_b) = \sum_{z_c \in Z} \frac{1}{\log(frequency(z_c))}, \tag{8}$$

where Z denotes the set of features shared by home pages p_a and p_b, and $frequency(z_c)$ represents the number of times z_c appears in the studied set of pages. Note that the method gives high weights to rare features and low weights to features that are common to most of the pages. Inspired by this measure, this paper proposes a new similarity measure that takes into account not only the weights between the vertices, but also the degree information of their common neighbors. Specifically, the quality of the common vertex is determined by the rarity of links connecting itself to other vertices. The neighbor set of the current community $C \in \mathscr{C}^c$ is $CN(C) = \{v_j | (v_i, v_j) \in E, v_i \in C, v_j \notin C\}$. The proposed similarity measure between a vertex $u \in V^c$ and a community $C \in \mathscr{C}^c$ is defined as

$$\begin{aligned} sim(u, C) = & \sum_{v \in N(u) \bigcap C} W^c(u, v) + \\ & \sum_{v \in N(u) \bigcap C} \sum_{v' \in N(u) \bigcap N(v)} \frac{1}{\sum_{v'' \in N(v')} W^c(v', v'')}. \end{aligned} \tag{9}$$

In the above similarity measure, $N(u) \bigcap C$ represents the vertex set of vertex u connected to community C. $N(u) \bigcap N(v)$ denotes the common neighbors of vertices u and v. $\frac{1}{\sum_{v'' \in N(v')} W^c(v', v'')}$ expresses the importance for one of the common neighbors v'. This similarity measure is composed of two terms. The first term $\sum_{v \in N(u) \bigcap C} W^c(u, v)$ denotes the sum of weights between the unlabeled vertex u and the community C in the compressed graph. The second term $\sum_{v \in N(u) \bigcap C} \sum_{v' \in N(u) \bigcap N(v)} \frac{1}{\sum_{v'' \in N(v')} W^c(v', v'')}$ is used to measure the similarity from the point of view of a common neighbor.

The vertex assignment to communities on the compressed graph in Fig. 2(b) is illustrated as follows. After the community seed determination step, the community seeds of this compressed graph are v_4 and v_9. That is, the initial communities are $\mathscr{C}^c = \{C_1, C_2\}$, where $C_1 = \{v_4\}$ and $C_2 = \{v_9\}$. In the first iteration of this step, the union of the neighbors of the initial communities is $\{v_3, v_5, v_6, v_7, v_8, v_{10}, v_{12}\}$. Intuitively, except for vertex v_6, the rest of the vertices are directly connected only to a community. For the vertex v_6, according to Eq. (9), $sim(v_6, C_1) = 1 + \frac{1}{1+1+1.5} + \frac{1}{1+1+1} = 1.619$, and $sim(v_6, C_2) = 1$. Therefore, vertex v_6 will go with community C_1 with the highes similarity. In addition, vertices $\{v_3, v_5, v_7\}$ and $\{v_8, v_{10}, v_{12}\}$ are assigned to communities C_1 and C_2, respectively. In the second iteration, only vertex v_{11} is unassigned. Because it only connects to community C_2, it will be added to C_2. The community structure of the compressed graph is $\mathscr{C}^c = \{C_1, C_2\}$, where $C_1 = \{v_3, v_4, v_5, v_6, v_7\}$ and $C_2 = \{v_8, v_9, v_{10}, v_{12}\}$. The graph is shown in Fig. 6(a).

The pseudocode description of this step is shown in the following Algorithm 3.

Algorithm 3. The Seed Expansion Algorithm

```
Input: Compressed graph $G^c = (V^c, E^c, W^c)$ and its community seeds $CS = \{cs_1, cs_2, \dots, cs_k\}$.
Output: Communities $\mathscr{C}^c$.
1: Initialize communities $\mathscr{C}^c = \{C_1, C_2, \dots, C_k\}$, where $C_i = \{cs_i\}, i = 1, 2, \dots, k$;
2: Initialize a set of the unlabeled vertices $UL = V^c - CS$;
3: while $UL \neq \varnothing$
4: $\mathscr{TC} = \{TC_1, TC_2, \dots, TC_k\}$, where $TC_i = \varnothing$;
5: Compute common neighbors of the communities $CV = \bigcup CN(C_i),\ i = 1, 2, \dots, k$;
6: for each vertex $u \in CV$
7: if $u$ is unlabeled
8: if $N(u) \subseteq C_i,\ i = 1, 2, \dots, k$
9: $TC_i = TC_i \bigcup \{u\}$;
10: else
11: $h = \underset{i=1,2,\dots,k}{argmax} sim(u, C_i), TC_h = TC_h \bigcup \{u\}$.
12: end if
13: $UL = UL - \{u\}$;
14: end if
15: end for
16: Update $C_i = C_i \bigcup TC_i,\ i = 1, 2, \dots, k$.
17: end file
18: return Communities $\mathscr{C}^c$.
```

3.4. Community structure propagation

Once the communities on the compressed graph are obtained, we further expand each of the communities to the compressed vertices, which were merged in the graph compression step. This community propagation step is straightforward.

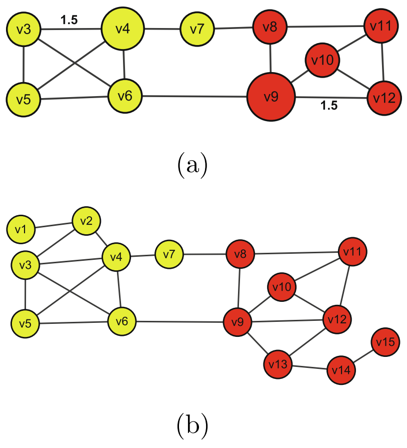

Fig. 6. Community of the original graph and the compressed graph in Fig. 2: (a) the compressed graph; (b) the. original graph.

The unlabeled vertices are assigned to corresponding communities according to the category information of the vertex that these vertices have been merged into in the graph compression step. For the graph in Fig. 2(a), the sets of vertices $\{v_1, v_2\}$ and $\{v_{13}, v_{14}, v_{15}\}$ are merged into v_4 and v_9 after the graph compression step, respectively. v_4 and v_9 belong to communities C_1 and C_2, respectively. Therefore, $\{v_1, v_2\}$ and $\{v_{13}, v_{14}, v_{15}\}$ are added into communities C_1 and C_2, respectively. The community structure of the original graph is shown in Fig. 6(b). This step is described in Algorithm 4.

Algorithm 4. The Community Propagation Algorithm

```
Input: Graph $G = (V, E, W)$, Compressed graph $G^c = (V^c, E^c, W^c)$ and communities of $G^c : \mathscr{C}^c = \{C_1, C_2, \dots, C_k\}$.
Output: Communities of $G : \mathscr{C}$ .
1: Initialize communities of $G : \mathscr{C} = \mathscr{C}^c$;
2: for each community $C_i \in \mathscr{C}^c$
3: for each vertex $u \in C_i$
4: /* $IV(u)$: a set of including vertices for each vertex $u$ after graph compression */ 5: if $IV(u) > 1$
6: Update $C_i = C_i \bigcup IV(u)$.
7: end if
8: end for
9: end for
10: return Communities $\mathscr{C}$.
```

3.5. Time complexity analysis

In this section, the time complexity of the proposed CDEP algorithm is analyzed. As described above, the proposed CDEP includes four steps, graph compression, community seed determination, seed expansion, and community structure propagation. Given a social network with n vertices and m edges and a compressed graph with n_c vertices and m_c edges, the graph compression step needs to iteratively merge the vertices with a degree 1 or 2. Thus the time complexity of this step is $O((n+m)t)$, where t is the number of iterations. In the community seed determination step, the complexity is bounded by constructing a decreasing sequence using the sorting algorithm. Therefore, the time complexity of this step is $O(n_c \log n_c)$. Community seed expansion requires measuring the similarity between unlabeled vertices and each community. The complexity of this operation scales with the size of each community. Therefore, the time complexity for this step scales as $O\left(n_c\overline{C}k\right)$, where $\overline{C}$ is the average size of each community during the expansion process and k denotes the number of communities. Finally, the community propagation step costs $O(n_c)$ time to check whether the vertices in the compression graph include merged vertices. Therefore, the total time complexity of our overall algorithm scales as $O\left((n+m)t + n_c\overline{C}k\right)$.

4. Experimental analysis

In this section, we evaluate the performance of the proposed algorithm against several benchmark algorithms over a variety of social networks in terms of two important evaluation criteria: the modularity(Q) and the normalized mutual information (NMI).

4.1. Experimental settings

The descriptions of different data sets, evaluation criteria, comparison algorithms, and related experimental environment settings are given in this subsection.

4.1.1. Data sets

In order to illustrate the effectiveness and efficiency of the proposed algorithm, we use 14 social networks that have been widely adopted in the literature. A detailed description of these networks is shown in Table 1.

4.1.2. Evaluation criteria

Measuring the quality of detected communities for a social network is challenging since different measures may lead to different quality communities. The quality measures used differ according to whether the ground-truth communities for a specific social network are known or not. In this section, the normalized mutual information (NMI) will be used as an evaluation measure for the quality of the detected communities for datasets with known communities. For social networks without ground truth information, we employ the modularity to measure the quality of the community detection results.

• *Normalized Mutual Information (NMI).* This is one of the most popular community detection algorithm validation metrics. It estimates the quality of the community with respect to the given labels of the data. More formally, NMI can effectively measure the amount of statistical information shared by random variables representing the cluster assignments and the predefined label assignments of the objects. Consider a graph $G = (V, E)$ with n vertices (or nodes) and m edges, and suppose that $\mathscr{C} = \{C_1, C_2, \ldots, C_k\}$ and $\mathscr{P} = \{P_1, P_2, \ldots, P_{k'}\}$ represent the sets of the communities detected by the community detection algorithm and the ground-truth communities of the n vertices, respectively. k and k' are the numbers of communities $\mathscr{C}$ and $\mathscr{P}$, respectively; $n_{i,j}$ is the number of common vertices for community C_i and ground-truth community P_j; n_i^c is the number of vertices in community C_i ; and n_j^p is the number of vertices in community P_j. Then NMI is defined and computed according to the following formula [45]:

$$NMI = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k'} n_{i,j} log \frac{n \cdot n_{i,j}}{n_i^c \cdot n_j^p}}{\sqrt{\sum_{i=1}^{k} n_i^c \cdot log \frac{n_i^c}{n} \cdot \sum_{j=1}^{k'} n_j^p \cdot log \frac{n_j^p}{n}}}. \tag{10}$$

• *Modularity (Q)*. Modularity is currently used to measure the performance of community detection algorithms when the underlying community labels of networks are unknown. For a set of communities of a given graph $G = (V, E)$, the modularity is defined as [34]:

$$Q = \frac{1}{2m}\sum_{i,j \in V}\left(A_{ij} - \frac{d(i)d(j)}{2m}\right) \times \delta(l_i, l_j), \tag{11}$$

where Q represents the modularity, m is the number of edges in the network, and A is the adjacency matrix of the network. If vertices v_i and v_j are directly connected, $A_{ij} = 1$; otherwise $A_{ij} = 0$. l_i and l_j are the labels of the community to which v_i and v_j belong, respectively. If $l_i = l_j$, then $\delta(l_i, l_j) = 1$; otherwise, $\delta(l_i, l_j) = 0$.

Note that the higher the values of the above two measures, the better the community detection algorithm.

4.1.3. Comparison algorithms

To fully investigate the performance of the proposed community detection algorithm, the following state-of-the-art community detection algorithms, which cover most of the main categories of existing approaches, will be adopted for benchmarking purposes. Since the proposed algorithm is a global community detection method, it is compared with the representative global community methods in the experiment analysis. These algorithms include fast modularity maximization (FMM) [33], normalized spectral clustering (NSC) [42], nonnegative matrix factorization (NMF) [35], the label propagation algorithm (LPA) [36], and the fast unfolding communities (FUC) [9].

In the NSC and NMF algorithms, the k-means algorithm with random initialization cluster centers is used. The maximum number of iterations of the LPA algorithm is set as 200. For the FUC algorithm, we set the parameter s as the default value, i.e., $s = 1$, indicating recursive computation. The methods other than the NSC and NMF algorithms, can determine the number of communities automatically. In addition, according to whether the ground-truth communities for a specific social network are known, the main parameter of NSC and NMF, the number of communities, was set to the number of ground-truth communities or the number of communities detected by our method. The time complexity of these algorithms is analyzed as follows. For the FMM algorithm, a maximum of $n - 1$ join operations are necessary to construct the complete dendrogram; hence, its time complexity is $O(n^2)$. Because matrix decomposition is needed in the NMF and NSC algorithms, the time complexity of these algorithms is $O(n^3)$. Although the time complexity of the LPA is low, the running time of the LPA is closely related to the initialization conditions. The most time-consuming part of the FUC algorithm is the bottom-level community

Table 1
Summary of social networks.

Datasets	# Vertices	# Edges	# Communities
Karate [16]	34	78	2
Dolphin [29]	62	159	2
Football [16]	115	613	12
Polbooks [24]	105	441	3
Polblogs [16]	1,490	16,718	2
Email [18]	1,133	5,451	NA
PGP [31]	10,681	24,316	NA
CA_AstroPh [25]	18,772	396,160	NA
CA_CondMat [25]	23,133	186,936	NA
Email_Enron [25]	36,692	183,831	NA
soc_Epinions [37]	75,879	508,837	NA
Email_EuAll [25]	265,214	420,045	NA
com_Youtube [49]	1,134,890	2,987,624	NA
WikiTalk [49]	2,394,385	5,021,410	NA

division. The other part is the calculation of the change in the modularity during community merging. The total time complexity of FUC is only related to the edges. The proposed algorithm and the compared algorithms were all implemented in the MATLAB computing environment and all experiments were conducted on a workstation with an Intel Xeon CPU E5-2650@2.60 GHz and 128gb of RAM.

4.2. Results of the effectiveness analysis

In this section, we present the performance results of the effectiveness analysis obtained by the different community detection algorithms. The NMI index is used to measure the quality of communities for the real dataset with known communities. The corresponding results are shown in Table 2. Note that the numbers in boldface denote the largest values in the corresponding row. From this table, we observe that the CDEP algorithm outperformed all the competing methods. Especially, for the Karate network, the communities detected by the proposed algorithm are consistent with the real community structure. For those social networks without ground-truth communities, we use the modularity index to measure the quality of the community detection results. The numbers of communities on these 9 networks are determined as 9, 4, 3, 5, 9, 7, 5, 6 and 8, respectively. The comparative modularity results are shown in Table 3. In this table, if the running time of an algorithm is greater than 10 h, we terminate it and set the result as 'NA'. In terms of modularity, we obtain the following observations. The CDEP algorithm performs better and obtains optimal results on 6 networks, compared to the other algorithms. In addition, the FUC algorithm acquires the best results on the other 3 networks. That is, the modularity values of our partitions are lower than those obtained by the FUC algorithm on some networks. This can be expected because our method is not specifically designed to optimize the modularity as with the FUC algorithm. However, the modularity values obtained by our method are almost always better than those of the other algorithms. Therefore, we can conclude that the CEDP algorithm is an effective and competitive method for identifying community structures.

4.3. Results of the efficiency analysis

In order to investigate how well the method scales over networks as they increase in size, the time performance of different algorithms on these given networks is compared in Table 4. We note that if the scale of a data set is large, some algorithms cannot obtain the community results within an acceptable time. Therefore, in our experiment, if the running time of an algorithm is greater than 10 h, we terminate it and set the running time as 'NA'. This table shows that the CDEP algorithm outperforms the competing methods in terms of efficiency as well. We conclude that the CDEP algorithm is the only method that can process all large-scale networks in a reasonable time.

4.4. Performance analysis of graph compression

In order to show the effectiveness of the graph compression step, we report the compression ratio of the proposed algorithm. The compressed rate indicates how many vertices and edges are merged into the compressed graph. If $|V^c|$ and $|E^c|$ represent the number of vertices and edges for the compressed graph, respectively, then the compression ratio (CR) is calculated as

$$CR = \frac{1}{2}\left(\frac{|V| - |V^c|}{|V|} + \frac{|E| - |E^c|}{|E|}\right). \tag{12}$$

The higher the value of this index, the better the compression performance. The summary of social networks after the graph compression step is shown in Table 5. In addition, the results of the compression ratio for these networks without a known community structure are shown in Fig. 7. One important observation is that the compressing performance increases as the network scale increases. However, this relationship is not strictly monotonic.

In summary, the above experimental results show that the proposed algorithm not only obtains higher or comparable quality community detection results, but it also has high computational efficiency. The larger the variance of the degree of vertices in a network, the sparser the degree distribution of the vertices in the network. That is, a sparse network with low rich-club connectivity includes more lower degree vertices. First, for this kind of network, the graph compression step can reduce the network size efficiently by compressing vertices. Second, on the compressed graph, the initial community

Table 2
The NMI values of communities found by different algorithms.

Datasets	FMM	NSC	NMF	LPA	FUC	CDEP
Karate	0.7069	0.7329	0.8365	0.8372	0.7236	**1.0000**
Dolphin	0.5020	0.5270	0.4528	0.5290	0.5649	**0.5996**
Football	0.7569	0.8302	0.8368	0.8651	0.8581	**0.8691**
Polbooks	0.5314	0.4033	0.3256	0.4944	0.5311	**0.5436**
Polblogs	0.3262	0.1968	0.2402	0.3505	0.4378	**0.4403**

Table 3
The modularity (Q) values of communities found by different algorithms.

Datasets	FMM	NSC	NMF	LPA	FUC	CDEP
Email	0.5083	0.2669	0.2518	0.3833	**0.5502**	0.3837
PGP	0.8407	0.0928	NA	0.5961	**0.8474**	0.7865
CA_AstroPh	NA	NA	NA	0.3000	0.5038	**0.5326**
CA_CondMat	NA	NA	NA	0.5940	**0.5942**	0.4714
Email_Enron	NA	NA	NA	0.3195	NA	**0.4538**
soc_Epinions	NA	NA	NA	NA	NA	**0.2205**
Email_EuAll	NA	NA	NA	NA	NA	**0.5463**
com_Youtube	NA	NA	NA	NA	NA	**0.4753**
WikiTalk	NA	NA	NA	NA	NA	**0.3674**

Table 4
Computational time (seconds) of different algorithms.

Datasets	FMM	NSC	NMF	LPA	FUC	CDEP
Karate	0.0310	0.0190	0.0290	0.0380	0.0310	**0.0160**
Dolphin	0.0180	0.0110	0.0140	0.0760	0.0780	**0.0104**
Football	0.0300	0.0340	0.0370	0.0630	0.0780	**0.0251**
Polbooks	0.0170	0.0210	0.0180	0.0780	0.0620	**0.0152**
Polblogs	33.7280	17.7040	2.4540	44.4000	6.9730	**2.0460**
Email	18.2770	4.7200	1.9000	31.6803	1.9790	**1.6150**
PGP	18248.3210	31130.8600	NA	783.5310	1376.4520	**9.8080**
CA_AstroPh	NA	NA	NA	3024.1190	2239.7330	**33.3670**
CA_CondMat	NA	NA	NA	3907.8630	10321.9240	**33.2150**
Email_Enron	NA	NA	NA	7903.3200	NA	**67.5040**
soc_Epinions	NA	NA	NA	NA	NA	**247.7370**
Email_EuAll	NA	NA	NA	NA	NA	**3053.5550**
com_Youtube	NA	NA	NA	NA	NA	**4621.5726**
WikiTalk	NA	NA	NA	NA	NA	**5428.4692**

Table 5
Summary of social networks after graph compression.

Datasets	# Vertices	# Edges
Email	940	5,221
PGP	3,996	16,701
CA_AstroPh	15,813	193,730
CA_CondMat	17,824	85,745
Email_Enron	23,332	168,273
soc_Epinions	33,041	359,124
Email_EuAll	17,824	85,745
com_Youtube	350,441	2,020,611
WikiTalk	277,650	2,555,576

center discovery method can effectively detect the initial community structure, and then obtain the community results of the whole network.

5. Conclusion and future work

Community detection is a challenging problem, especially in social network analysis when the scale of the network is large. This paper aims to improve the community detection efficiency within social networks by compressing the network scale. The proposed algorithm includes four steps: graph compression, community seed determination, seed expansion, and community structure propagation. First, a compressed graph is obtained by iteratively merging vertices with a degree of 1 or 2 into their neighbors with a higher degree. Then, the number of communities and initial seeds of communities for the compressed graph are determined automatically by considering the density and quality of vertices. Finally, after obtaining the community structure of the compressed social network via seed expansion, the community results are propagated to the original social network. The advantages of the proposed algorithm are demonstrated on 14 real social networks. The experimental results show that the proposed algorithm has much better performance in extracting community structures than the other algorithms in terms of efficiency and effectiveness.

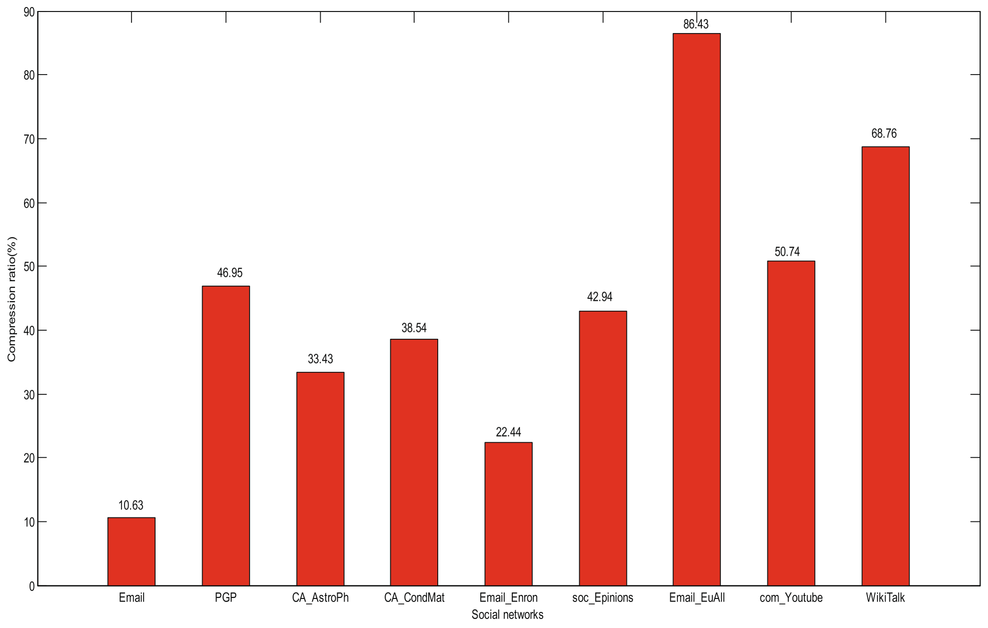

Fig. 7. Compression ratio of the social networks.

The proposed algorithm based on graph compression can improve the efficiency while maintaining the effectiveness for large-scale networks, but it is only suited for undirected networks. Therefore, how to extend the graph compression strategy to the community discovery of attribute networks [10] and multilayer networks [21] is the focus of future research.

CRediT authorship contribution statement

Xingwang Zhao: Conceptualization, Methodology, Software, Writing - original draft. **Jiye Liang:** Supervision, Methodology. **Jie Wang:** Writing - review & editing, Software, Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors are very grateful to the anonymous reviewers and editor. Their many helpful and constructive comments and suggestions helped us significantly improve this work. This work was supported by National Key Research and Development Program of China (No. 2020AAA0106100), National Natural Science Fund of China (Nos. 62072293, 61876103, 61976128, 61906111, 62006145), Key R & D projects of Shanxi Province (No. 201903D121162), Scientific and Technologial Innovation Programs of Higher Education Institutions in Shanxi (No. 2020L0245), and the 1331 Engineering Project of Shanxi Province, China.

References

[1] A. Abou-Rjeili, G. Karypis, Multilevel algorithms for partitioning power-law graphs, in: Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium, 2006, pp. 103–114.
[2] L.A. Adamic, E. Adar, Friends and neighbors on the web, Social Networks 25 (3) (2003) 211–230.
[3] L.A. Adamic, B.A. Huberman, A.L. Barabsi, R. Albert, H. Jeong, G. Bianconi, Power-law distribution of the world wide web, Science 287 (5461) (2000) 2115.
[4] L. Bai, X. Cheng, J. Liang, Y. Guo, Fast graph clustering with a new description model for community detection, Inf. Sci. 388 (2017) 37–47.
[5] L. Bai, J. Liang, H. Du, Y. Guo, A novel community detection algorithm based on simplification of complex networks, Knowl.-Based Syst. 143 (2018) 58–64.
[6] M.J. Barber, J.W. Clark, Detecting network communities by propagating labels under constraints, Phys. Rev. E Stat. Nonlinear Soft Matter Phys. 80 (2009) 026129.
[7] N. Barbieri, F. Bonchi, E. Galimberti, F. Gullo, Efficient and effective community search, Data Min. Knowl. Discovery 29 (5) (2015) 1406–1433.
[8] P. Bedi, C. Sharma, Community detection in social networks, Wiley Interdisc. Rev. Data Min. Knowl. Discovery 6 (3) (2016) 115–135.
[9] V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. 2008 (10) (2008) 155–168.

[10] Z. Bu, H. Li, J. Cao, Z. Wang, G. Gao, Dynamic cluster formation game for attributed graph clustering, IEEE Trans. Cybern. 49 (1) (2019) 328–341.
[11] Z. Bu, H. Li, C. Zhang, J. Cao, A. Li, Y. Shi, Graph k-means based on leader identification, dynamic game and opinion dynamics, IEEE Trans. Knowl. Data Eng. 32 (7) (2020) 1348–1361.
[12] J. Cao, Z. Bu, Y. Wang, H. Yang, J. Jiang, H. Li, Detecting prosumer-community groups in smart grids from the multiagent perspective, IEEE Trans. Syst. Man Cybern. Syst. 49 (8) (2019) 1652–1664.
[13] J. Chen, Y. Saad, Dense subgraph extraction with application to community detection, IEEE Trans. Knowl. Data Eng. 24 (7) (2012) 1216–1230.
[14] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (3–5) (2009) 75–174.
[15] S.E. Garza, S.E. Schaeffer, Community detection with the label propagation algorithm: a survey, Phys. A Stat. Mech. Appl. 534 (2019) 122058.
[16] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Nat. Acad. Sci. USA 99 (12) (2002) 7821–7826.
[17] A. Grover, J. Leskovec, Node2vec: scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864..
[18] R. Guimer, L. Danon, A. Daz-Guilera, F. Giralt, A. Arenas, Self-similar community structure in a network of human interactions, Phys. Rev. E 68 (6) (2003) 065103.
[19] M.S. Handcock, A.E. Raftery, J.M. Tantrum, Model-based clustering for social networks, J. Roy. Stat. Soc. Ser. A 127 (2) (2007) 301–354.
[20] W. Hu, Finding statistically significant communities in networks with weighted label propagation, Social Network. 2 (3) (2013) 138–146.
[21] R. Interdonato, M. Magnani, D. Perna, A. Tagarelli, D. Vega, Multilayer network simplification: approaches, models and methods, Comput. Sci. Rev. 36 (2020) 100246.
[22] R. Interdonato, A. Tagarelli, D. Ienco, A. Sallaberry, P. Poncelet, Local community detection in multilayer networks, Data Min. Knowl. Discovery 31 (5) (2017) 1444–1479.
[23] M.A. Javed, M.S. Younis, S. Latif, J. Qadir, A. Baig, Community detection in networks: a multidisciplinary review, J. Network Comput. Appl. 108 (2018) 87–111.
[24] V. Krebs, Books about us politics, http://www.orgnet.com/, 2004.
[25] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: densification and shrinking diameters, ACM Trans. Knowl. Discovery Data 1 (1) (2006) 2.
[26] C. Lin, On the convergence of multiplicative update algorithms for nonnegative matrix factorization, IEEE Trans. Neural Networks 18 (6) (2007) 1589–1596.
[27] Y. Liu, T. Safavi, A. Dighe, D. Koutra, Graph summarization methods and applications: a survey, ACM Comput. Surveys 51 (3) (2018), Article 62.
[28] W. Luo, N. Lu, L. Ni, W. Zhu, W. Ding, Local community detection by the nearest nodes with greater centrality, Inf. Sci. 517 (2020) 377–392.
[29] D. Lusseau, K. Schneider, O.J. Boisseau, The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations, Behav. Ecol. Sociobiol. 54 (4) (2003) 392–405.
[30] K. Macropol, A. Singh, Scalable discovery of best clusters on large graphs, Proc. VLDB Endowment 3 (1–2) (2010) 693–702.
[31] B. Marin, P.S. Romualdo, D.G. Albert, A. Alex, Models of social networks based on social distance attachment, Phys. Rev. E Stat. Nonlinear Soft Matter Phys. 70 (2) (2004) 056122.
[32] S. Moon, J.G. Lee, M. Kang, M. Choy, J.W. Lee, Parallel community detection on large graphs with mapreduce and graphchi, Data Knowl. Eng. 104 (2016) 17–31.
[33] M.E.J. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (6) (2004) 066133.
[34] M.E.J. Newman, Modularity and community structure in networks, Proc. Nat. Acad. Sci. USA 103 (23) (2006) 8577–8582..
[35] G. Nicolas, G. Franois, Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization, Neural Comput. 24 (4) (2012) 1085–1105.
[36] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E 76 (2) (2007) 036106.
[37] M. Richardson, R. Agrawal, P. Domingos, Trust management for the semantic web, Lect. Notes Comput. Sci. 284 (10) (2003) 351–368.
[38] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (2014) 1492–1496.
[39] V. Satuluri, S. Parthasarathy, Y. Ruan, Local graph sparsification for scalable clustering, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2011, pp. 721–732.
[40] S.E. Schaeffer, Graph clustering, Comput. Sci. Rev. 1 (1) (2007) 27–64.
[41] R. Sharma, S. Oliveira, Community detection algorithm for big social networks using hybrid architecture, Big Data Res. 10 (2017) 44–52.
[42] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[43] D.A. Spielmat, S. Teng, Spectral partitioning works: Planar graphs and finite elementmeshes, Technical report, USA, 1996.
[44] C.L. Staudt, H. Meyerhenke, Engineering parallel algorithms for community detection in massive networks, IEEE Trans. Parallel Distrib. Syst. 27 (1) (2015) 171–184.
[45] A. Strehl, J. Ghosh, Cluster ensembles-a knowledge reuse framework for combining multiple partitions, J. Mach. Learn. Res. 3 (2002) 583–617.
[46] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 1067–1077.
[47] J.J. Whang, D.F. Gleich, I.S. Dhillon, Overlapping community detection using neighborhood-inflated seed expansion, IEEE Trans. Knowl. Data Eng. 28 (5) (2016) 1272–1284.
[48] W. Wu, S. Kwong, Y. Zhou, Y. Jia, W. Gao, Nonnegative matrix factorization with mixed hypergraph regularization for community detection, Inf. Sci. 435 (2018) 263–281.
[49] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, in: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics, 2012, pp. 1–8.