Improving NMF-based community discovery using distributed robust nonnegative matrix factorization with SimRank similarity measure

He, C., Fei, X., Li, H., Tang, Y., Liu, H. & Liu, S.

DOI 10.1007/s11227-018-2500-9
ISSN 0920-8542
ESSN 1573-0484
Publisher: Springer

Improving NMF-based community discovery using distributed robust nonnegative matrix factorization with SimRank similarity measure

Chaobo He · Xiang Fei · Hanchao Li · Yong Tang · Hai Liu · Shuangyin Liu

Received: 12 May 2018 / Accepted: date

Abstract Nonnegative Matrix Factorization (NMF) has become a powerful model for community discovery in complex networks. Existing NMF-based methods for community discovery often factorize the corresponding adjacent matrix of complex networks to obtain its community indicator matrix. However, the adjacent matrix cannot represent the global structure feature of complex networks very well, and this leads to the performance degradation of community discovery. Besides, most of existing methods are not robust and scalable enough, so they are not effective to deal with complex networks with noises and large-scales. Aiming at these problems above, in this paper we propose a method for community discovery using distributed robust NMF with SimRank similarity measure. This method selects SimRank measure to

✉ Chaobo He
School of Information Science and Technology, ZhongKai University of Agriculture and Engineering, Guangzhou, China
E-mail: hechaobo@foxmail.com

Xiang Fei
Department of Computing, Coventry University, Coventry, UK
E-mail: aa5861@coventry.ac.uk

Hanchao Li
Department of Computing, Coventry University, Coventry, UK
E-mail: lih30@uni.coventry.ac.uk

Yong Tang
School of Computer Science, South China Normal University, Guangzhou, China
E-mail: ytang@m.scnu.edu.cn

Hai Liu
School of Computer Science, South China Normal University, Guangzhou, China
E-mail: liuhai@scnu.edu.cn

Shuangyin Liu
School of Information Science and Technology, ZhongKai University of Agriculture and Engineering, Guangzhou, China
E-mail: shuangyinliu@126.com

construct the feature matrix, which can more accurately represent the global structure feature of complex networks. To improve the robustness, we select $\ell_{2,1}$ norm instead of the widely used Frobenius norm to construct its NMF-based community discovery model. In addition, to improve the scalability, we implement its key components by using MapReduce distributed computing framework, including computing SimRank feature matrix and iteratively solving the NMF-based model for community discovery. We conduct extensive experiments on several typical complex networks. The results show that our method has better performance and robustness than other representative NMF-based methods for community discovery. Moreover, our method presents good scalability, and hence can be used to discover communities in the large-scale complex networks.

Keywords Community discovery · Robust nonnegative matrix factorization · SimRank · MapReduce · Complex networks

1 Introduction

Various real world complex networks, such as online social networks and co-authorship networks, not only have small-world and scale-free properties, but also generally have significant community structures. Members in the same community connect to each other more densely than those in different communities [1, 2]. Discovering high quality communities is not only very useful for the analysis of the structures and functions of complex networks, but also of great value in practical applications, such as similar user groups detection and social markets in online social networks. Recently, lots of methods have been proposed to deal with the problem of community discovery in complex networks, including graph partitioning based methods [3–5], modularity optimization based methods [6–8] and statistical inference based methods [9, 10]. It is worth pointing out that Nonnegative Matrix Factorization (NMF) based method for community discovery has also gained great attentions. Existing NMF-based methods for community discovery have proven that NMF can effectively capture the underlying community structure of complex networks in the low dimensional data space and has good interpretability of the community discovery results. Moreover, due to the good extensibility of NMF, we can easily extend the basic NMF model to resolve various problems of community discovery, such as overlapped community discovery using bounded NMF [11], community discovery across multiple heterogeneous networks using joint N-MF [12–14] and constraint community discovery using semi-supervised NMF [15, 16]. Overall, these features above have helped NMF become an ideal and powerful model for community discovery.

Although many NMF-based methods for community discovery have been proposed, we find that most of them still have the following common shortcomings:

– They often select the simple adjacent matrix of complex networks as the feature matrix used for factorization. However, the adjacent matrix only represents the local structure feature and cannot represent the global structure feature of complex networks very well. Obviously, if the feature matrix cannot provide enough accurate information, it will degrade the performance of community discovery based on NMF.

- They often use the Frobenius norm to devise the objective function of NMF-based model for community discovery, but this least square error function is not robust with respect to noises. Actually, many real world complex networks contain lots of noises, such as lots of casual user links existing in online social networks. If the objective function is not robust enough to deal with these noises, it will also degrade the performance of community discovery.
- They all have high time complexities. For a complex network with n nodes, their time complexities are all above $O(n^2)$. When facing the large-scale complex networks, these methods are hard to run efficiently on a single machine due to limited computing and storage resources. Therefore, to improve their scalabilities, it is imperative to have them run on the distributed computing environment.

These findings above motivate us to seek for a better measure to construct the feature matrix of complex networks, which can further improve the performance of community discovery based on NMF. Meanwhile, we also devote ourselves to improving the robustness and scalability of NMF-based method for community discovery. This paper is an extended version of a paper that first appeared in [17], where a method for community discovery using distributed SimRank NMF is proposed. In this paper, we further improve this method by employing robust distributed NMF. In particular, we add extensive theoretical and experimental analyses of robustness. Our main works are summarized as follows:

- We investigate several widely used similarity measures of graph nodes and select SimRank measure to construct the feature matrix, which can better represent the global structure feature of complex networks.
- We propose a new method for community discovery using distributed robust NMF with SimRank similarity measure. This method devises the objective function of the NMF-based model for community discovery using $\ell_{2,1}$ norm, which can help to improve its robustness. In particular, to improve the scalability of our method, we implement its key components based on the MapReduce distributed computing framework, including computing SimRank feature matrix and iteratively solving the NMF-based model for community discovery.
- We conduct extensive experiments on several typical complex network datasets. The results show that our method can obtain better performance than the representative NMF-based method with different similarity measures and is effective to handle noises with enhanced robustness. Moreover, our method presents good scalability by running on the MapReduce cluster and is efficient enough to deal with the problem of community discovery in the large-scale complex networks.

The rest of this paper is organized as follows. Firstly, a brief review of related work is given in Section 2. Then, Section 3 elaborates the key parts of our proposed method. The experiments and analysis are reported in Section 4. Finally, the conclusions are drawn in Section 5.

2 Related work

2.1 SimRank similarity measure

Without loss of generality, the complex network can be formally denoted as an undirected and unweighted graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes and $E = \{e_{ij} | v_i \in V \wedge v_j \in V\}$ is the set of all edges between nodes. $n = |V|$ is the number of nodes. We can use the adjacent matrix $A = [a_{ij}]^{n\times n}$ to represent G. For $\forall a_{ij} \in A$, if $e_{ij} \in E$, then $a_{ij} = 1$, else $a_{ij} = 0$. Let $S \in \mathbb{R}_{+}^{n\times n}$ denote the node similarity matrix of G and $\forall s_{ij} \in S$ represents the similarity between v_i and v_j, then S is suitable to serve as the feature matrix of the complex network used in NMF-based model for community discovery. Recently, there have been several representative node similarity measures used for constructing S, including common neighbors using Jaccard index (CN) [18], regular equivalence (RE) [18], diffusion kernels (DK) [19], random walk (RW) [20] and SimRank (SR) [21]. In particular, SimRank is based on the concept that two nodes are similar if they are referenced by similar nodes and has become a powerful measure for the similarity of pairs of nodes in a graph. The SimRank similarity s_{ij} between v_i and v_j is defined as (i) $s_{ij} = 1$, if $i = j$; (ii) $s_{ij} = 0$, if $N(i) = \emptyset$ or $N(j) = \emptyset$; (iii) otherwise,

$$s_{ij} = \frac{\rho}{|N(i)||N(j)|} \sum_{a\in N(i)} \sum_{b\in N(j)} s_{ab}, \tag{1}$$

where $N(i)$ and $N(j)$ denote the set of neighbors of v_i and v_j, respectively. $\rho \in [0, 1]$ is a damping factor (we set $\rho = 0.6$ in this paper). Based on SimRank definition above we can obtain $s_{ii} = 1$ and $s_{ij} = s_{ji}$. Hence the SimRank similarity matrix S is reflexive and symmetric. Obviously, Equation 1 is recursive and can be rewritten in the following iterative form:

$$s_{ij}^{t} = \frac{\rho}{|N(i)||N(j)|} \sum_{a\in N(i)} \sum_{b\in N(j)} s_{ab}^{t-1}. \tag{2}$$

Starting with $s_{aa}^0 = 1$ and $s_{ab}^0 = 0$ if $a \neq b$, Equation 2 will converge to the exact solution of Equation 1 when $t \to \infty$. Then, the converged s_{ij} is the final similarity between v_i and v_j. By this iterative computation, SimRank can compute the similarity of pairs of nodes in a graph by combining both local and global topology feature. Hence, it can provide a more accurate feature matrix to the complex network. In this paper we select SimRank similarity matrix S to serve as the feature matrix used for our NMF-based method for community discovery.

2.2 NMF-based method for community discovery

NMF is originally presented for image processing [22,23] and later on it is widely used in a variety of data mining tasks, including document clustering [24,25], multimedia data analysis [26–28] and collaborative filtering [29]. In [30], Ding *et al.* successfully proved the equivalence of NMF and spectral clustering, which is closely related to the node clustering in a graph. This arouses many researches on NMF-based methods for community discovery in complex networks. Generally, the basic NMF-based model for community discovery can be formulated as: given the feature matrix $X \in \mathbb{R}_{+}^{n\times n}$ of the complex network, then X can be decomposed into the community feature matrix W and the community indicator matrix H, such that $X \approx WH^T$, where $W \in \mathbb{R}_{+}^{n\times k}$ and $H \in \mathbb{R}_{+}^{n\times k}$. k is the desired number of communities and should satisfy $k \ll \min(m,n)$. W and H can be obtained by minimizing the objective function: $J(X, WH^T)$, where J is the objective function that measures the error between X and WH^T. The most widely used error measure function is Frobenius norm and its corresponding basic NMF-based model for community discovery is:

$$\begin{aligned} \min J(X, WH^T) = \min \parallel X - WH^T \parallel_F^2 . \\ s.t. \quad W \geq 0, H \geq 0 \end{aligned} \tag{3}$$

In practice, existing NMF-based methods for community discovery often extend this model to resolve various problems of community discovery. For example, targeting undirected, directed and compound complex networks, Wang *et al.* [31] proposed symmetric NMF (SNMF), asymmetric NMF (ANMF) and joint NMF (JNMF) to discover communities, respectively. Related experiments conducted on synthetic and real world complex networks shown their effectiveness. Aiming at identifying overlapped communities, Nguyen *et al.* [32] introduced two approaches, namely iSNMF and iANMF, which can extract meaningful overlapping communities via soft community assignments produced by NMF. Zhang *et al.* [33] developed a symmetric binary matrix factorization model (SBNMF) to identify overlapping communities. This model allows us not only to assign community memberships explicitly to nodes, but also to distinguish outliers from overlapping nodes. Besides these classical methods, many new variants are also continuously proposed, such as MHGNMF method using mixed hypergraph regularization [34], ENMF method using evolutionary modularity density [35], PSSNMF method using the node popularity [36] and BSNMF method [37] using Bayesian inference.

Generally speaking, existing NMF-based methods for community discovery are effective, but they nearly have the same drawbacks. Firstly, they often construct the feature matrix X by using the adjacent matrix A, which is not accurate to represent the global feature of the complex network. This often leads to the degraded performance of community discovery. Secondly, they often devise the objective function by using the Frobenius norm. It is not robust enough, because a few noises with large errors easily dominate the objection function by the form of the squared errors shown in Equation 3. Finally, their solutions to the NMF-based model for community discovery all refer to the multiplication operations of multiple $n \times n$ matrices, so their time complexities are all above $O(n^2)$. Although we can use sparse matrix multiplication or other optimization strategies to further reduce the time complexity, the computational cost is still heavy when encountering the large-scale complex networks. To resolve these problems above, in this paper we propose a new method using distributed robust NMF with SimRank similarity measure.

3 Our method

In this section, we first present the community discovery model based on robust NMF with SimRank similarity measure, including its solution and algorithm, and then give the detailed convergence proof of the solution. Finally, we present how to improve its scalability by using MapReduce distributed computing framework.

3.1 Community discovery model

Owing to the feature matrix S being symmetric, following the idea about symmetric NMF in [38] we can approximate S by the form of HH^T. Namely, $S \approx HH^T$, where $H \in \mathbb{R}_+^{n\times k}$ denotes the community indicator matrix and each entry $h_{ij} \in H$ represents the strength that the ith node belongs to the jth community. The bigger the value of h_{ij}, the higher degree of membership with which the ith node belongs to the jth community. To improve the robustness of our method, we select $\ell_{2,1}$ norm to measure the error between S and HH^T instead of the widely used Frobenius norm. Its corresponding community discovery model is formulated as:

$$\min J(H) = \frac{1}{2}||S - HH^T||_{2,1}, \quad s.t. \quad H \geq 0 \tag{4}$$

In this model, the error between S and HH^T is $\sum_{i=1}^{n} ||s_i - Hh_i^T||$, where s_i and h_i^T represent the ith column vectors of S and H^T, respectively. This formulation doesn't square the error, and thus the large errors due to noises do not dominate the objective function $J(H)$. The minimization of the multivariable objective function $J(H)$ above can be transformed into the typical constrained optimization problem, which can be solved using the gradient descent method.

Firstly, we can rewrite the objective function $J(H)$ as:

$$\begin{aligned}
J(H) &= \frac{1}{2}\sum_{i=1}^{n} ||s_i - Hh_i^T||^2 \frac{1}{||s_i - Hh_i^T||} \\
&= \frac{1}{2}\sum_{i=1}^{n}((S - HH^T)^T(S - HH^T))_{ii}d_{ii} \\
&= \frac{1}{2}tr((S - HH^T)D(S - HH^T)^T) \\
&= \frac{1}{2}(tr(SDS) - 2tr(HH^TDS) + tr(HH^TDHH^T)),
\end{aligned} \tag{5}$$

where $tr(\cdot)$ denotes the trace of a matrix and $D = [d_{ii}]^{n\times n}$ is a diagonal matrix with the diagonal elements given by

$$d_{ii} = \frac{1}{||s_i - Hh_i^T||} \tag{6}$$

Then, we can compute the gradient of $J(H)$ with respect to h_{ij}:

$$\begin{aligned}
\nabla_{h_{ij}} J(H) &= \frac{1}{2}(\frac{\partial tr(SDS)}{\partial h_{ij}} - \frac{2\partial tr(HH^TDS)}{\partial h_{ij}} \\
&+ \frac{\partial tr(HH^TDHH^T)}{\partial h_{ij}}) \\
&= (-DSH - SDH + 2DHH^TH)_{ij}.
\end{aligned} \tag{7}$$

Finally, using the gradient descent method, we have

$$h_{ij} \leftarrow h_{ij} - \epsilon_{ij}\nabla_{h_{ij}} J(H). \tag{8}$$

Setting $\epsilon_{ij} = \frac{h_{ij}}{(6DHH^TH)_{ij}}$, we can obtain the following iterative update rule for h_{ij}:

$$h_{ij} \leftarrow \frac{2}{3}h_{ij}(1 + \frac{(DSH + SDH)_{ij}}{4(DHH^TH)_{ij}}). \tag{9}$$

The objective function $J(H)$ can monotonically decrease and converge under the iterative update rule in Equation 9. Its convergence proof is given in Section 3.2. The community membership of every node in G can be determined based on converged H. Let $C = \{c_1, c_2, ..., c_k\}$ denote the set of communities and T_1 denote the number of iterations, we devise the community discovery algorithm shown in Algorithm 1.

In Algorithm 1, there are two parts which are time consuming. One is constructing S using SimRank measure (line 1). It still requires $O(dn^2)$ time per iteration based on the state-of-the-art SimRank computation work presented in [39], where d is the number of average degrees in G. The other time-consuming part is updating H iteratively (line 4). Updating H per iteration mainly refers to matrix multiplication operations and it requires $O(kn^2)$ time per iteration.

Algorithm 1 Community discovery algorithm

```
Input: $G = (V, E), k, T_1$;
Output: $C = \{c_1, c_2, ..., c_k\}$;
1: Construct $S$ using SimRank measure;
2: Initialize $H$ with random nonnegative values;
3: for $t = 1$ to $T_1$ do
4:     $\forall h_{ij} \in H$, $h_{ij} \leftarrow \frac{2}{3} h_{ij}(1 + \frac{(DSH+SDH)_{ij}}{4(DHH^T H)_{ij}})$;
5: end for
6: $\forall i = 1...k$, $c_i \leftarrow \emptyset$;
7: for $\forall v_i \in V$ do
8:     $p = \underset{l}{\operatorname{argmax}}\, h_{il}$;
9:     $c_p = c_p \bigcup \{v_i\}$;
10: end for
```

It is not hard to conclude that the total time complexity of Algorithm 1 is beyond $O(n^2)$. Therefore, Algorithm 1 running on serial mode is hard to deal with the large-scale complex networks. In Section 3.3, we will introduce how to compute SimRank and update H using MapReduce distributed computing framework, which can improve the scalability of Algorithm 1 remarkably under the distributed computing environment.

3.2 Convergence proof

To prove the convergence of the iterative update rule of H in Equation 9, we will make use of the auxiliary function method proposed in [23]. Firstly, we introduce the basic definitions and lemmas about the auxiliary function.

Definition 1. If a function satisfies

$$L(h, h') \geq F(h), \quad L(h, h) = F(h) \tag{10}$$

we say $L(h, h')$ is an auxiliary function of $F(h)$.

Lemma 1. If $L(h, h')$ is an auxiliary function, then $F(h)$ monotonically decreases under the update

$$h^{t+1} = \arg\min_h L(h, h^t) \tag{11}$$

Proof: $F(h^{t+1}) \leq L(h^{t+1}, h^t) \leq L(h^t, h^t) = F(h^t)$

Note that $F(h^{t+1}) = F(h^t)$ only if h^t is a local minimum of $L(h, h^t)$. Thus, by iterating the update in Equation 11 $F(h)$ can converge to a local minimum $h_{min} = \arg\min_h F(h)$. Namely, we can obtain a sequence of estimates: $F(h_{min}) \leq ... \leq F(h^{t+1}) \leq F(h^t) \leq ... \leq F(h^1) \leq F(h^0)$. This implies us that if we can define an appropriate auxiliary function for $J(H)$ and the update rule in Equation 9 also follows Equation 11, then the update rule in Equation

9 can guarantee the convergence of $J(H)$. Next we will present the detailed proofs.

For convenience, let function $J(h)$ denote the part related to h_{ij} in $J(H)$, then we can compute the first order, second order, third order and fourth order derivatives of $J(h)$ with respect to h_{ij}, respectively:

$$J'(h) = (-DSH - SDH + 2DHH^T H)_{ij} \tag{12}$$

$$\begin{aligned} J''(h) = &-(DS + SD)_{ii} + 2((DHH^T)_{ii} \\ &+ (DH)_{ij} h_{ij} + d_{ii} \sum_p h_{pj}^2) \end{aligned} \tag{13}$$

$$J'''(h) = 12 d_{ii} h_{ij} \tag{14}$$

$$J^{(4)}(h) = 12 d_{ii} \tag{15}$$

Let h_{ij}^t denote the tth iterative update value of h_{ij}, then the Taylor series expansion of $J(h)$ in h_{ij}^t can be written as:

$$\begin{aligned} J(h) = &J(h_{ij}^t) + J'(h_{ij}^t)(h - h_{ij}^t) \\ &+ \frac{1}{2} J''(h_{ij}^t)(h - h_{ij}^t)^2 + \frac{1}{6} J'''(h_{ij}^t)(h - h_{ij}^t)^3 \\ &+ \frac{1}{24} J^{(4)}(h_{ij}^t)(h - h_{ij}^t)^4 \end{aligned} \tag{16}$$

We define the following function $L(h, h_{ij}^t)$ as the auxiliary function of $J(h)$:

$$\begin{aligned} L(h, h_{ij}^t) = &J(h_{ij}^t) + J'(h_{ij}^t)(h - h_{ij}^t) \\ &+ \frac{1}{2} [\frac{6(DHH^T H)_{ij}}{h_{ij}^t}](h - h_{ij}^t)^2 + \frac{1}{6} J'''(h_{ij}^t)(h - h_{ij}^t)^3 \\ &+ \frac{1}{24} J^{(4)}(h_{ij}^t)(h - h_{ij}^t)^4 \end{aligned} \tag{17}$$

Proof. Because $L(h, h) = J(h)$ is obvious, we only need to demonstrate that $L(h, h_{ij}^t) \geq J(h)$. Comparing the Taylor series expansion of $J(h)$ in Equation 16 with $L(h, h_{ij}^t)$ in Equation 17, we can find that $L(h, h_{ij}^t) \geq J(h)$ is actually equivalent to

$$\frac{6(DHH^T H)_{ij}}{h_{ij}^t} \geq J''(h) \tag{18}$$

Owing to H and D being nonnegative, it is easy to prove that

$$\frac{(DHH^TH)_{ij}}{h_{ij}^t} = \frac{\sum_p (DHH^T)_{ip}h_{pj}}{h_{ij}^t} \geq (DHH^T)_{ii} \tag{19}$$

$$\frac{(DHH^TH)_{ij}}{h_{ij}^t} = \frac{\sum_p \sum_q (DH)_{iq}h_{pq}h_{pj}}{h_{ij}^t} \geq (DH)_{ij}h_{ij} \tag{20}$$

$$\frac{(DHH^TH)_{ij}}{h_{ij}^t} = \frac{\sum_p \sum_q \sum_l d_{il}h_{lq}h_{pq}h_{pj}}{h_{ij}^t} \geq d_{ii}\sum_p h_{pj}^2 \tag{21}$$

$$\begin{aligned}\frac{6(DHH^TH)_{ij}}{h_{ij}^t} &\geq 2((DHH^T)_{ii} + (DH)_{ij}h_{ij} + d_{ii}\sum_p h_{pj}^2) \\ &\geq J''(h)\end{aligned} \tag{22}$$

So Equation 18 holds and $L(h, h_{ij}^t)$ can be the auxiliary function of $J(h)$. Next, we just need to prove the update rule in Equation 9 can guarantee $L(h, h_{ij}^t)$ converges to a local minimum. The derivatives of $L(h, h_{ij}^t)$ exist and are continuous in a small neighborhood of h_{ij}^t, so its local minimum can be obtained when $L'(h, h_{ij}^t) = 0$, which can be solved using Newton's Method:

$$h_{ij}^{t+1} = h_{ij}^t - \frac{L'(h_{ij}^t, h_{ij}^t)}{L''(h_{ij}^t, h_{ij}^t)} \tag{23}$$

From Equation 17, we can compute $L'(h_{ij}^t, h_{ij}^t)$ and $L''(h_{ij}^t, h_{ij}^t)$, respectively:

$$L'(h_{ij}^t, h_{ij}^t) = (-DSH - SDH + 2DHH^TH)_{ij} \tag{24}$$

$$L''(h_{ij}^t, h_{ij}^t) = \frac{6(DHH^TH)_{ij}}{h_{ij}^t} \tag{25}$$

Substituting $L'(h_{ij}^t, h_{ij}^t)$ and $L''(h_{ij}^t, h_{ij}^t)$ in Equation 23, we can obtain the following update rule:

$$h_{ij}^{t+1} = \frac{2}{3}h_{ij}^t(1 + \frac{(DSH + SDH)_{ij}}{4(DHH^TH)_{ij}}) \tag{26}$$

which is the same as the update rule showed in Equation 9. Therefore, the update rule of H in Equation 9 can guarantee the convergence of the $J(H)$.

3.3 Improving the scalability using MapReduce

3.3.1 Computing SimRank using MapReduce

From Equation 2, we can see that the similarity between any two nodes depends on similarities between their adjacent nodes. In each iteration the similarity can continuously propagate from a pair of nodes to another pair until it reaches the convergence state. In order to explain this propagation process more intuitively, we can derive a directed node-pair graph $G^2 = (V^2, E^2)$, where $(v_a, v_b) \in V^2$, if $v_a \in V$ and $v_b \in V$; $((v_a, v_b), (v_c, v_d)) \in E^2$, if $e_{ac} \in E$ and $e_{bd} \in E$. To simplify G^2, isolated nodes in G^2 can be deleted, because their similarities are always 0 in each iteration. Also, edges pointing to nodes (v_a, v_a) can be deleted, because the similarity of node (v_a, v_a) is always 1 in each iteration. In G^2, the similarity of any node can be propagated to any reachable nodes along the directed edges. Taking Figure 1 as an example of G and G^2, assuming that its SimRank computing begins at $(3, 3)$, $(3, 3)$ propagates its initial value 1 to $(2, 4)$ in the first iteration, then $(2, 4)$ computes its value based on Equation 2. In the second iteration, $(2, 4)$ propagates again its value to $(1, 2)$, $(1, 3)$, $(2, 3)$ and $(3, 4)$ to compute their values. Subsequent iterations will perform the same operations until every node in G^2 can finally obtain a stable value.

Through the analysis above, every iteration of SimRank is suitable to be implemented using one MapReduce job: Map phase transmits the similarity of every node-pair to their adjacent node-pairs and Reduce phase aggregates outputs from Maps to compute the similarity of every node-pair. The whole iterative process of SimRank computing is composed of many continuous jobs. Every job performs the same operations and the output of the previous job is used as the input of the next job. Let S^0 and S^* denote respectively the initial and the final SimRank values sets, and T_2 denote the number of iterations, we devise the pseudo-code of computing SimRank using MapReduce shown in Algorithm 2.

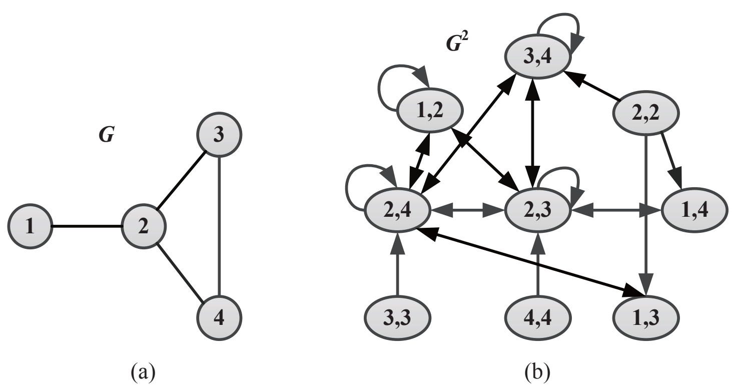

Fig. 1 An example of G and G^2

3.3.2 Updating H using MapReduce

Updating H using Equation 9 mainly involves matrix multiplication operations between S, D and H, which are all suitable to be implemented using

Algorithm 2 Computing SimRank using MapReduce

```
Input: $G = (V, E), S^0, T_2, \rho$;
Output: $S^*$;
 1: for $t = 1$ to $T_2$ do
 2:    Map: Read $< key = (v_a, v_b), value = s_{ab}^{t-1} >$;
 3:    for $\forall (v_c, v_d) \in \{(v_c, v_d) | v_c \in N(a) \wedge v_d \in N(b)\}$ do
 4:       Emit $< key = (v_c, v_d), value = s_{ab}^{t-1} >$;
 5:    end for
 6:    Reduce: Read $< key = (v_c, v_d), value = vs[0...] >$; //$vs$ is a list of $s_{ab}$ elements with the same key
 7:    if $c = d$ then $s_{cd}^t = 1$;
 8:    else
 9:       $s_{cd}^t = \frac{\rho}{len(vs)} sum(vs)$;
10:    end if
11:    Emit $< key = (v_c, v_d), value = s_{cd}^t >$;
12: end for
13: $S^* = S^t$;
14: Emit $S^*$;
```

MapReduce. For a large-scale complex network, its corresponding SimRank matrix S is actually very sparse, so S can be naturally stored in the format of $< (i, j), s_{ij} >$ key-value pairs. Also, the diagonal matrix D can be stored in the format of $< i, d_{ii} >$. For H, we can first partition it along the short dimension and this partition will reconstruct H as $(h_1, h_2, ..., h_n)$, where h_i is a k-dimension row vector of H. Then, H can also be stored in the format of $< i, h_i >$ key-value pairs. To improve the efficiency, D and H can both directly stored in the distributed cache of machine clusters and support global fast access. Considering that we must first compute D before computing H, updating H using Equation 9 can be divided into two phases. The first phase is computing D and the second phase is computing H. D is a diagonal matrix and we only need to compute its diagonal elements by using Equation 6, which can be implemented using the following two sets of MapReduce operations.

– Map-I: map $< j, (i, s_{ji}) >$ and $< j, (i, h_j, h_i) >$ on j such that tuples with the same j are shuffled to the same machine in the form of $< j, \{(i, s_{ji}), (i, h_j, h_i)\} >$.
– Reduce-I: read $< j, \{(i, s_{ji}), (i, h_j, h_i)\} >$, emit $< i, t_j >$, where t_j is a temporary variable and $t_j = (s_{ji} - h_j \bullet h_i)^2$.
– Map-II: map $< i, t_j >$ such that tuples with the same i are shuffled to the same machine in the form of $< i, \{t_j\} >$.
– Reduce-II: read $< i, \{t_j\} >$, emit $< i, d_{ii} >$, where $d_{ii} = \frac{1}{\sqrt{\sum_{j=1}^n t_j}}$.

For maximizing parallelism on MapReduce cluster, we divide the phase of computing H into 3 components: $P_1 = DSH + SDH$, $P_2 = 4DHH^TH$ and $H \leftarrow \frac{2}{3}H. * (1 + P_1./P_2)$, where every component can be implemented using MapReduce. The computing process of every component is described as follows.

(1) Computing $P_1 = DSH + SDH$. Let p_{1i} denote the ith row vector of P_1, then we have $p_{1i} = \sum_{j=1}^{n}(d_{ii}s_{ij}h_j + s_{ij}d_{jj}h_j)$, which can be implemented by two sets of MapReduce operations.

– Map-III: map $< j, (i, s_{ij}, d_{ii}, d_{jj}) >$ and $< j, h_j >$ on j such that tuples with the same j are shuffled to the same machine in the form of $< j, \{h_j, (i, s_{ij}, d_{ii}, d_{jj})\} >$.
– Reduce-III: read $< j, \{h_j, (i, s_{ij}, d_{ii}, d_{jj})\} >$, emit $< i, t_j >$, where $t_j = d_{ii}s_{ij}h_j + s_{ij}d_{jj}h_j$.
– Map-IV: map $< i, t_j >$ such that tuples with the same i are shuffled to the same machine in the form of $< i, \{t_j\} >$.
– Reduce-IV: read $< i, \{t_j\} >$, emit $< i, p_{1i} >$, where $p_{1i} = \sum_{j=1}^{n} t_j$.

(2) Computing $P_2 = 4DHH^TH$. DHH^T is a $n \times n$ matrix and H^TH is a $k \times k$ matrix. Therefore, to reduce the amount of intermediate data, we should first compute $T^{'} = H^TH$ and then compute $P_2 = DHT^{'}$. Owing to $T^{'} = \sum_{i=1}^{n} h_i^T h_i$, its computation can be implemented by single MapReduce operation.

– Map-V: map $< i, h_i >$ and emit $< 0, h_i^T h_i >$, where 0 is a dummy key value for data shuffling.
– Reduce-V: read $< 0, \{h_i^T h_i\} >$, compute $T^{'} = \sum_{i=1}^{n} h_i^T h_i$ and finally emit $< i, T^{'} >$.

$T^{'}$ is a $k \times k$ small matrix and hence can be directly stored in the distributed cache of machine clusters, which every computing node can access to. After that, we can also use a single MapReduce operation to compute $P_2 = 4DHT^{'}$:

– Map-VI: map $< i, h_i >$ and emit $< i, d_{ii}h_iT^{'} >$.
– Reduce-VI: read $< i, d_{ii}h_iT^{'} >$, emit $< i, p_{2i} >$, where $p_{2i} = 4d_{ii}h_iT^{'}$.

(3) Computing $H \leftarrow \frac{2}{3}H. * (1 + P_1./P_2)$. Updating $h_i \in H$ can be parallelized through the following single MapReduce operation.

– Map-VII: read $< i, h_i >$, $< i, p_{1i} >$ and $< i, p_{2i} >$ on i such that tuples with the same i are shuffled to the same machine in the form of $< i, \{h_i, p_{1i}, p_{2i}\} >$.
– Reduce-VII: read $< i, \{h_i, p_{1i}, p_{2i}\} >$ and emit $< i, h_i^{new} >$, where $h_i^{new} = \frac{2}{3}h_i. * (1 + p_{1i}./p_{2i})$.

Updating H on MapReduce cluster comprises of seven continuous MapReduce jobs above and its entire flowchart is depicted in Figure 2.

4 Experimental study

In this section, we will report the results of our experimental study on our proposed method (For convenience, we call it DRNMFSR hereafter). We conduct four types of experiments. The first one is the performance comparisons,

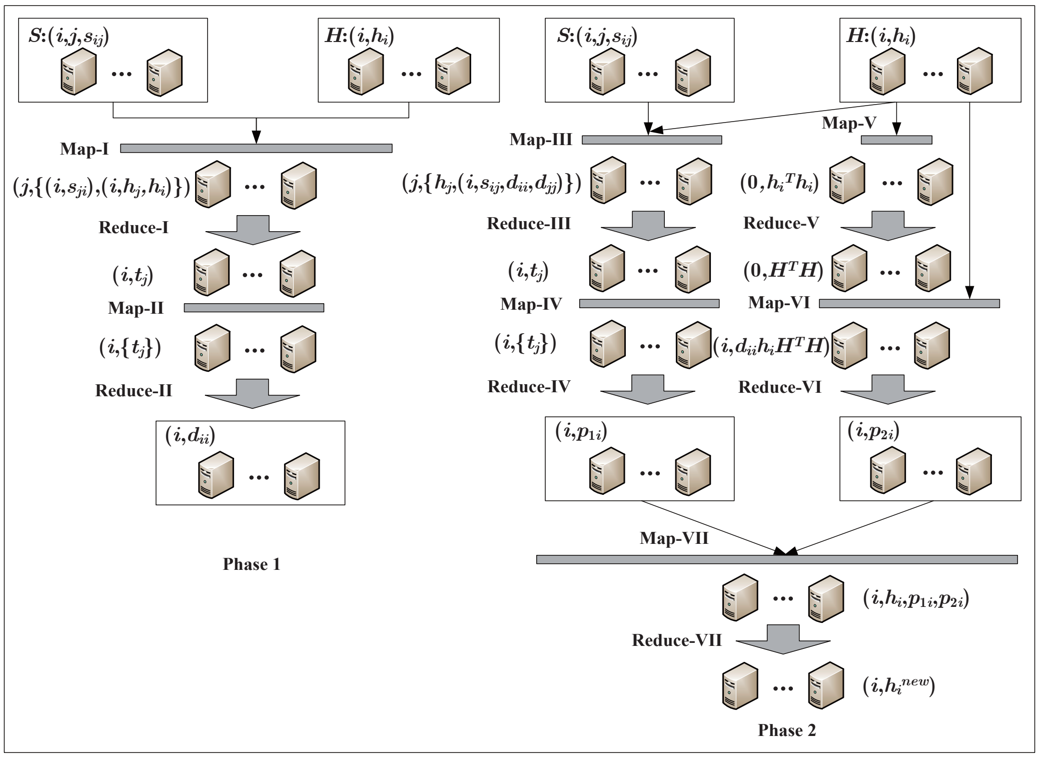

Fig. 2 Updating H on MapReduce cluster

the second one is the robustness validation, the third one is convergence analysis and the last one is the scalability test on the MapReduce cluster. Each experiment is run 10 times and the average is reported here.

4.1 Datasets

To compare the performance, we select 5 widely used complex network datasets, including Karate club member network[1], Dolphin social network[1], American college football network [1], Cora [40] and WebKB [40]. Except for Cora and WebKB datasets, Karate, Dolphin and Football datasets all have ground-truth community partitions: both Karate and Dolphin have 2, and Football have 12. These three datasets are also used to validate the robustness of DRNMFSR. To test the scalability of DRNMFSR using MapReduce, we select two large-scale online social networks datasets: LiveJournal [41] and Orkut [41]. Although these two datasets have ground-truth communities, these communities are overlapped. Our method only focuses on discovering non-overlapped communities, so we only use LiveJournal and Orkut to test the scalability of DRNMFSR. The statistics of these datasets above are summarized in Table 1.

Table 1 Statistics of datasets

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	k
Karate	34	78	2
Dolphin	62	159	2
Football	115	613	12
Cora	2708	5429	N/A
WebKB	877	1608	N/A
LiveJournal	3997962	34681189	287512
Orkut	3072441	117185083	6288363

4.2 Performance evaluation metrics

(1) Normalized Mutual Information (NMI). In the community discovery problem, NMI is a standard metric to measure the similarity of the ground-truth communities and discovered communities [42]. Given the discovered communities set C and the ground-truth communities set C', we can first construct a confusion matrix N, whose entry n_{ij} denotes the number of nodes in the ground-truth community i that appear in the discovered community j, and then $NMI(C, C')$ can be defined as follows:

$$NMI(C, C') = \frac{-2\sum_{i=1}^{|C|}\sum_{j=1}^{|C'|} n_{ij}\log(\frac{n_{ij}n}{n_{i.}n_{.j}})}{\sum_{i=1}^{|C|} n_{i.}\log(\frac{n_{i.}}{n}) + \sum_{j=1}^{|C'|} n_{.j}\log(\frac{n_{.j}}{n})}, \tag{27}$$

where $n_{i.}$ denotes the sum over row i of N and $n_{.j}$ denotes the sum over column j of N. If C is identical to C', then $NMI(C, C')$ takes its maximum value of 1. If C is totally independent of C', then $NMI(C, C')$ takes its minimum value of 0. Therefore, $NMI(C, C') \in [0, 1]$. The larger the $NMI(C, C')$ value, the better the discovered communities.

(2) Modularity Q. The modularity function Q is another way to evaluate the performance of community discovery methods, especially when the real community structure is not clear [1]. It is often used to directly evaluate the performance of a particular discovered communities set. Given $C = \{c_1, c_2, ..., c_k\}$ is the discovered communities set of $G(V, E)$, its modularity function Q can be defined as follows:

$$Q(C) = \sum_{i=1}^{k}(\frac{L(V_{c_i}, V_{c_i})}{L(V, V)} - (\frac{L(V_{c_i}, V)}{L(V, V)})^2), \tag{28}$$

where V_{c_i} is the set of nodes in c_i, $L(V_{c_i}, V_{c_i}) = \sum\limits_{v_i \in V_{c_i}, v_j \in V_{c_i}} a_{ij}$, $L(V_{c_i}, V) = \sum\limits_{v_i \in V_{c_i}, v_j \in V} a_{ij}$ and $L(V, V) = 2|E|$. Generally, a bigger $Q(C)$ corresponds to a better community partition, and hence maximizing Q can also be used to automatically determine the optimal number of k when the ground-truth communities are unknown.

In our experiments, we select NMI as the performance evaluation metric on three datasets with ground-truth community partitions (i.e., Karate, Dolphin and Football) and use Q as the performance evaluation metric on two datasets without ground-truth community partitions (i.e., Cora and WebKB).

4.3 Performance comparative experiments using different similarity measures

In this subsection, we make performance comparisons with some representative NMF-based methods for community discovery mentioned in Section 2.2, including SNMF [31], MHGNMF [34] and BSNMF [35] methods. It is worth noting that SNMF, MHGNMF and BSNMF methods all use the Frobenius norm and our method DRNMFSR is the only method using $\ell_{2,1}$ norm. In our experiments, for each method we all apply different similarity measures introduced in Section 2.1 and make comparisons by evaluating the performance of community discovery. The adjacency matrix A itself also acts as a similarity measure to be used in the tests (denoted as A).

(1) NMI comparisons

For every method, we apply different similarity measures on Karate, Dolphin and Football datasets respectively and make comparisons in term of N-MI. Note that in each experiment the value of k is set equal to the number of ground-truth communities of corresponding dataset. Comparison results are shown in Table 2. We can see that on every dataset every method using SimRank all performs better than the most frequently used adjacent matrix measure and other 4 measures (i.e., CN, RE, DK and RW). This means that SimRank measure can improve the performance of NMF-based methods for community discovery. Additionally, we also find that every NMI obtained by DRNMFSR using SimRank is equal to 1 and other four methods using SimRank all can not guarantee that every NMI is equal to 1. Therefore, DRNMF-SR performs the best in identifying the ground-truth community partitions.

(2) Q comparisons

We also firstly apply different similarity measures on Cora and WebKB datasets, and then run DRNMFSR and other three comparative methods (i.e., SNMF, MHGNMF and BSNMF), respectively. We compute the Q values by constantly varying k value of Cora in the domain of [2,100] and [2,50] for WebKB. The best Q values and its corresponding k values of every method with different similarity measures are shown in Table 3. We can see that, among 6 measures, every method using SimRank can obtain the biggest Q values. On WebKB dataset, DRNMFSR using SimRank even improve by 25% than DRNMFSR using the adjacent matrix measure. Also, we find that DRNMFSR using SimRank performs better than other three methods using SimRank. For example, on WebKB dataset the biggest Q value is 0.85, which is obtained by DRNMFSR and is far greater than the ones obtained by other three methods.

On the whole, the results of comparative experiments above all demonstrate that similarity matrix constructed using SimRank can better improve the performance of NMF-based method for community discovery. Actually,

Table 2 NMI comparison results

Method	Dataset	Measure					
		A	CN	RE	DK	RW	SR
SNMF	Karate	0.95	0.98	0.93	0.52	0.22	**1.0**
	Dolphin	0.89	0.91	0.89	0.31	0.26	**0.95**
	Football	0.92	0.92	0.67	0.39	0.31	**0.93**
MHGNMF	Karate	0.96	0.97	0.96	0.55	0.31	**1.0**
	Dolphin	0.91	0.93	0.92	0.36	0.32	**0.96**
	Football	0.92	0.92	0.69	0.41	0.35	**0.95**
BSNMF	Karate	0.94	0.95	0.93	0.53	0.24	**1.0**
	Dolphin	0.89	0.92	0.89	0.34	0.28	**0.95**
	Football	0.92	0.92	0.68	0.39	0.33	**0.94**
DRNMFSR	Karate	0.95	0.96	0.92	0.58	0.36	**1.0**
	Dolphin	0.92	0.93	0.92	0.43	0.39	**1.0**
	Football	0.92	0.92	0.73	0.45	0.42	**1.0**

Table 3 Q comparison results

Method	Dataset	Measure											
		A		CN		RE		DK		RW		SR	
		Q	k	Q	k	Q	k	Q	k	Q	k	Q	k
SNMF	Cora	0.48	9	0.45	15	0.53	25	0.29	26	0.23	26	**0.61**	**33**
	WebKB	0.53	6	0.58	5	0.41	5	0.24	6	0.21	7	**0.69**	**5**
MHGNMF	Cora	0.56	11	0.52	18	0.59	21	0.36	27	0.25	25	**0.65**	**33**
	WebKB	0.59	7	0.63	6	0.51	6	0.34	7	0.26	9	**0.73**	**6**
BSNMF	Cora	0.52	10	0.51	16	0.57	26	0.31	28	0.24	26	**0.63**	**33**
	WebKB	0.56	9	0.61	5	0.46	5	0.27	5	0.22	6	**0.70**	**7**
DRNMFSR	Cora	0.61	9	0.56	15	0.68	25	0.41	27	0.36	23	**0.76**	**33**
	WebKB	0.68	5	0.71	4	0.58	5	0.36	4	0.29	4	**0.85**	**5**

SimRank based on recursive style can combine local and global topology feature to assess the similarity of pairs of nodes in a graph. Hence, it can provide a more accurate structure feature matrix to the complex network than other measures which only consider the local topology feature (e.g., the most widely used measure: the adjacent matrix). Accurate feature matrix is very helpful to improve the performance of method for community discovery using NMF. Related experiments have also confirmed this. In addition, we also find that DRNMFSR using $\ell_{2,1}$ norm has better performance than other three NMF-based methods using Frobenius norm. This is because $\ell_{2,1}$ norm can better improve the robustness than Frobenius norm, and good robustness is also very helpful to improve the performance of NMF-based method for community discovery. In the next section, we will specially present the robustness advantages of DRNMFSR by also comparing with other three methods.

4.4 Robustness validation

We validate the robustness of DRNMFSR by comparing with SNMF, MHGN-MF and BSNMF methods on Karate, Dolphin and Football. These three datasets all have the ground-truth community partitions, so we can make noises by artificially creating links between nodes located in different communities. Supposing that the ground-truth communities is $C' = \{c'_1, ..., c'_r\}$, where every c'_i contains its own member nodes, then the number of all possible noise links is $\sum_{i=1}^{r} \sum_{j=i+1}^{r} |c'_i||c'_j|$. These virtual noise links can be used to test the robustness of every method. For the convenience of comparisons, all methods use the SimRank measure. In our experiments, for every testing dataset we firstly randomly select different ratios (we set 5 levels: 2%, 4%, 6%, 8% and 10%) of noise links to reconstruct its corresponding SimRank similarity matrix S, and then respectively evaluate the NMI performance of every method. The comparison results on every testing dataset are shown in Figure 3 to Figure 5, respectively.

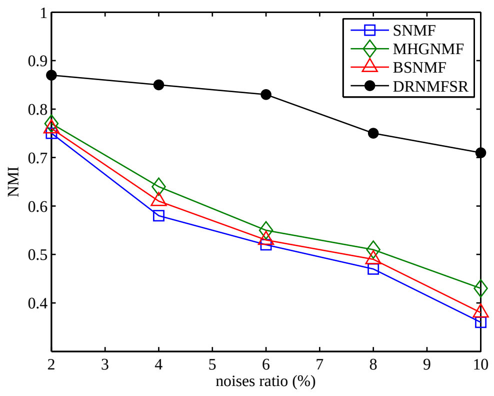

Fig. 3 Robustness comparison on Karate

From Figure 3 to Figure 5, we can see that with the ratio of noise links increasing on every testing dataset, the NMIs of SNMF, MHGNMF and B-SNMF all fall faster than DRNMFSR. For example, when we set the ratio of noise links on Football to be 10%, the NMI of DRNMFSR is 0.71, which falls by 29% than the one obtained without noise links, but it is still a relatively ideal value. However, the NMI of SNMF is 0.43 (falls by 54%), the NMI of MHGNMF is 0.47 (falls by 50%) and the NMI of BSNMF is 0.46 (falls by 51%). These values also means that their results of community discovery are very bad. On the whole, these comparison results show that DRNMFSR has better robustness than other three methods. The reason is that DRNMFSR selects the $\ell_{2,1}$ norm instead of Frobenius norm used in SNMF, MHGNMF and BSNMF to devise the objective function of NMF-based community discovery

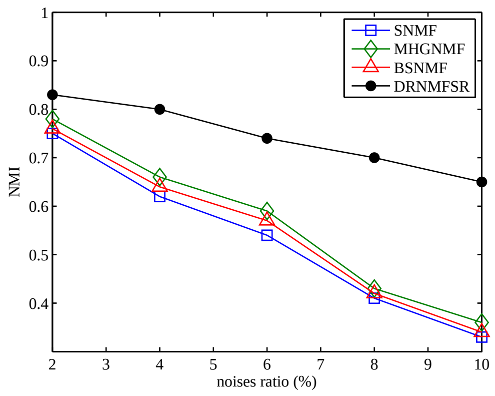

Fig. 4 Robustness comparison on Dolphin

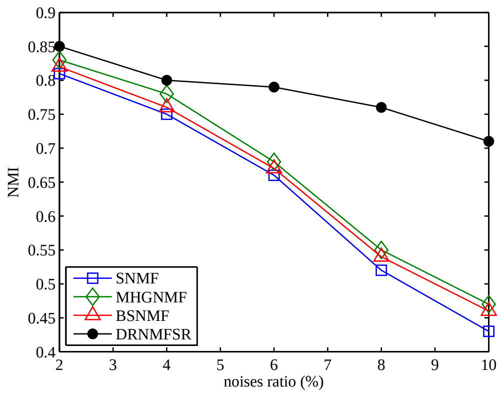

Fig. 5 Robustness comparison on Football

model, and the large errors due to noise links do not dominate the objective function because they are not squared. Therefore, DRNMFSR can improve the robustness by effectively reducing the affection caused by noise links.

4.5 Convergence analysis

We make convergence analysis for DRNMFSR by comparing with SNMF, MHGNMF and BSNMF on Karate, Dolphin, Football, Cora and WebKB. The k value for every dataset is set to its corresponding ground-truth or optimal number obtained by DRNMFSR. The major comparison metrics include the iterative number of convergence: Iterations and convergence time. For the fairness of comparisons, all methods use the SimRank measure and are implemented using serial programming model. All the experiments are conducted on a machine powered by a Intel Core(TM) i7-6700 CPU 3.4GHz with 16G-

B RAM, running Windows 7 64bits and the program language is Java. Our experiment results are shown in Table 4. The results show that, on every dataset DRNMFSR all takes less iterations and time to converge than other three methods. Furthermore, the larger the dataset scale is, the more obvious the advantage of DRNMFSR is. For example, on Cora DRNMFSR only needs 85 iterations and 31.25 seconds to converge, which respectively decreases by 29.6% and 32.5% than the corresponding suboptimal results obtained by MHGNMF. Although DRNMFSR using $\ell_{2,1}$ norm needs to additionally update the diagonal matrix D in every iterations, it needs less iterations which can make it run faster than SNMF, MHGNMF and BSNMF.

Table 4 Convergence analysis results

Dataset	Metric	SNMF	MHGNMF	BSNMF	DRNMFSR
Karate	Iterations	30	27	28	**16**
	Time (sec.)	0.036	0.025	0.029	**0.023**
Dolphin	Iterations	34	30	31	**18**
	Time (sec.)	0.049	0.032	0.038	**0.029**
Football	Iterations	77	65	69	**35**
	Time (sec.)	1.876	1.564	1.623	**0.986**
Cora	Iterations	149	121	136	**85**
	Time (sec.)	56.78	46.32	49.73	**31.25**
WebKB	Iterations	108	87	98	**67**
	Time (sec.)	33.13	26.56	31.79	**19.21**

4.6 Scalability test

In this subsection, we conduct experiments on two large-scale datasets: LiveJournal and Orkut to test the scalability of DRNMFSR using MapReduce. Our experimental platform is a cluster, which is composed of 10 machines with the same configuration: 2.93G Hz CPU, 4GB memory and 1TB disk. We select Twister [43] as the MapReduce distributed computing framework, which is more suitable for running iterative MapReduce program than other similar frameworks (e.g., Hadoop or HaLoop). To evaluate the the scalability of DRNMFSR running on MapReduce cluster, we introduce the $Speedup = T_s/T_c$ as the evaluation criterion, where T_s is the running-time of machine clusters under standalone mode, namely only including one worker machine, and T_c is the running time of machine cluster with specified number of worker machines. When *Speedup* is more proportional to the number of worker machines, the scalability of DRNMFSR will be better. Our experiments include two parts: computing SimRank and updating H on MapReduce cluster. For convenience, on every testing dataset we all set the iterative numbers of computing SimRank and updating H to be the fixed values. Through increasing the number of worker machines in MapReduce cluster from 1 to 10, we obtain the corresponding running-time of computing SimRank and updating H on LiveJournal and Orkut. The corresponding speedup changing curves are shown in Figure 6 and Figure 7, respectively.

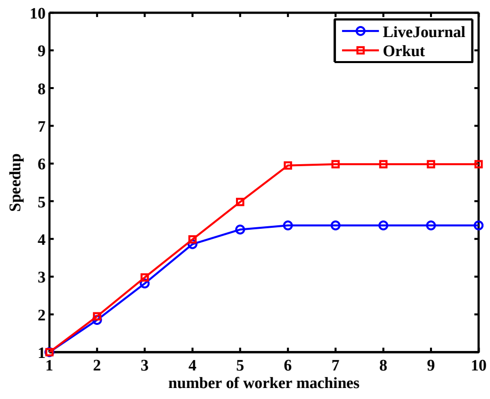

Fig. 6 Speedup of computing SimRank

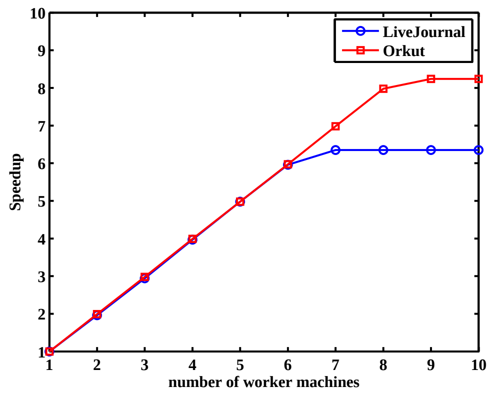

Fig. 7 Speedup of updating H

In Figure 6 and Figure 7 all the speedup changing curves increase nearly linearly at the beginning and stabilize at a certain value when the number of worker machines increase to a threshold. The reason is that the size of dataset limits the number of MapReduce tasks. When the number of worker machines is bigger than the number of tasks, the tasks have already been fully parallelized and the extra worker machines in the cluster no longer help increase the speedup. To confirm this finding, we double the amount of data by simply duplicating original datasets and conduct the same experiments. We find that all of new speedup changing curves always increase nearly linearly when the number of worker machines increase from 1 to 10. This means that all of worker machines undertake computation tasks. Generally, these results above indicate that DRNMFSR using MapReduce is scalable enough to deal with the large-scale complex networks.

5 Conclusions

Many existing works have proved that NMF is a very effective model for community discovery in complex networks, but there are still many issues worthy of further research. In this paper, we focus on how to improve NMF-based community discovery and propose a new method using distributed robust NMF with SimRank similarity measure (DRNMFSR). DRNMFSR selects SimRank similarity measure to construct the feature matrix of complex networks and can improve the performance of community discovery by more accurately representing the global structure feature of complex networks. Meanwhile, DRNMFSR has better robustness by using $\ell_{2,1}$ norm instead of the widely used Frobenius norm and is very effective to handle noises existing in complex networks. Furthermore, this method can be implemented using MapReduce distributed computing framework and has a good scalability to resolve the problem of community discovery in the large-scale complex networks. We conduct extensive experiments on several typical complex networks and the results show that our method is very effective. Considering that recently there have appeared several other new similarity measures for graph nodes, including network representation based methods and deep learning based methods, we will further conduct comparative experiments with these new measures in our future work. Additionally, we will study more strategies to further improve the efficiency of our method (e.g., implemented and optimized using the memory computing based framework: Spark).

Acknowledgments

This work was partially supported by the following projects: National Natural Science Foundation of China (No. 61772211), Natural Science Foundation of Guangdong Province, China (No. 2016A030313441), Science and Technology Support Program of Guangdong Province, China (Nos. 2017A040405057, 2017A030303074, 2016A030303058, 2017B010126001, 2017A070712019, 2016A070712020, 2015A040405014, 2016A040402043, 2016A030303058), Science and Technology Support Program of GuangZhou City, China (Nos. 201604016035, 201604046017, 201704020030, 201704030098, 201807010043), Research Projects of Universities in Guangdong Province of China (No. 2016KZDXM001).

References

1. Newman ME, Girvan M (2004) Finding and evaluating community structure in networks. Physical Review E 69:026113
2. Hao F, Park DS (2018) cSketch: a novel framework for capturing cliques from big graph. The Journal of Supercomputing, 74(3):1202-1214
3. Newman ME (2013) Spectral methods for community detection and graph partitioning. Physical Review E 88:042822
4. Ianni MD, Gambosi G, Rossi G, et al. (2016) Min-max communities in graphs: complexity and computational properties. Theoretical Computer Science 613: 94-114
5. Peng DL, Lei X, Huang T (2015) DICH: a framework for discovering implicit communities hidden in tweets. World Wide Web 18(4):795-818
6. Xiang J, Tao H, Zhang Y, Hu K, et al. (2016) Local modularity for community detection in complex networks. Physica A 443:451-459
7. Shakarian P, Roos P, Callahan D, et al. (2013) Mining for geographically disperse communities in social networks by leveraging distance modularity. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp 1402-1409
8. Clauset A, Newman ME, Moore C (2004) Finding community structure in very large networks. Physical Review E 70:066111
9. Zhu WY, Peng WC, Hung CC, et al. (2014) Exploring sequential probability tree for movement-based community discovery. IEEE Transactions on Knowledge and Data Engineering 6(11):2717-2730
10. Costa G, Ortale R (2013) Probabilistic analysis of communities and inner roles in networks: bayesian generative models and approximate inference. Social Network Analysis and Mining 3(4):1015-1038
11. Zhang Y, Yeung DY (2012) Overlapping community detection via bounded nonnegative matrix tri-factorization. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp 606-614
12. Ma Y, Hu X, He T, et al. (2017) Clustering and integrating of heterogeneous microbiome data by joint symmetric nonnegative matrix factorization with laplacian regularization. IEEE/ACM Transactions on Computational Biology and Bioinformatics 99:1
13. He CB, Li HC, Fei X, et al. (2017) A topic community-based method for friend recommendation in large-scale online social networks. Concurrency and Computation: Practice and Experience 29(6):e3924
14. Liu SY, Wang SH (2017) Trajectory community discovery and recommendation by multi-source diffusion modeling. IEEE Transactions on Knowledge and Data Engineering 29(4):898-911
15. Yang L, Cao XC, Jin D, et al. (2015) A unified semi-supervised community detection framework using latent space graph regularization. IEEE Transactions on Cybernetics, 45(11):2585-2598
16. Li WM, Xie J, Xin MJ, et al. (2018) An overlapping network community partition algorithm based on semi-supervised matrix factorization and random walk. Expert Systems with Applications 91:277-285
17. He CB, Fei X, Li HC, et al. (2017) Community discovery in large-scale complex networks using distributed SimRank nonnegative matrix factorization. In: Proceedings of the 5th International Conference on Advanced Cloud and Big Data. IEEE, pp 226-231
18. Leicht EA, Holme P, Newman, ME (2006) Vertex similarity in networks. Physical Review E 73:026120
19. Kondor RI, Lafferty J (2002) Diffusion kernels on graphs and other discrete input space. In: Proceedings of the 19th International Conference on Machine Learning. IEEE, pp 315-322
20. Liu WP, Lü LY (2010) Link prediction based on local random walk. Europhysics Letters 89(5): 58007-58012
21. Jeh G, Widom J (2002) SimRank: a measure of structural-context similarity. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp 538-543

22. Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401:788-791
23. Lee DD, Seung HS (2000) Algorithms for non-negative matrix factorization. In: Proceedings of the 24th Annual Conference on Neural Information Processing Systems.Curran Associates, pp 556-562
24. Ling Y, Pan XL, Li GR, et al. (2015) Clinical documents clustering based on medication/symptom names using multi-view nonnegative matrix factorization. IEEE Transactions on NanoBioscience 14(5):500-504
25. Trung ND, Schmidt-Thieme L, Trung ND, et al. (2017) On discovering the number of document topics via conceptual latent space. In: Proceedings of the 26th ACM on Conference on Information and Knowledge Management. ACM, pp 2051-2054
26. Mohammadiha N, Smaragdis P, Leijon A (2013) Supervised and unsupervised speech enhancement using nonnegative matrix factorization. IEEE Transactions on Audio, Speech, and Language Processing 21(10):2140-2151
27. Yang ZG, Li Q, Liu WY, et al. (2017) Dual graph regularized NMF model for social event detection from Flickr data. World Wide Web, 20(5):995-1015
28. Lai HJ, Yan P, Shu XB, et al. (2016) Instance-aware hashing for multi-label image retrieval. IEEE Transactions on Image Processing, 25(6):2469-2479
29. Kannan R, Ishteva M, Park H (2014) Bounded matrix factorization for recommender system. Knowledge and Information Systems, 39(3):491-511
30. Ding CHQ, He X (2015) On the equivalence of nonnegative matrix factorization and spectral clustering. In: Proceedings of 5th SIAM International Conference on Data Mining. ACM, pp 606-610
31. Wang F, Li T, Wang X, et al. (2011) Community discovery using nonnegative matrix factorization. Data Mining and Knowledge Discovery 22(3):493-521
32. Nguyen NP, Thai MT (2012) Finding overlapped communities in online social networks with nonnegative matrix factorization. In: Proceedings of 33rd IEEE Military Communications Conference. IEEE, pp 1-6
33. Zhang ZY, Wang Y, Ahn YY (2013) An overlapping community detection in complex networks using symmetric binary matrix factorization. Physical Review E 87:062803
34. Wu WH, Kwong S, Zhou Y, et al. (2018) Nonnegative matrix factorization with mixed hypergraph regularization for community detection. Information Sciences 435:263-281
35. Ma XK, Dong D. Evolutionary nonnegative matrix factorization algorithms for community detection in dynamic networks. IEEE Transactions on Knowledge and Data Engineering 29(5):1045-1058
36. Liu X, Wang WJ, He DX, et al. (2017) Semi-supervised community detection based on non-negative matrix factorization with node popularity, Information Sciences 381:304-321
37. Shi XH, Lu HT (2016) Community inference with Bayesian non-negative matrix factorization. In: Proceedings of the 18th Asia-Pacific Web Conference on Web Technologies and Applications. Springer, pp 208-219
38. Kuang D, Yun S, Park H (2015) SymNMF: nonnegative low-rank approximation of a similarity matrix for graph clustering. Journal of Global Optimization 62(3):545-574
39. Yu WR, Lin XM, Zhang WJ (2013) Towards efficient SimRank computation on large networks. In: Proceedings of the 29th IEEE International Conference on Data Engineering. IEEE, pp 601-612
40. Du YT, Guan XH, Cai ZM (2010) Enhancing web page classification via local co-training. In: Proceedings of the 20th International Conference on Pattern Recognition. IEEE, pp 23-26
41. Yang J, Leskovec J (2012) Defining and evaluating network communities based on ground-truth. In: Proceedings of 12th IEEE International Conference on Data Mining. IEEE, pp 745-754
42. Lai DR, Nardini C (2016) A corrected normalized mutual information for performance evaluation of community detection. Journal of Statistical Mechanics Theory & Experiment, 2016(9):093403
43. Ekanayake J, Li H, Zhang BJ, et al. (2010) Twister: a runtime for iterative MapReduce. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. ACM, pp 810-818