Local Community Detection in Multiple Networks

Dongsheng Luo  
Pennsylvania State University  
dul262@ist.psu.edu  

Yuchen Bian  
Baidu Research, USA  
yuchenbian@baidu.com  

Yaowei Yan  
Pennsylvania State University  
yxy230@psu.edu  

Xiao Liu  
Pennsylvania State University  
xxl213@psu.edu  

Jun Huan  
StylingAI Inc.  
lukehuan@shenshangtech.com  

Xiang Zhang  
Pennsylvania State University  
xzhang@ist.psu.edu  

ABSTRACT

Local community detection aims to find a set of densely-connected nodes containing given query nodes. Most existing local community detection methods are designed for a single network. However, a single network can be noisy and incomplete. Multiple networks are more informative in real-world applications. There are multiple types of nodes and multiple types of node proximities. Complementary information from different networks helps to improve detection accuracy. In this paper, we propose a novel RWM (Random Walk in Multiple networks) model to find relevant local communities in all networks for a given query node set from one network. RWM sends a random walker in each network to obtain the local proximity w.r.t. the query nodes (i.e., node visiting probabilities). Walkers with similar visiting probabilities reinforce each other. They restrict the probability propagation around the query nodes to identify relevant subgraphs in each network and disregard irrelevant parts. We provide rigorous theoretical foundations for RWM and develop two speeding-up strategies with performance guarantees. Comprehensive experiments are conducted on synthetic and real-world datasets to evaluate the effectiveness and efficiency of RWM.

CCS CONCEPTS

- Information systems → Data mining;

ACM Reference Format:


1 INTRODUCTION

As a fundamental task in large network analysis, local community detection has attracted extensive attention recently. Unlike the time-consuming global community detection, the goal of local community detection is to detect a set of nodes with dense connections (i.e., the target local community) that contains a given query node or a query node set [1, 2, 4, 6, 7, 15, 17, 29, 33].

Most existing local community detection methods are designed for a single network. However, a single network may contain noisy edges and miss some connections which results in unsatisfactory performance. On the other hand, in real-life applications, more informative multiple network structures are often constructed from different sources or domains [18, 20, 22, 25, 28].

There are two main kinds of multiple networks. Fig. 1 shows an example of multiple networks with the same node set and different types of edges (which is called multiplex networks [14, 21] or multi-layer network). Each node represents an employee of a university [13]. These three networks reflect different relationships between employees: co-workers, lunch-together, and Facebook-friends. Notice that more similar connections exist between two off-line networks (i.e., co-worker and lunch-together) than that between the off-line and online relationships (i.e., Facebook-friends).

Fig. 2 is another example from the DBLP dataset with multiple domains of nodes and edges (which is called multi-domain networks [22]). The left part is the author-collaboration network and the right part is the paper-citation network. A cross-edge connecting an author and a paper indicates that the author writes the paper.
We see that authors and papers in the same research field may have dense cross-links. Suppose that we are interested in one author, we want to find both the relevant author community as well as the paper community, which includes papers in the research field of the query author. Then cross-edges can provide complementary information from the query (author) domain to other domains (paper) and vice versa. Detecting relevant local communities in two domains can enhance each other.

Note that multiplex networks are a special case of multi-domain networks with the same set of nodes and the cross-network relations are one-to-one mappings. Few works discuss local community detection in comprehensive but complicated multiple networks. Especially, no existing work can find query-relevant local communities in all domains in the more general multi-domain networks.

In this paper, we focus on local community detection in multiple networks. Given a query node in one network, our task is to detect a query-relevant local community in each network.

A straightforward method is to independently find the local community in each network. However, this baseline only works for the special multiplex networks and cannot be generalized to multi-domain networks. In addition, this simple approach does not consider the complementary information in multiple networks (e.g., similar local structures among the three relations in Fig. 1).

In this paper, we propose a random walk based method, RWM (Random Walk in Multiple networks), to find query-relevant local communities in multiple networks simultaneously. The key idea is to integrate complementary influences from multiple networks. We send out a random walker in each network to explore the network topology based on corresponding transition probabilities. For networks containing the query node, the probability vectors of walkers are initiated with the query node. For other networks, probability vectors are initialized by transiting visiting probability via cross-connections among networks. Then probability vectors of the walkers are updated step by step. The vectors represent walkers’ visiting histories. Intuitively, if two walkers share similar or relevant visiting histories (measured by cosine similarity and transited with cross-connections), it indicates that the visited local typologies in corresponding networks are relevant. And these two walkers should reinforce each other with highly weighted influences. On the other hand, if visited local typologies of two walkers are less relevant, smaller weights should be assigned. We update each walker’s visiting probability vector by aggregating other walkers’ influences at each step. In this way, the transition probabilities of each network are modified dynamically. Comparing with traditional random walk models where the transition matrix is time-independent [2, 15, 17, 30, 31], RWM can restrict most visiting probability in the most query-relevant subgraph in each network and ignore irrelevant or noisy parts.

Theoretically, we provide rigorous analyses of the convergence properties of the proposed model. Two speeding-up strategies are developed as well. We conduct extensive experiments on real and synthetic datasets to demonstrate the advantages of our RWM model on effectiveness and efficiency for local community detection in multiple networks.

Our contributions are summarized as follows.

- We propose a novel random walk model, RWM, to detect local communities in multiple networks for a given query node set. Especially, based on our knowledge, this is the first work which can detect all query-relevant local communities in all networks for the general multi-domain networks.
- Fast estimation strategies and sound theoretical foundations are provided to guarantee effectiveness and efficiency of RWM.
- Results of comprehensive experiments on synthetic and real multiple networks verify the advances of RWM.

2 RELATED WORK

Local community detection is one of the cutting-edge problems in graph mining. In a single network, local search based method [23, 28], flow-based method [29], and subgraph cohesiveness-optimization based methods, such as k-clique [24, 36], k-truss [9, 11], k-core [5] are developed to detect local communities. A recent survey about local community detection and community search can be found in [10].

Except for that, random walk based methods have also been routinely applied to detect local communities in a single network [2, 5, 8, 15, 32, 34, 35]. A walker explores the network following the topological transitions. The node visiting probability is usually utilized to determine the detection results. For example, PRN [2] runs the lazy random walk to update the visiting probability and sweeps the ranking list to find the local community with the minimum conductance. The heat kernel method applies the heat kernel coefficients as the decay factor to find small and dense communities [15]. In [35], the authors propose a motif-based random walk model and obtain node sets with the minimal motif conductance. MWC [5] sends multiple walkers to explore the network to alleviate the query-bias issue. Note that all aforementioned methods are designed for a single network.

Some methods have been proposed for global community detection in multiple networks. An overview can be found in [13]. However, they focus on detecting all communities, which are time-consuming and are independent of a given query node. Very limited work aim to find query-oriented local communities in multiple networks. For the special case, i.e., multiplex networks, a greedy algorithm is developed to find a community shared by all networks [12]. Similarly, a modified random walk model is proposed in [8] for multiplex networks. For the general multi-domain networks, the method in [34] can only detect one local community in the query network domain. More importantly, these methods assume that all networks share similar or consistent structures. Different from these methods, our method RWM does not make such an assumption. RWM can automatically identify relevant local structures and ignore irrelevant ones in all networks.

3 RANDOM WALK IN MULTIPLE NETWORKS

In this section, we first introduce notations. Then the reinforced updating mechanism among random walkers in RWM is proposed. A query-restart strategy is applied in the end to detect the query-relevant local communities in multiple networks.
Table 1: Main notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( K )</td>
<td>The number of networks</td>
</tr>
<tr>
<td>( G_i = (V_i, E_i) )</td>
<td>The ( i )th network with nodes set ( V_i ) and edge set ( E_i )</td>
</tr>
<tr>
<td>( P_i )</td>
<td>Column-normalized transition matrix of ( G_i )</td>
</tr>
<tr>
<td>( E_{i \rightarrow j} )</td>
<td>The cross-edge set between network ( G_i ) and ( G_j )</td>
</tr>
<tr>
<td>( S_{i \rightarrow j} )</td>
<td>Column-norm. cross-trans. mat. from ( V_i ) to ( V_j )</td>
</tr>
<tr>
<td>( u_q )</td>
<td>A given query node ( u_q ) from ( V_q )</td>
</tr>
<tr>
<td>( e_q )</td>
<td>One-hot vector with only one value-1 entry for ( u_q )</td>
</tr>
<tr>
<td>( x^{(t)}_{i} )</td>
<td>Node visit. prob. vec. of the ( i )th walker in ( G_i ) at ( t )</td>
</tr>
<tr>
<td>( W^{(t)} )</td>
<td>Relevance weight matrix at time ( t )</td>
</tr>
<tr>
<td>( \mathcal{P}^{(t)}_i )</td>
<td>Modified trans. mat. for the ( i )th walker in ( G_i ) at ( t )</td>
</tr>
<tr>
<td>( \alpha, \lambda, \theta )</td>
<td>Restart factor, decay factor, covering factor</td>
</tr>
</tbody>
</table>

3.1 Notations

Suppose that there are \( K \) undirected networks, the \( i \)th \((1 \leq i \leq K)\) network is represented by \( G_i = (V_i, E_i) \) with node set \( V_i \) and edge set \( E_i \). We denote its transition matrix as a column-normalized matrix \( P_i \in \mathbb{R}^{|V_i| \times |V_i|} \). The \((s, u)\)th entry \( P_i(s, u) \) represents the transition probability from node \( s \) to node \( u \) in \( V_i \). Then the \( u \)th column \( P_i(\cdot, u) \) is the transition distribution from node \( u \) to all nodes in \( V_i \). Next, we denote \( E_{i \rightarrow j} \) the cross-connections between nodes in two networks \( V_i \) and \( V_j \). The corresponding cross-transition matrix is \( S_{i \rightarrow j} \in \mathbb{R}^{|V_i| \times |V_j|} \). Then the \( u \)th column \( S_{i \rightarrow j}(\cdot, u) \) is the transition distribution from node \( u \) to all nodes in \( V_j \). Note that \( S_{i \rightarrow j} \neq S_{j \rightarrow i} \). And for the multiplex networks with the same node set (e.g., Fig. 1), \( S_{i \rightarrow j} \) is just an identity matrix I for arbitrary \( i, j \).

Suppose we send a random walker in \( G_i \), we let \( x^{(t)}_i \) be the node visiting probability vector in \( G_i \) at time \( t \). Then the updated vector \( x^{(t+1)}_i = P_i x^{(t)}_i \) means the probability transitioning among nodes in \( G_i \). And \( S_{i \rightarrow j} x^{(t)}_i \) is the probability vector propagated into \( G_j \) from \( G_i \).

Given a query node \( u_q \) in the query network \( G_q \), the target of local community detection in multiple networks is to detect relevant local communities in all networks \( G_i (1 \leq i \leq K) \). Important notations are summarized in Table 1.

3.2 Reinforced Updating Mechanism

In RWM, we send out one random walker for each network. Initially, for the query node \( u_q \in G_q \), the corresponding walker’s \( x^{(0)}_q = e_q \) where \( e_q \) is a one-hot vector with only one value-1 entry corresponding to \( u_q \). For other networks \( G_i (i \neq q) \), we initialize \( x^{(0)}_i = S_{q \rightarrow i} x^{(0)}_q \). That is:

\[
\begin{align*}
    x^{(0)}_i &= e_q \quad \text{if } i = q \\
    x^{(0)}_i &= S_{q \rightarrow i} e_q \quad \text{otherwise}
\end{align*}
\]

To update \( i \)th walker’s vector \( x^{(t)}_i \), the walker not only follows the transition probabilities in the corresponding network \( G_i \), but also obtains influences from other networks. Intuitively, networks sharing relevant visited local structures should influence each other with higher weights. And the ones with less relevant visited local structures have fewer effects. We measure the relevance of visited local structures of two walkers in \( G_i \) and \( G_j \) with the cosine similarity of their vectors \( x^{(t)}_i \) and \( x^{(t)}_j \). Since different networks consist of different node sets, we define the relevance as \( \cos(x^{(t)}_i, S_{j \rightarrow i} x^{(t)}_j) \). Notice that when \( t \) increases, walkers will explore nodes further away from the query node. Thus, we add a decay factor \( \lambda (0 < \lambda < 1) \) in the relevance to emphasize the similarity between local structures of two different networks within a shorter visiting range. In addition, as shown in the theoretical analysis in Appendix, \( \lambda \) can guarantee and control the convergence of the RWM model. Formally, we let \( W^{(t)}(i, j) \in \mathbb{R}^{K \times K} \) be the local relevance matrix among networks at time \( t \) and initialize it with identity matrix I. We update each entry \( W^{(t)}(i, j) \) as follow:

\[
W^{(t)}(i, j) = W^{(t-1)}(i, j) + \lambda^t \cos(x^{(t)}_i, S_{j \rightarrow i} x^{(t)}_j)
\]

For the \( i \)th walker, influences from other networks are reflected in the dynamic modification of the original transition matrix \( P_i \) based on the relevance weights. Specifically, the modified transition matrix of \( G_i \) is:

\[
\mathcal{P}^{(t)}_i = \sum_{j=1}^{K} W^{(t)}(i, j) S_{j \rightarrow i} P_j S_{i \rightarrow j}
\]

where \( S_{j \rightarrow i} P_j S_{i \rightarrow j} \) represents the propagation flow pattern \( G_i \rightarrow G_j \rightarrow G_i \) (counting from the right side) and \( W^{(t)}(i, j) = \sum_{k} W^{(t)}(i, k)W^{(t)}(k, j) \) is the row-normalized local relevance weights from \( G_j \) to \( G_i \). To guarantee the stochastic property of transition matrix, we also column-normalize \( \mathcal{P}^{(t)}_i \) after each updating.

At time \( t+1 \), the visiting probability vector of the walker in \( G_i \) is updated:

\[
x^{(t+1)}_i = \mathcal{P}^{(t)}_i x^{(t)}_i
\]

Different from the classic random walk model, transition matrices in RWM dynamically evolve with local relevance influences among walkers from multiple networks. As a result, the time-dependent property enhances RWM with the power of aggregating relevant and useful local structures among networks for the local community detection.

Next, we theoretically analyze the convergence properties. First, in Theorem 1, we present the weak convergence property [5] of the modified transition matrix \( \mathcal{P}^{(t)}_i \). The convergence of the visiting probability vector will be provided in Theorem 3.

**Theorem 1.** When applying RWM in multiple networks, for any small tolerance \( 0 < \varepsilon < 1 \), for all \( i \), when \( t > \frac{\log_2 (1/\varepsilon)}{\log_2 (K/|V_i|)} \),

\[
\| x^{(t+1)}_i - x^{(t)}_i \|_\infty < \varepsilon, \text{ where } V_i \text{ is the node set of network } G_i \text{ and } \| \cdot \|_\infty \text{ is the } \infty\text{-norm of a matrix}.
\]

For the multiple networks with the same node set (i.e., multiplex networks), we have a faster convergence rate.

http://personal.psu.edu/dul262/RWM/appendix.pdf
Theorem 2. When applying RWM in multiple networks with the same node set, for any small tolerance 0 < ε < 1, for all i, \[ \| P_i^{(t+1)} - P_i^{(t)} \|_m < \varepsilon, \text{ when } t > [\log \frac{\varepsilon}{2}] \].

Please refer to the Appendix for the proof details of these theorems. Next, we describe how to detect relevant local communities in multiple networks given a query node from one network.

4 LOCAL COMMUNITY DETECTION WITH RESTART STRATEGY

RWM is a general random walk model for multiple networks and can be further customized into different variations. In this section, we integrate the idea of random walk with restart (RWR) [26] into RWM and use it for local community detection. Note that the original RWR is only for a single network instead of multiple networks.

In RWR, at each time point, the random walker explores the network based on topological transitions with \( a(0 < a < 1) \) probability and jumps back to the query node with probability \( 1 - a \). The restart strategy enables RWR to obtain proximities of all nodes to the query node.

Similarly, we apply the restart strategy for RWM in updating visiting probability vectors. For the \( i \)th walker, we have:

\[
W_i^{(t+1)}(i,j) = W_i^{(t)}(i,j) + \lambda (1 - a) W_i^{(0)}(i,j)
\]

where \( W_i^{(t)}(i,j) \) is obtained by Eq. (3). Since the restart component does not provide any information for the visited local structure, we dismiss this part when calculating local relevance weights. Therefore, we modify the \( \cos(\cdot, \cdot) \) in Eq. (2) and have:

\[
W(i,j) = W^{(t-1)}(i,j) + \lambda^t \cos((x_i^{(t)} - (1 - a)x_i^{(0)}), S_{j,i}x_i^{(t)} - (1 - a)x_i^{(0)})
\]

Theorem 3. Adding the restart strategy in RWM does not affect the convergence properties in Theorems 1 and 2. The visiting vector \( x_i^{(t)}(1 \leq i \leq K) \) will also converge.

We skip the proof here. The main idea is that \( \lambda \) first guarantees the weak convergence of \( P_i^{(t)} (\lambda \) has the same effect as in Theorem 1). After obtaining a converged \( P_i^{(t)} \), Perron-Frobenius theorem [19] can guarantee the convergence of \( x_i^{(t)} \) with a similar convergence proof of the traditional RWR model.

To find the local community in network \( G_i \), we follow the common process in the literature [2, 5, 15, 34]. We first sort nodes according to the converged score vector \( x_i^{(T)} \), where \( T \) is the number of iterations. Suppose there are \( L \) non-zero elements in \( x_i^{(T)} \), for each \( I (1 \leq I \leq L) \), we compute the conductance of the subgraph induced by the top \( I \) ranked nodes. The node set with the smallest conductance will be considered as the detected local community.

Time complexity of RWM. As a basic method, we can iteratively update vectors until convergence or stop the updating at a given number of iterations \( T \). Algorithm 1 shows the overall process.

In each iteration, for the \( I \)th walker (line 4–5), we update \( x_i^{(t+1)} \) based on Eq. (3) and (5) (line 4). Note that we do not compute the modified transition matrix \( P_i^{(t)} \) and output. If we substitute Eq. (3) in Eq. (5), we have a probability propagation \( S_{j,i}P_jS_{i,j}x_i^{(t)} \) which reflects the information flow \( G_i \rightarrow G_j \rightarrow G_i \). In practice, a network is stored as an adjacent list. So we only need to update the visiting probabilities of direct neighbors of visited nodes and compute the propagation from right to left. Then calculating \( S_{j,i}P_jS_{i,j}x_i^{(t)} \) costs \( O(|V_i| + |E_{i,j}| + |E_{j,i}||V_j|) \). And the restart addition in Eq. (5) costs \( O(|V_i|) \). As a result, line 4 costs \( O(|V_i| + |E_i| + \sum_{|j\neq i}(|E_{i,j}| + |E_{j,i}|)) \) where \( O(|E_i|) \) is from the propagation in \( G_i \) itself.

In line 5, based on Eq. (6), it takes \( O(|E_{i,j}| + |V_j|) \) to get \( S_{j,i}x_i^{(t)} \) and \( O(|V_j|) \) to compute cosine similarities. Then updating \( W(i,j) \) (line 5) costs \( O(|E_{i,j}| + |V_i| + |V_j|) \). In the end, normalization (line 6) costs \( O(K^2) \) which can be ignored.

In summary, with \( T \) iterations, power iteration method for RWM costs \( O((\sum_{i}|V_i| + \sum_{i}|E_i| + \sum_{|j\neq i}(|E_{i,j}| + |E_{j,i}|))KT) \). For the multiplex networks with the same node set \( V \), the complexity shrinks to \( O((\sum_{i}|E_i| + K|V|)KT) \) because of the one-to-one mappings in \( E_{i,j} \).

Note that power iteration methods may propagate probabilities to the entire network which is not a true “local” method. Next, we present two speeding-up strategies to restrict the probability propagation into only a small subgraph around the query node.

5 SPEEDING UP

In this section, we introduce two approximation methods which not only dramatically improve the computational efficiency but also guarantee performances.

5.1 Early Stopping

With the decay factor \( \lambda \), the modified transition matrix of each network converges before the visiting score vector does. Thus, we can approximate the transition matrix with early stopping by splitting the computation into two phases. In the first phase, we update both transition matrices and score vectors, while in the second phase, we keep the transition matrices static and only update score vectors.

Now we give the error bound between the early-stopping updated transition matrix and the power-iteration updated one. In \( G_i \), we denote \( P_i^{(0)} \) the converged modified transition matrix.

In \( G_i (1 \leq i \leq K) \), when properly selecting the split time \( T_e \), the following theorem demonstrates that we can securely approximate the power-iteration updated matrix \( P_i^{(\infty)} \) with \( P_i^{(T_e)} \).
Theorem 4. For a given small tolerance \( \epsilon \), when \( t > T_\epsilon = \lceil \log_\lambda \frac{\epsilon (1-\lambda)}{K} \rceil \), \( \| P_i^{(t)} - P_i^{(0)} \|_\infty < \epsilon \). For the multiplex networks with the same node set, we can choose \( T_\epsilon = \lceil \log_\lambda \frac{\epsilon (1-\lambda)}{K} \rceil \) to get the same estimation bound \( \epsilon \).

Please refer to Appendix for the proof.

The time complexity of the first phase is \( O((\sum_i |V_i| + \sum_i |E_i| + \sum_{i \neq j} |E_{i,j}|)K T_\epsilon) \).

Algorithm 2: Partial Updating

Input: \( x_i^{(0)}, x_i^{(t)} \), \( \{P_i\}_i \), \( \{S_{i \rightarrow j}\}_{i,j} \), \( \lambda, \alpha, \theta \)
Output: approximation score vector \( \tilde{x}_i^{(t+1)} \)
1. initialize an empty queue \( que \);
2. initialize a zero vector \( x_0^{(t)} \) with the same size with \( x_i^{(t)} \);
3. \( que.push(Q) \) where \( Q \) is the node set with positive values in \( x_0^{(t)} \);
4. \( cover \leftarrow 0 \);
5. while \( que \) is not empty and \( cover < \theta \) do
6. \( u \leftarrow que.pop(); \)
7. mark \( u \) as visited;
8. for each neighbor node \( v \) of \( u \) do
9. if \( v \) is not marked as visited then
10. \( que.push(v); \)
11. \( x_0^{(t)}[u] \leftarrow \tilde{x}_i^{(t)}[u]; \)
12. \( cover \leftarrow cover + \tilde{x}_i^{(t)}[u]; \)
13. Update \( \tilde{x}_i^{(t+1)} \) based on Eq. (7);

5.2 Partial Updating

In this section, we propose a heuristic strategy to further speed up the vector updating in Algorithm 1 (line 4) by only updating a subset of nodes that covers most probabilities.

Specifically, given a covering factor \( \theta \in (0, 1] \), for walker \( i \), in the \( t \)th iteration, we separate \( x_i^{(t)} \) into two non-negative vectors, \( x_0^{(t)} \) and \( \Delta x_i^{(t)} \), so that \( x_i^{(t)} = x_0^{(t)} + \Delta x_i^{(t)} \), and \( \| x_0^{(t)} \|_1 \geq \theta \). Then, we approximate \( x_i^{(t+1)} \) with

\[
\tilde{x}_i^{(t+1)} = \alpha P_i^{(t)} x_0^{(t)} + (1 - \alpha) \| x_0^{(t)} \|_1 x_i^{(0)}
\]  

(7)

Thus, we replace the updating operation of \( x_i^{(t+1)} \) in Algorithm 1 (line 4) with \( \tilde{x}_i^{(t+1)} \). The details are shown in Algorithm 2. Intuitively, nodes close to the query node have higher scores than nodes far away. Thus, we utilize the breadth-first search (BFS) to expand \( x_0^{(t)} \) from the query node set until \( \| x_0^{(t)} \|_1 \geq \theta \) (lines 3-12). Then, in line 13, we approximate the score vector in the next iteration according to Eq. (7).

Let \( V_i^{(t)} \) be the set of nodes with positive values in \( x_i^{(t)} \), and \( |E_i^{(t)}| \) be the summation of out-degrees of nodes in \( V_i^{(t)} \). Then \( |E_i^{(t)}| \ll |E_i| \) which dramatically reduces the number of nodes to update in each iteration.

6 EXPERIMENTS

We perform comprehensive experimental studies to evaluate the effectiveness and efficiency of the proposed methods. Our algorithms are implemented with C++. The code and data used in this work are available.\(^3\) All experiments are conducted on a PC with Intel Core i7-6700 CPU and 32 GB memory, running 64 bit Windows 10.

6.1 Datasets and Baselines

Datasets. Six real-world datasets are used to evaluate the effectiveness of the selected methods. Statistics are summarized in Table 2. The first two are multiplex networks with the same node set.

- RM [13] has 10 networks, each with 91 nodes. Nodes represent phones and one edge exists if two phones detect each other under a mobile network. Each network describes connections between phones in a month. Phones are divided into two communities according to their owners’ affiliations.

- BrainNet [27] has 468 brain networks, one for each participant. In the brain network, nodes represent human brain regions and an edge depicts the functional association between two regions. Different participants may have different functional connections. Each network has 264 nodes, among which 177 nodes are studied to belong to 12 high-level functional systems, including auditory, memory retrieval, visual, etc. Each functional system is considered as a community.

The other four are general multiple networks with different types of nodes and edges and many-to-many cross-edges between nodes in different networks.

- 6-NG & 9-NG [22] are two multi-domain network datasets constructed from the 20-Newsgroup dataset\(^4\). 6-NG contains 5 networks of sizes \{600, 750, 900, 1050, 1200\} and 9-NG consists of 5 networks of sizes \{900, 1125, 1350, 1575, 1800\}. Nodes represent news documents and edges describe their semantic similarities. The cross-network relationships are measured by cosine similarity between two documents from two networks. Nodes in the five networks in 6-NG and 9-NG are selected from 6 and 9 news groups, respectively. Each news group is considered as a community.

- Citeseer [18] is from an academic search engine, Citeseer. It contains a researcher collaboration network, a paper citation network, and a paper similarity network. The collaboration network has 3,284 nodes (researchers) and 13,781 edges (collaborations). The paper citation network has 2,035 nodes (papers) and 3,356 edges (paper citations). The paper similarity network has 10,214 nodes (papers).

\(^3\)https://github.com/lyingdoog/RWM/

\(^4\)http://vpone.com/jason/20Newsgroups/
The state-of-the-art methods. We compare RWM with seven state-of-the-art local community detection methods. RWR [26] uses a lazy variation of random walk to rank nodes and sweeps the ranked nodes to detect local communities. MWC [5] uses the multiwalk chain model to measure node proximity scores. QDC [31] finds local communities by extracting query biased densest connected subgraphs. LEMON [17] is a local spectral approach. k-core [3] conducts graph core-decomposition and queries the community containing the query node. Note that these five methods are for single networks. The following two are for multiple networks. ML-LCD [12] uses a greedy strategy to find the local communities on multiplex networks with the same node set. MRWR [34] only focuses on the query network. Besides, without prior knowledge, MRWR treats other networks contributing equally.

For our method, RWM, we adopt two approximations and set \( \varepsilon = 0.01, \theta = 0.9 \). Restart factor \( \alpha \) and decay factor \( \lambda \) are searched from 0.1 to 0.9. Extensive parameter studies are conducted in Sec. 6.4. For baseline methods, we tune parameters according to their original papers and report the best results.

### 6.2 Effectiveness Evaluation

#### Evaluation on detected communities.

For each dataset, in each experiment trial, we randomly pick one node with label information from a network as the query. Our method RWM can detect all query-relevant communities from all networks, while all baseline methods can only detect one local community in the network containing the query node. Thus, in this section, to be fair, we only compare detected communities in the query network. In Sec. 6.5, we also verify that RWM can detect relevant and meaningful local communities from other networks.

Each experiment is repeated 1000 trials, and the Macro-F1 scores are reported in Table 3. Note that ML-LCD can only be applied to the multiplex network with the same node set.

From Table 3, we see that, in general, random walk based methods including RWR, MWC, MRWR, and RWM perform better than others. It demonstrates the advance of applying random walk for local community detection. Generally, the size of detected communities by QDC is very small, while that detected by ML-LCD is much larger than the ground truth. k-core suffers from finding a proper node core structure with a reasonable size, it either considers a very small number of nodes or the whole network as the detected result. Second, performances of methods for a single network, including RWR, MWC, QDC, and LEMON, are relatively low, since the single networks are noisy and incomplete. MRWR achieves the second best results on 6-NG, RM, and DBLP but performs worse than RWR and MWC on other datasets. Because not all networks provide equal and useful assistance for the detection, treating all networks equally in MRMR may introduce noises and decrease the performance. Third, our method RWM achieves the highest F1-scores on all datasets and outperforms the second best methods by 6.13% to 17.4%. This is because RWM can actively aggregate information from highly relevant local structures in other networks during updating visiting probabilities. We emphasize that only RWM can detect relevant local communities from other networks except for the network containing the query node. Please refer to Sec. 6.5 for details.

#### Ranking evaluation.

To gain further insight into why RWM outperforms others, we compare RWM with other random walk based methods, i.e., RWR, MWC, and MRWR, as follows. Intuitively, nodes in the target local community should be assigned with high proximity scores w.r.t. the query node. Then we check the precision of top-ranked nodes based on score vectors of those models, i.e., \( \text{prec} = (\text{top-k nodes } \cap \text{ground truth}) / k \).

The precision results are shown in Fig. 3. First, we can see that precision scores of the selected methods decrease when the size of detected community \( k \) increases. Second, our method consistently outperforms other random walk-based methods. Results indicate that RWM ranks nodes in the ground truth more accurately. Since ranking is the basis of a random walk based method for local community detection, better node-ranking generated by RWM leads to high-quality communities (Table 3).

### 6.3 Efficiency Evaluation

Note that the running time of RWM using basic power-iteration (Algorithm 1) is similar to the iteration-based random walk local community detection methods but RWM obtains better performance than other baselines with a large margin (Table 3). Thus, in this section, we focus on RWM and use synthetic datasets to evaluate its efficiency. There are three methods to update visiting probabilities in RWM: (1) power iteration method in Algorithm 1 (PowerIt), (2) power iteration with early stopping introduced in Sec. 5.1 (A1), and (3) partial updating in Sec. 5.2 (A2).

We first evaluate the running time w.r.t. the number of networks. We generate 9 datasets with different numbers of networks (2 to 10). For each dataset, we first use a graph generator [16] to generate a base network consisting of 1,000 nodes and about 7,000 edges. Then we obtain multiple networks from the base network. In each network, we randomly delete 50% edges from the base network. For each dataset, we randomly select a node as the query and

### Table 3: Accuracy performances comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>RM</th>
<th>BrainNet</th>
<th>6-NG</th>
<th>9-NG</th>
<th>Citeseer</th>
<th>DBLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>RWR</td>
<td>0.621</td>
<td>0.314</td>
<td>0.282</td>
<td>0.261</td>
<td>0.426</td>
<td>0.120</td>
</tr>
<tr>
<td>MWC</td>
<td>0.588</td>
<td>0.366</td>
<td>0.309</td>
<td>0.246</td>
<td>0.367</td>
<td>0.116</td>
</tr>
<tr>
<td>QDC</td>
<td>0.150</td>
<td>0.262</td>
<td>0.174</td>
<td>0.147</td>
<td>0.132</td>
<td>0.103</td>
</tr>
<tr>
<td>LEMON</td>
<td>0.637</td>
<td>0.266</td>
<td>0.336</td>
<td>0.250</td>
<td>0.107</td>
<td>0.090</td>
</tr>
<tr>
<td>k-core</td>
<td>0.599</td>
<td>0.189</td>
<td>0.283</td>
<td>0.200</td>
<td>0.333</td>
<td>0.052</td>
</tr>
<tr>
<td>MRWR</td>
<td>0.671</td>
<td>0.308</td>
<td>0.470</td>
<td>0.238</td>
<td>0.398</td>
<td>0.126</td>
</tr>
<tr>
<td>ML-LCD</td>
<td>0.361</td>
<td>0.171</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RWM (ours)</td>
<td><strong>0.732</strong></td>
<td><strong>0.403</strong></td>
<td><strong>0.552</strong></td>
<td><strong>0.277</strong></td>
<td><strong>0.478</strong></td>
<td><strong>0.133</strong></td>
</tr>
</tbody>
</table>

**Improve** 9.09% 10.11% 17.4% 6.13% 12.21% 5.56%
For each dataset, we generate multiplex networks with three
query node. Note that we compute
\( T \) w.r.t. \( \alpha \), \( \lambda \) and the end of updating (averaged over 100 trails).

In Table 4, we show the number of visited nodes both at the split-
tive visiting probability scores in RWM, in different sized networks.

Next, we evaluate the running time w.r.t. the number of nodes.
Similar to the aforementioned generating process, we synthesize 4
datasets from 4 base networks with the number of nodes ranging
from \( 10^4 \) to \( 10^7 \). The average node degree is 7 in each base
network. For each dataset, we generate multiplex networks with three
networks, by randomly removing half of the edges from the base
network. The running time is shown in Fig. 4(b). Comparing to
PowerIte. We see that in the end, only a very
many cases, however, the brain network generated from a single
Detecting and monitoring
local communities. In Fig. 4(a), we report the running time of
the three methods averaged over 100 trials. The early stopping
in A1 saves time by about 2 times for the iteration method. The
partial updating in A2 can further speed up by about 20 times.
Furthermore, we can observe that the running time of A2 grows
slower than PowerIte. Thus the efficiency gain tends to be larger
with more networks.

Next, we evaluate the running time w.r.t. the number of nodes.
Similar to the aforementioned generating process, we synthesize 4
datasets from 4 base networks with the number of nodes ranging
from \( 10^4 \) to \( 10^7 \). The average node degree is 7 in each base
network. For each dataset, we generate multiplex networks with three
networks, by randomly removing half of the edges from the base
network. The running time is shown in Fig. 4(b). Comparing to
power iteration method, the approximation methods are much
faster, especially when the number of nodes is large. In addition,
they grow much slower, which enables their scalability on even
larger datasets.

We further check the number of visited nodes, which have positive
visiting probability scores in RWM, in different sized networks.
In Table 4, we show the number of visited nodes both at the splitt-
ing time \( T_e \) and the end of updating (averaged over 100 trails).
Note that we compute \( T_e = \lfloor \log_{0.1}(1 - A) \rfloor \) according to Theorem 4
with \( K = 3, \varepsilon = 0.01, \lambda = 0.7 \). We see that in the end, only a very
small proportion of nodes are visited (for the biggest \( 10^7 \) network,
probability only propagates to 0.02% nodes). This demonstrates
the locality of RWM. Besides, in the first phase (early stop at \( T_e \)),
visiting probabilities are restricted in about 50 nodes around the
query node.

### 6.4 Parameter Study

In this section, we show the effects of the four important parameters
in RWM: the restart factor \( \alpha \), the decay factor \( \lambda \), tolerance \( \varepsilon \) for
early stopping, and the covering factor \( \theta \) in the partial updating.
We report the F1 scores and running time on two representative
datasets RM and 6-NG. Note that \( \varepsilon \) and \( \theta \) control the trade-off
between running time and accuracy.

The parameter \( \alpha \) controls the restart of a random walker in Eq.
(5). The F1 scores and the running time w.r.t. \( \alpha \) are shown in Fig. 5(a)
and Fig. 5(b). When \( \alpha \) is small, the accuracy increases as \( \alpha \) increases
because larger \( \alpha \) encourages further exploration. When \( \alpha \) reaches
an optimal value, the accuracy begins to drop slightly. Because a
too large \( \alpha \) impairs the locality property of the restart strategy.
The running time increases when \( \alpha \) increases because larger \( \alpha \) requires
more iterations for score vectors to converge.

\( \lambda \) controls the updating of the relevance weights in Eq. (6). Results
in Fig. 5(c) and Fig. 5(d) reflect that for RM, RWM achieves
the best result when \( \lambda = 0.9 \). This is because large \( \lambda \) ensures that
a enough number of neighbors are included when calculating relevance
similarities. For 6-NG, RWM achieves high accuracy in a
wide range of \( \lambda \). For the running time, according to Theorem 4,
larger \( \lambda \) results in larger \( T_e \), i.e., more iterations in the first phase,
and longer running time.

\( \varepsilon \) controls \( T_e \), the splitting time in the first phase. Instead of
adjusting \( \varepsilon \), we directly tune \( T_e \). Theoretically, a larger \( \gamma \) (i.e., a
smaller \( \varepsilon \)) leads to more accurate results. Based on results shown
in Fig. 5(e) and Fig. 5(f), we notice that RWM achieves good perfor-
ance even with a small \( T_e \) in the first phase. The running time
decreases significantly as well. This demonstrates the rationality
of early stopping.

\( \theta \) controls the number of updated nodes (in Sec. 5.2). In Fig. 5(g)
and 5(h), we see that the running time decreases along with \( \theta \) de-
creasing, because less number of nodes are updated. The consistent
accuracy performance shows the effectiveness of the speeding-up
strategy.

### 6.5 Case Studies

BrainNet and DBLP are two representative datasets for multiplex
networks (with the same node set) and the general multi-domain
network (with flexible nodes and edges). We do two case studies to
show the detected local communities by RWM.

**Case Study on the BrainNet Dataset.** Detecting and monitoring
functional systems in the human brain is an important task in
neuroscience. Brain networks can be built from neuroimages where
nodes and edges are brain regions and their functional relations. In
many cases, however, the brain network generated from a single
subject can be noisy and incomplete. Using brain networks from many subjects helps to identify functional systems more accurately. For example, brain networks from three subjects are shown in Fig. 6. Subjects 1 and 2 have similar visual conditions (red nodes); subjects 1 and 3 are with similar auditory conditions (blue nodes). For a given query region, we want to find related regions with the same functionality.

**Case Study on DBLP Dataset.** In the general multiple networks, for a query node from one network, we only have the ground truth local community in that network but no ground truth about the relevant local communities in other networks. So in this section, we use DBLP as a case study to demonstrate the relevance of local communities detected from other networks by RWM. We use Prof. Danai Koutra from UMich as the query. The DBLP dataset was collected in May 2014 when she was a Ph.D. student advised by

![Figure 5: Parameter study on RM and 6-NG](image)

![Figure 6: Brain networks of three subjects.](image)

![Figure 7: Similarity scores of walkers w.r.t. the iteration](image)

![Figure 8: Visualization of detected local community.](image)
Prof. Christos Faloutsos. Table 5 shows the detected author community and paper community. Due to the space limitation, instead of showing the details of the paper community, we list venues that these papers published in. For example, “KDD(3)” means that 3 KDD papers are included in the detected paper community. The table shows that our method can detect local communities from both networks with high qualities. Specifically, the detected authors are mainly from her advisor’s group. The detected paper communities are mainly published in Database/Data Mining conferences.

7 CONCLUSION
In this paper, we propose a novel random walk model, RWM, for local community detection in multiple networks. Unlike other baselines, RWM can detect all query-relevant local communities from all networks. Random walkers in different networks sent by RWM mutually affect their visiting probabilities in a reinforced manner. By aggregating their effects from query-relevant local subgraphs in different networks, RWM restricts walkers’ most visiting probabilities near the query nodes. Rigorous theoretical foundations are provided to verify the effectiveness of RWM. Two speeding-up strategies are also developed for efficient computation. Extensive experimental results verify the advantages of RWM in effectiveness and efficiency.

ACKNOWLEDGMENTS
This project was partially supported by NSF projects IIS-1707548 and CBET-1638320.

REFERENCES
[17] Yixuan Li, Kun He, David Bindel, and John E Hopcroft. 2015. Uncovering the small world of the Author community. Knowledge and Data Engineering 27, 3 (2015), 555–564.
[23] Yixuan Li, Kun He, David Bindel, and John E Hopcroft. 2015. Uncovering the small world of the Author community. Knowledge and Data Engineering 27, 3 (2015), 555–564.
[29] Yixuan Li, Kun He, David Bindel, and John E Hopcroft. 2015. Uncovering the small world of the Author community. Knowledge and Data Engineering 27, 3 (2015), 555–564.
Appendix of the KDD 2020 paper "Local Community Detection in Multiple Networks"

APPENDIX
The proof of Theorem 1

In Theorem 1, we discuss the weak-convergence of the modified transition matrix \( P_i^{(t)} \) in general multiple networks. Because after each iteration, \( P_i^{(t)} \) needs to be column-normalized to keep the stochastic property, we let \( \tilde{P}_i^{(t)} \) represent the column-normalized one. Then Theorem 1 is for the residual \( \Delta(t+1) = \| P_i^{(t+1)} - \tilde{P}_i^{(t)} \|_\infty \). In the proof, we will utilize some results in the proof of Theorem 2.

Proof.

\[
\begin{align*}
\Delta(t+1) &= \| P_i^{(t+1)} - \tilde{P}_i^{(t)} \|_\infty \\
&= \max \{ \| P_i^{(t+1)}(x,y) - \tilde{P}_i^{(t)}(x,y) \| \} \\
&= \max \{ \| \sum_{z} P_i^{(t+1)}(z,y) - \tilde{P}_i^{(t)}(z,y) \| \}
\end{align*}
\]

Based on Eq. (2), we have for all \( t, \frac{1}{K} \leq \sum_{z} P_i^{(t)}(z,y) \leq 1 \).

We denote \( \min \{ \sum_{z} P_i^{(t+1)}(z,y), \sum_{z} P_i^{(t)}(z,y) \} \) as \( p \) and \( \min \{ \sum_{z} P_i^{(t+1)}(z,y), \sum_{z} P_i^{(t)}(z,y) \} \) as \( m \). From the proof of Theorem 2, we know that

\[
\frac{\| P_i^{(t+1)}(x,y) - \tilde{P}_i^{(t)}(x,y) \|}{\| P_i^{(t)}(x,y) \|} \leq \lambda^t K
\]

and

\[
\frac{1}{m} \sum_{z} P_i^{(t+1)}(z,y) - \frac{1}{m} \sum_{z} P_i^{(t)}(z,y) \leq \lambda^t K \frac{1}{m} \left| V_i \right|
\]

Then, we have

\[
\frac{1}{m} \sum_{z} P_i^{(t+1)}(z,y) - \frac{1}{m} \sum_{z} P_i^{(t)}(z,y) \leq \frac{p + \lambda^t K \frac{1}{m} \left| V_i \right|}{m} \leq \lambda^t \frac{m K \frac{1}{m} \left| V_i \right|}{m} \leq \lambda^t \left| V_i \right|
\]

When \( t > \left[ \log_\lambda \left( K^2 \frac{\left| V_i \right|}{m} \right) \right] \), we have \( \lambda^t K^2 \frac{1}{m} \left| V_i \right| \leq 1 \). Thus,

\[
\Delta(t+1) \leq \lambda^t \frac{m K \frac{1}{m} \left| V_i \right|}{m} \leq \lambda^t K \frac{1}{m} \left| V_i \right| + 2
\]

Then, it’s derived that \( \Delta(t+1) \leq \epsilon \) when \( t > \left[ \log_\lambda \left( K^2 \frac{\left| V_i \right|}{m} \right) \right] \).

The proof of Theorem 2

Theorem 2 discusses the weak-convergence of the modified transition matrix \( P_i^{(t)} \) in the special multiplex networks which have the same node set in all networks. In the multiplex networks, cross-transition matrices are just I, so the stochastic property of \( P_i^{(t)} \) can be naturally guaranteed without the column-normalization of \( P_i^{(t)} \).

We then define \( \Delta(t+1) = \| P_i^{(t+1)} - P_i^{(t)} \|_\infty \).

Proof. Based on Eq. (3) and Eq. (2), we have

\[
\begin{align*}
\Delta(t+1) &= \| P_i^{(t+1)} - P_i^{(t)} \|_\infty \\
&= \| \sum_{j=0}^{K} | W^{(t+1)}(i,j) - W^{(t)}(i,j) | P_j \|_\infty \\
&= \| \sum_{j \in L_i} | W^{(t+1)}(i,j) - W^{(t)}(i,j) | P_j \|_\infty \\
&\quad + \sum_{j \in L_i} | W^{(t+1)}(i,j) - W^{(t)}(i,j) | P_j \|_\infty \\
&= \| \sum_{j \in L_i} | W^{(t+1)}(i,j) - W^{(t)}(i,j) | P_j \|_\infty \\
&= \| \sum_{j \in L_i} | W^{(t+1)}(i,j) - W^{(t)}(i,j) | P_j \|_\infty \\
&\quad + \sum_{j \in L_i} | W^{(t+1)}(i,j) - W^{(t)}(i,j) | P_j \|_\infty \\
&\quad + \sum_{j \in L_i} | W^{(t+1)}(i,j) - W^{(t)}(i,j) | P_j \|_\infty
\end{align*}
\]

where \( L_i \) = \{ j | W^{(t+1)}(i,j) \geq W^{(t)}(i,j) \}, and \( L_i = \{ 1, 2, ..., K \} - L_i \).

Since all entries in \( P_j \) are non-negative, for all \( j \), all entries in the first part is non-negative and all entries in the second part is non-positive. Thus, we have

\[
\begin{align*}
\Delta(t+1) &= \max \{ \| \sum_{j \in L_i} | W^{(t+1)}(i,j) - W^{(t)}(i,j) | P_j \|_\infty, \\
&\quad \| \sum_{j \in L_i} | W^{(t+1)}(i,j) - W^{(t)}(i,j) | P_j \|_\infty \}
\end{align*}
\]

Since for all \( i \), score vector \( x_i^{(t)} \) is non-negative, we have \( \cos(x_i^{(t)}, x_i^{(t+1)}) \geq 0 \). Thus, \( \sum_k W^{(t)}(i,k) \geq \sum_k W^{(t)}(i,k) \). For the first part, we have

\[
\begin{align*}
\| \sum_{j \in L_i} | W^{(t+1)}(i,j) - W^{(t)}(i,j) | P_j \|_\infty \\
&= \| \sum_{j \in L_i} W^{(t+1)}(i,j) - \sum_{j \in L_i} W^{(t)}(i,j) | P_j \|_\infty \\
&\leq \| \sum_{j \in L_i} W^{(t+1)}(i,j) + \lambda^{t+1} | P_j \|_\infty \\
&\leq \| \sum_{j \in L_i} W^{(t+1)}(i,j) | P_j \|_\infty \\
&\leq \| \sum_{j \in L_i} W^{(t+1)}(i,j) | P_j \|_\infty
\end{align*}
\]

Similarly, we can prove the second part has the same bound.

\[
\begin{align*}
\| \sum_{j \in L_i} | W^{(t+1)}(i,j) - W^{(t)}(i,j) | P_j \|_\infty \\
&= \| \sum_{j \in L_i} W^{(t+1)}(i,j) - \sum_{j \in L_i} W^{(t)}(i,j) | P_j \|_\infty \\
&\leq \| \sum_{j \in L_i} W^{(t+1)}(i,j) + \lambda^{t+1} | P_j \|_\infty \\
&\leq \| \sum_{j \in L_i} W^{(t+1)}(i,j) | P_j \|_\infty \\
&\leq \| \sum_{j \in L_i} W^{(t+1)}(i,j) | P_j \|_\infty
\end{align*}
\]

Then, we know \( \Delta(t+1) \leq \epsilon \) when \( t > \left[ \log_\lambda \left( \frac{\epsilon}{K} \right) \right] \).
The proof of Theorem 4

Proof. According to the proof of Theorem 1, when $T_e > \lceil -\log_\lambda (K^2 |V_i|) \rceil$, we have

$$\|P_i(\infty) - P_i(T_e)\|_\infty \leq \|\sum_{t=0}^\infty \|P_i^{(t+1)} - P_i^{(t)}\|_\infty \leq \sum_{t=0}^\infty \lambda^t K^2 (|V_i| + 2) \leq \frac{\lambda T_e K^2 (|V_i| + 2)}{1 - \lambda}$$

So we can select $T_e = \lceil \log_\lambda \frac{\epsilon (1-\lambda)}{K^2 (|V_i| + 2)} \rceil$ such that when $t > T_e$, $\|P_i^{(\infty)} - P_i^{(t)}\|_\infty < \epsilon$.

□