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
yy230@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\(^\text{1}\) 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 \(^\text{2}\).

\(^{1}\)We use graph and network interchangeably in this paper.

\(^{2}\)Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD ’20, August 23–27, 2020, Virtual Event, USA
© 2020 Association for Computing Machinery.
https://doi.org/10.1145/3394486.3403069

Figure 1: An example of multiple social networks with the same node set.

Figure 2: An example of local community detection in general multiple networks.

(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 topologies in corresponding networks are relevant. And these two walkers should reinforce each other with highly weighted influences. On the other hand, if visited local topologies 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 [3] 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 node 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>$E_{i,j}$</td>
<td>The cross-edge set between network $G_i$ and $G_j$</td>
</tr>
<tr>
<td>$S_{i,j}$</td>
<td>Column-norm-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>$x(t)$</td>
<td>Node visit. prob. vec. of the $t$th walker in $G_i$ at time $t$</td>
</tr>
<tr>
<td>$W(t)$</td>
<td>Relevance weight matrix at time $t$</td>
</tr>
<tr>
<td>$\mathcal{P}_i(t)$</td>
<td>Modified trans. mat. for the $t$th walker in $G_i$ at time $t$</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 network $(1 \leq i \leq K)$ 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 $(u,v)$th entry $P_{ij}(u,v)$ represents the transition probability from node $u$ to node $v$ in $G_i$. Then the $u$th column $P_i(:, u)$ is the transition distribution from node $u$ to all nodes in $V_i$. Next, we denote $E_{i,j}$ the cross-connections between nodes in two networks $V_i$ and $V_j$. The corresponding cross-transition matrix is $S_{i,j} \in \mathbb{R}^{|V_i| \times |V_j|}$. Then the $u$th column $S_{i,j}(:, u)$ is the transition distribution from node $u$ to all nodes in $V_j$. Note that $S_{i,j} \neq S_{j,i}$. And for the multiplex networks with the same node set (e.g., Fig. 1), $S_{i,j}$ is just an identity matrix $I$ for arbitrary $i,j$.

Suppose we send a random walker in $G_i$, we let $x_i(t)$ be the node visiting probability vector in $G_i$ at time $t$. Then the updated vector $x_i(t+1) = P_i x_i(t)$ means the probability transiting among nodes in $G_i$. And $S_{i,j} x_j(t)$ 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 $x_q(0) = e_q$ 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 $x_i(0) = S_{q,i} x_q(0)$. That is:

$$x_i(0) = \begin{cases} e_q & \text{if } i = q \\ S_{q,i} e_q & \text{otherwise} \end{cases}$$ (1)

To update $i$th walker’s vector $x_i(t)$, the walker not only follows the transition probabilities in the corresponding network $G_i$, but also obtains influences from other networks. Intuitively, networks share 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 $x_i(t)$ and $x_j(t)$. Since different networks consist of different node sets, we define the relevance as $\cos(x_i(t), S_{j,i} x_j(t))$. 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 5, $\lambda$ can guarantee and control the convergence of RWM model. Formally, we let $W(t) \in \mathbb{R}^{K \times K}$ be the local relevance matrix among networks at time $t$ and we initialize it with identity matrix $I$. We update each entry $W(t)(i,j)$ as follows:

$$W(t)(i,j) = W(t-1)(i,j) + \lambda \cos(x_i(t), S_{j,i} x_j(t))$$ (2)

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:

$$\mathcal{P}_i(t) = \sum_{j=1}^{K} W(t)(i,j) S_{j,i} P_i S_{i,j}$$ (3)

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

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

$$x_i(t+1) = \mathcal{P}_i(t) x_i(t)$$ (4)

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 $\mathcal{P}_i(t)$. 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 $0 < \epsilon < 1$, for all $i$, when $t > \lceil \frac{1}{\lambda \epsilon \|S_{i,j}\|} \rceil$, $\|\mathcal{P}_i(t+1) - \mathcal{P}_i(t)\|_\infty < \epsilon$, where $V_i$ is the node set of network $G_i$ and $\| \cdot \|_\infty$ is the $\infty$-norm of a matrix.

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

---

1For simplicity, the illustration is for one query node. We can easily modify our model for a set of query nodes by initializing the visiting vector $x_q(0)$ with uniform entry value $1/n$ if there are $n$ query nodes.

2If there are no direct connections from $u_q$ to nodes in $V_i (i \neq q)$, we first propagate probability to other nodes in $G_q$ from $u_q$ via breadth-first-search layer by layer until we reach a node which has cross-edges to nodes in $V_i$. Then we initialize $x_i(0) = S_{q,i} P_i e_q$, where $t$ is the number of hops that we first reach the effective node from $u_q$ in $G_q$.

3http://personal.psu.edu/dul62/RWM/appendix.pdf
Theorem 2. When applying RWM in multiple networks with the same node set, for any small tolerance $0 < \varepsilon < 1$, for all $i$, \[ \| P_i^{(t+1)} - P_i^{(t)} \|_\infty < \varepsilon, \text{ when } t > \lceil \log_\lambda \frac{1}{\varepsilon} \rceil. \]

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 $\alpha(0 < \alpha < 1)$ probability and jumps back to the query node with probability $1 - \alpha$. 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:

\[ x_i^{(t+1)} = \alpha \hat{P}_i^{(t)} x_i^{(t)} + (1 - \alpha)x_i^{(0)} \]

where $\hat{P}_i^{(t)}$ 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 cost $\cdot$ in Eq. (2) and have:

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

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 $\hat{P}_i^{(t)}$ ($\lambda$ has the same effect as in Theorem 1). After obtaining a converged $\hat{P}_i^{(t)}$, 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 $l$ 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 $i^{th}$ walkers (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 $\hat{P}_i^{(t)}$ and output. If we substitute Eq. (3) in Eq. (5), we have a probability propagation $S_{j \rightarrow i}(P_j S_{i \rightarrow j} x_i^{(t)})$ 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 $S_{j \rightarrow i}(P_j S_{i \rightarrow j} x_i^{(t)})$ costs $O(|V_i| + |E_{i \rightarrow j}| + |E_j|)$. And the restart addition in Eq. (5) costs $O(|E_i|)$. As a result, line 4 costs $O(|V_i| + |E_i| + \sum_{j \in \delta i}(|E_{i \rightarrow j}| + |E_j|))$ where $O(|E_i|)$ is from the propagation in $G_i$ itself.

In line 5, based on Eq. (6), it takes $O(|E_{i \rightarrow j}| + |V_j|)$ to get $S_{j \rightarrow i}(x_i^{(t)})$ and $O(|V_j|)$ to compute cosine similarities. Then updating $W(i,j)$ (line 5) costs $O(|E_{i \rightarrow j}| + |V_i| + |V_j|)$. 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 $O((\sum_i |V_i| + \sum_i |E_i| + \sum_{j \in \delta i} |E_{i \rightarrow j}|)KT)$ 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 $E_{i \rightarrow j}$.

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 $P_i^{(\infty)}$ the converged 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 $\varepsilon$, when $t > T_e = \lceil \log_\lambda \frac{\varepsilon}{K\delta (|V_i| + 2)} \rceil$, $\| P_i^{(\infty)} - P_i^{(T_e)} \|_\infty < \varepsilon$. For the multiplex networks...

**Algorithm 1: Random walk in multiple networks**

**Input:** Transition matrices $P_i$, cross-transition matrices $\{S_{i \rightarrow j}\}_{i,j}$, tolerance $\varepsilon$, query node $u_q \in G_q$, decay factor $\lambda$, restart factor $\alpha$, iteration number $T$.

**Output:** Visiting prob. vec. $K$ walkers ($P_i^{(T_e)}$).

1. Initialize $x_i^{(0)}$ based on Eq. (1), and $W^{(0)} = \hat{W}^{(0)} = I$.
2. For $t = 0$ to $T$
3. For $i \leftarrow 1$ to $K$
4. Calculate $x_i^{(t+1)}$ according to Eq. (3) and (5);
5. Calculate $W^{(t+1)}$ according to Eq. (6);
6. $W^{(t+1)} \rightarrow$ row-normalize $W^{(t+1)}$;
with the same node set, we can choose $T_\epsilon = \lceil \log_\delta \frac{e^{(1-\lambda)} - \epsilon}{\lambda} \rceil$ to get the same estimation bound $\epsilon$.

Please refer to Appendix for the proof.

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

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

5.2 Partially 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 $i^{th}$ iteration, we separate $x_i^{(t)}$ into two non-negative vectors, $x_i^{(t)}$ and $\tilde{x}_i^{(t)}$, so that $x_i^{(t)} = x_i^{(t)} + \tilde{x}_i^{(t)}$, and $\|x_i^{(t)}\|_1 \geq \theta$. Then, we approximate $x_i^{(t+1)}$ with $x_i^{(t+1)} = \alpha P_i^{(t)} x_i^{(t)} + (1 - \alpha) \|x_i^{(t)}\|_1 x_i^{(0)} \geq \theta$ (lines 3-12). Then, in line 13, we approximate the score vector in the next iteration according to Eq. (7).

Thus, we replace the updating operation of $x_i^{(t+1)}$ in Algorithm 1 (line 4) with $x_i^{(t+1)}$. 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 $x_i^{(t)}$ from the query node set until $\|x_i^{(t)}\|_1 \geq \theta$ (lines 3-12). Then, in line 13, we approximate the score vector in the next iteration according to Eq. (7).

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.

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.

<table>
<thead>
<tr>
<th>Table 2: Statistics of datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
</tr>
<tr>
<td>RM</td>
</tr>
<tr>
<td>BrainNet</td>
</tr>
<tr>
<td>6-NG</td>
</tr>
<tr>
<td>9-NG</td>
</tr>
<tr>
<td>Citeeseer</td>
</tr>
<tr>
<td>DBLP</td>
</tr>
</tbody>
</table>

5. https://github.com/lyingsdog/RWM/
citation network. The collaboration network has 1,209,164 nodes and 4,532,273 edges. The citation network consists of 2,150,157 papers connected by 4,191,677 citations. These two networks are connected by 5,851,893 author-paper edges. From one venue, we form an author community by extracting the ones who published more than 3 papers in that venue. We select communities with sizes ranging from 5 to 200, leading to 2,373 communities.

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 multi-walk 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 $\epsilon = 0.01, \theta = 0.9$. 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 MMRM 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., prec $= (\text{top-} k \text{ nodes } \cap \text{ground truth})/k$

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) partially 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 detect 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 partially updating in A2 can further speed up by about 20 times. Furthermore, we can observe that the running time of A2 grows.
We report the F1 scores and running time on two representative datasets RM and 6-NG. Note that \( \epsilon \) 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 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.

\( \epsilon \) controls \( T_e \), the splitting time in the first phase. Instead of adjusting \( \epsilon \), we directly tune \( T_e \). Theoretically, a larger \( T_e \) (i.e., a smaller \( \epsilon \)) leads to more accurate results. Based on results shown in Fig. 5(e) and Fig. 5(f), we notice that RWM achieves good performance 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 \( \theta \) decreasing, 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.

**Detect relevant networks.** To see whether RWM can automatically detect relevant networks, we run RWM model for Query 1 and Query 2 in Fig. 6 separately. Fig. 7 shows the cosine similarity between the visiting probability vectors of different walkers along iterations. $x_1$, $x_2$, and $x_3$ are the three visiting vectors on the three brain networks, respectively. We see that the similarity between the visiting histories of walkers in relevant networks, i.e., subjects 1 and 2 in Fig. 7(a), subjects 1 and 3 in 7(b), increases along with the updating. But similarities in query-oriented irrelevant networks are very low during the whole process. This indicates that RWM can actively select query-oriented relevant networks to help better capture the local structure for each network.

**Identify functional systems.** In this case study, we further evaluate the detected results of RWM in BrainNet dataset with the other two methods. Note that MRWR and ML-LCD can only find a query-oriented local community in the network containing the query node.

Fig. 8(a) shows a brain network of a subject with a given query node and its ground truth local community (red nodes). We run the three methods aggregating information from all other brain networks in the dataset. The identified communities are highlighted in red in Fig. 8. It’s shown that the community detected by our method is very similar to the ground truth. MRWR (Fig. 8(b)) includes many false-positive nodes. The reason is that MRWR assumes all networks are similar and treat them equally. While ML-LCD (Fig. 8(d)) neglects some nodes in the boundary region.

**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 is collected in May 2014 when she was a Ph.D. student advised by 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

APPENDIX

The proof of Theorem 1

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

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

$$\Delta(t+1)=\|\hat{\mathcal{P}}_i^{(t+1)}-\hat{\mathcal{P}}_i^{(t)}\|_\infty$$

$$=\max\{\|\hat{\mathcal{P}}_i^{(t+1)}(x,y)-\hat{\mathcal{P}}_i^{(t)}(x,y)\|\}$$

$$=\max\{|\sum_z \hat{\mathcal{P}}_i^{(t+1)}(z,y)-\sum_z \hat{\mathcal{P}}_i^{(t)}(z,y)|\}$$

Based on Eq. (2), we have for all $t$, $\frac{1}{2} \leq \sum_z \hat{\mathcal{P}}_i^{(t)}(z,y) \leq 1$. We denote $\max\{\sum_z \hat{\mathcal{P}}_i^{(t+1)}(z,y),\sum_z \hat{\mathcal{P}}_i^{(t)}(z,y)\}$ as $p$ and $\min\{\sum_z \hat{\mathcal{P}}_i^{(t+1)}(z,y),\sum_z \hat{\mathcal{P}}_i^{(t)}(z,y)\}$ as $m$. From the proof of Theorem 2, we know that

$$|\sum_z \hat{\mathcal{P}}_i^{(t+1)}(z,y)-\sum_z \hat{\mathcal{P}}_i^{(t)}(z,y)| \leq \lambda^t K |V_i|$$

Then, we have

$$|\sum_z \hat{\mathcal{P}}_i^{(t+1)}(z,y)-\sum_z \hat{\mathcal{P}}_i^{(t)}(z,y)| \leq \frac{p+\lambda^t K}{m} - \frac{p}{m + \lambda^t K |V_i|}$$

$$\leq m \lambda^t K + |V_i| (\lambda^t K)^2 + p K |V_i|$$

When $t > -\log_\lambda (K^2 |V_i|)$, we have $\lambda^t K^2 |V_i| \leq 1$. Thus,

$$\Delta(t+1) \leq \lambda^t m K + K^2 |V_i| \lambda^t + p K |V_i|$$

$$\leq \lambda^t m K + K^2 |V_i| \lambda^t + m K |V_i|$$

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

The proof of Theorem 2

Theorem 2 discusses the weak-convergence of the modified transition matrix $\mathcal{P}_i^{(t)}$ in the special multiplex networks which have the same node set in all networks. In the multiplex networks, cross-transition matrices are just 1, so the stochastic property of $\mathcal{P}_i^{(t)}$ can be naturally guaranteed without the column-normalization of $\mathcal{P}_i^{(t)}$. We then define $\Delta(t+1)=\|\mathcal{P}_i^{(t+1)}-\mathcal{P}_i^{(t)}\|_\infty$.

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

$$\Delta(t+1)=\|\mathcal{P}_i^{(t+1)}-\mathcal{P}_i^{(t)}\|_\infty$$

$$=\max_{j \in L_i} \{\|\mathcal{P}_j^{(t+1)}(i,j) - \mathcal{P}_j^{(t)}(i,j)\|\}$$

$$=\max_{j \in L_i} \{\|\mathcal{P}_j^{(t+1)}(i,j) - \mathcal{P}_j^{(t)}(i,j)\|\}$$

where $L_i = \{ j | \mathcal{P}_j(i,j) > 0 \}$, and $L_i = \{ 1, 2, ..., K \} - L_i$. Since all entries in $\mathcal{P}_j$ are non-negative, all entries in the first part is non-negative and all entries in the second part is non-positive. Thus, we have

$$\Delta(t+1) = \max\{\sum_{j \in L_i} \|\mathcal{P}_j^{(t+1)}(i,j) - \mathcal{P}_j^{(t)}(i,j)\|\}$$

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

$$\sum_{j \in L_i} \{\|\mathcal{P}_j^{(t+1)}(i,j) - \mathcal{P}_j^{(t)}(i,j)\|\}$$

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

$$\sum_{j \in L_i} \{\|\mathcal{P}_j^{(t+1)}(i,j) - \mathcal{P}_j^{(t)}(i,j)\|\}$$

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

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

\[
\| P_i^{(\infty)} - P_i^{(T_\epsilon)} \|_\infty \\
\leq \| \sum_{t=T_\epsilon}^{\infty} P_i^{(t+1)} - P_i^{(t)} \|_\infty \\
\leq \sum_{t=T_\epsilon}^{\infty} \lambda^t K^2 (|V_i| + 2) \\
= \frac{\lambda T_\epsilon K^2 (|V_i| + 2)}{1 - \lambda}
\]

So we can select \( T_\epsilon = \lceil \log_\lambda \frac{\epsilon (1-\lambda)}{K^2 |V_i| + 2} \rceil \) such that when \( t > T_\epsilon \),

\[ \| P_i^{(\infty)} - P_i^{(t)} \|_\infty < \epsilon. \]

\( \square \)