




The multi-walker chain and its application in local community detection

Yuchen Bian¹  · Jingchao Ni¹ · Wei Cheng² · Xiang Zhang¹

Received: 15 December 2017 / Revised: 19 May 2018 / Accepted: 26 May 2018 / Published online: 4 October 2018
© Springer-Verlag London Ltd., part of Springer Nature 2018

Abstract

Local community detection (or local clustering) is of fundamental importance in large network analysis. Random walk-based methods have been routinely used in this task. Most existing random walk methods are based on the single-walker model. However, without any guidance, a single walker may not be adequate to effectively capture the local cluster. In this paper, we study a multi-walker chain (MWC) model, which allows multiple walkers to explore the network. Each walker is influenced (or pulled back) by all other walkers when deciding the next steps. This helps the walkers to stay as a group and within the cluster. We introduce two measures based on the mean and standard deviation of the visiting probabilities of the walkers. These measures not only can accurately identify the local cluster, but also help detect the cluster center and boundary, which cannot be achieved by the existing single-walker methods. We provide rigorous theoretical foundation for MWC and devise efficient algorithms to compute it. Extensive experimental results on a variety of real-world and synthetic networks demonstrate that MWC outperforms the state-of-the-art local community detection methods by a large margin.

Keywords Local community detection · Local clustering · Random walk · Multi-walker chain

✉ Yuchen Bian
yub31@ist.psu.edu
Jingchao Ni
jzn47@ist.psu.edu
Wei Cheng
weicheng@nec-labs.com
Xiang Zhang
xzhang@ist.psu.edu

¹ College of Information Sciences and Technology, The Pennsylvania State University, University Park, PA 16802, USA

² NEC Laboratories America, Princeton, USA

1 Introduction

Local community detection (or local clustering¹) is of fundamental importance in large network analysis. The goal is to find a cluster that contains the query node. Random walk-based methods are routinely used in this task [3,17,33].

Most of the existing random walk methods are based on the single-walker model. The basic idea is to allow a walker to randomly explore the network. A node having a high visiting probability is considered to have high proximity to the query node thus should be included in the local cluster. However, a single walker may not be able to capture the local clustering structure effectively.

For illustration and comparison purposes, we use random walk with restart (RWR) [24,28] as the representative of the single-walker models. The reason for choosing RWR as the reference is that it is one of the most widely used methods and has recently been shown to be the most effective in finding local clusters [17].

Figure 1 shows the results of RWR on an example network for different query nodes. The nodes are colored according to their stationary probabilities generated by RWR. A darker color represents a higher visiting probability. We can make a few key observations from the figure. First, there is no significant difference between the nodes inside and outside the local cluster (other than a few nodes near the query). Second, the identified local cluster is usually centered around the query node. Thus, the accuracy of the local clustering results heavily depends on the choice of the query node: a node in the center of the cluster might be a good query node, while using a non-center node as the query can significantly bias the result [17]. Third, as shown in Fig. 1c, when the walker reaches a node on the boundary of the cluster, there is no mechanism to prevent her from walking out of it. Note that similar observations can also be made for other existing methods.

To address the limitations of the single-walker methods, in this paper, we propose a multi-walker chain (MWC) model, which utilizes a group of walkers to explore the graph. The walkers in MWC follow a sequential order to walk in the graph. When deciding the next steps, each walker is pulled back by all other walkers so that the walkers will stay together as a group. In particular, when a walker reaches the cluster boundary, she is less likely to walk out of it because of the influence from other walkers. Thus, the entire group will be trapped in the local cluster.

Figure 2 shows the results by applying MWC with a group of 5 walkers on the example network. The nodes are colored according to the mean visiting probabilities of the walkers. As we can see from the figure, the difference between the nodes inside and outside the local cluster is clear. Moreover, MWC is robust to the choice of the query nodes. With different query nodes, MWC returns consistent clustering results. Even when a node on the cluster boundary is used as the query, MWC can still accurately find the local cluster as shown in Fig. 2b, c. This demonstrates that the influence from other walkers help the entire group to stay within the cluster.

In this paper, we introduce MWC and provide the rigorous theoretical foundation for it. We study its convergence property and introduce two score vectors based on the mean and standard deviation of the visiting probabilities of the walkers. These two score vectors provide accurate and rich information about the local cluster. Theoretically, the classic RWR can be treated as a special case of MWC. In RWR, the single walker is always pulled back (or influenced) by the query node. In MWC, each walker is influenced by the current positions of all other walkers.

¹ We use local clustering and local community detection interchangeably in this paper.

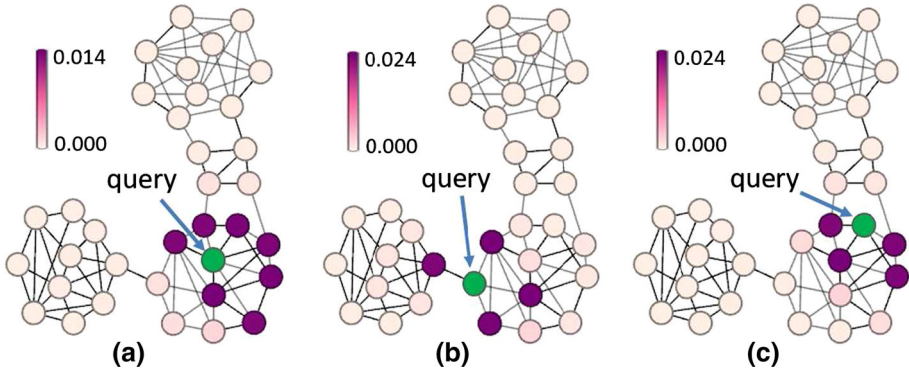


Fig. 1 Results of RWR for different query nodes

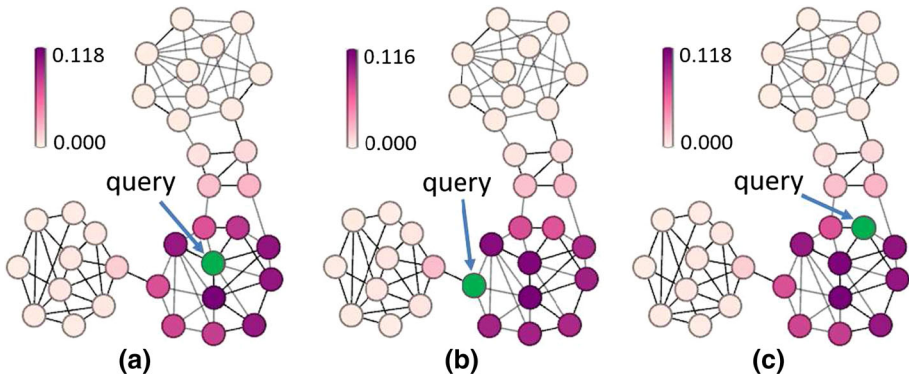


Fig. 2 Results of MWC for different query nodes

Compared to the existing single-walker models, MWC is more accurate and robust to the choice of the query nodes. The influence from other walkers can effectively prevent any walker from walking out of the cluster. Furthermore, the center nodes of the local cluster tend to have the highest scores, as can be observed in Fig. 2. On the contrary, in the single-walker model, the query node usually gets the highest score, and the identified local cluster is centered around the query node. Another unique advantage of MWC is that by examining the dispersion between the visiting probabilities of different walkers, we can identify boundary nodes of the local cluster. The intuition is that when a walker reaches a boundary node, she will hesitate whether to stay in or walk out of the cluster. Thus, the discrepancy between the visiting probabilities of different walkers will be high for the boundary nodes.

We develop efficient algorithms to compute MWC. In particular, our method allows to update only a very small subset of nodes while still achieves high accuracy. For example, by updating less than 0.02% of the nodes, we can achieve 0.98 correlation between the estimated and exact score vectors. We perform extensive experimental studies on a variety of real-world and synthetic networks to evaluate the performance of the proposed MWC method. The results show that MWC is more robust and achieves higher accuracy compared to the state-of-the-art methods.

2 Related work

Finding the local cluster for a given query node has recently attracted intensive research interests due to its practical importance [3,4,16,17,22,27,33]. Random walks have been routinely used to address this problem.

Single random walk models The classic PageRank–Nibble algorithm [3,4] first performs a lazy random walk and ranks the nodes according to the degree-normalized visiting probabilities. It then scans the ranking list to find the subset of top-ranked nodes that minimizes the conductance as the local cluster. The query-biased local clustering method [33] uses the visiting probabilities generated by random walks to assign weights to the nodes so that the densest subgraph shifts to the neighborhood of the query node. The query-biased densest subgraph is then identified as the local cluster. In [29], the authors studied various strategies to choose the proper query node to improve the clustering accuracy. Random walk is then used to find the local community that provides the optimal conductance. Comprehensive experimental evaluations have been conducted in [17] and the results suggest that RWR is the most effective in finding the local clusters. The recently proposed local spectral method uses short random walks to estimate the invariant subspace (local spectra) and finds the local community in the subspace [12,13,22]. The heat kernel method [16] uses the heat diffusion model to replace RWR in the PageRank–Nibble algorithm to find the local cluster. Heat kernel can be treated as a generalization of RWR. Random walk models are also used to detect motif patterns (small subgraphs) around the query node [35,38]. These methods extend the basic definition of conductance to high-order or motif conductance and iteratively detect a subgraph with small high-order conductance from a given query node. Note that all these methods are based on the single-walker model.

Multiple random walkers models Very limited work has been done on multi-walker models to detect local community. In [1], a multi-agent method was developed for local clustering. Each pair of agents is bounded by a rope with fixed length. The high space and time complexities [both are $O(n^{2a})$, where n is the number of nodes and a is the number of agents] render this approach infeasible for large networks.

In [20], the authors developed a double-walker model for bilayer image segmentation. One walker is used as the background walker and another as the foreground walker. The repulsive restarting rule restricts each walker to stay only in one segment. The authors later extended this approach for RGB-D image segmentation [21]. Some theoretical research has been done to speed up the cover time of multiple random walkers without targeting any specific application [2,7].

3 Multi-walker chain

In this section, we introduce the MWC model and study its theoretical foundation.

3.1 Preliminaries

Given an undirected and connected graph G , let \mathbf{W} be its weighted adjacent matrix, where $\mathbf{W}(i, j)$ is the weight on edge (i, j) , and \mathbf{P} be its transition matrix, where $\mathbf{P}(i, j)$ is the transition probability from node i to j , i.e., $\mathbf{P}(i, j) = \mathbf{W}(i, j) / \sum_{j \in V} \mathbf{W}(i, j)$.

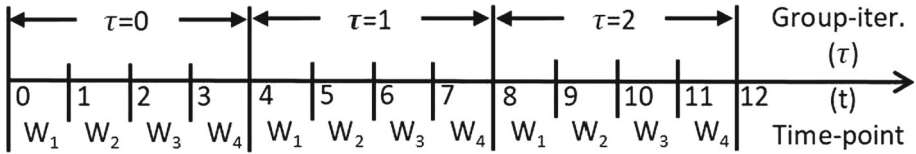


Fig. 3 MWC with 4 walkers

We first briefly review RWR. In RWR, a single walker randomly walks in the graph starting from a query node q . From time point t to $(t + 1)$, the walker has a probability of α to follow the transition probabilities in \mathbf{P} , and a probability of $(1 - \alpha)$ to jump back to q . In RWR, we have

$$\mathbf{x}^{(t+1)} = \alpha \mathbf{P}^T \mathbf{x}^{(t)} + (1 - \alpha) \mathbf{v} \tag{1}$$

where $\mathbf{x}^{(t)}$ is the node visiting probability column vector at time t , and \mathbf{v} is a column vector with value 1 for the q th entry and 0 for all other entries. Intuitively, \mathbf{v} represents the *influence* from the query to the walker, that is, the walk has $(1 - \alpha)$ probability to be influenced by q and jump to it.

3.2 The MWC model

In MWC, we have a group of K walkers $\{W_1, \dots, W_K\}$ ($K > 1$). These walkers take turns to walk in the graph. When all walkers finish one iteration, another iteration begins. Figure 3 shows an example of 4 walkers. We use τ ($\tau \geq 0$) as the index of the group iterations. In Fig. 3, there are three group iterations.

In each group iteration, the node visiting probability vector of W_K is only updated once. Let $\mathbf{x}_k^{(\tau)}$ be the updated node visiting probability vector of W_k in the τ th group iteration. When $\tau = 0$, $\mathbf{x}_k^{(\tau)}$ is initialized so that the entry corresponding to the query node is 1 and all other entries are 0. For $\tau \geq 1$, we have

$$\mathbf{x}_k^{(\tau+1)} = \alpha \mathbf{P}^T \mathbf{x}_k^{(\tau)} + (1 - \alpha) \mathbf{v}_k^{(\tau)} \tag{2}$$

where $\mathbf{v}_k^{(\tau)}$ is the *influential vector*. Next, we discuss how to determine the influential vector.

3.3 The influential nodes and vector

Intuitively, influential nodes represent the key positions visited by each walker. Thus, for each walker, we can use the nodes with large visiting probabilities as its influential nodes. Influential nodes can be selected in different ways. For example, suppose that for walker W_i , in group iteration τ , there are n ($n \geq 1$) nodes with the largest visiting probability. These n nodes can be treated as the influential nodes of W_i . We use vector $\mathbf{e}_i^{(\tau)}$ to represent the influential nodes of W_i in τ . In this case, the entries corresponding to the n influential nodes are set to $1/n$, and all other entries are 0. We can also include the 1-hop or 2-hop neighbors of the selected nodes as the influential nodes. Another way is to use the nodes with the top- l percent largest visiting probabilities as the influential nodes. Please see Sect. 7.2.4 for a comparison of different methods for selecting the influential nodes.

The influential vector $\mathbf{v}_k^{(\tau)}$ in Eq. (2) represents the averaged influence from all other walkers to W_k . That is,

$$\mathbf{v}_k^{(\tau)} = \frac{1}{K-1} \sum_{i \in \{1 \dots K\}, i \neq k} \mathbf{e}_i^{(\tau)} \tag{3}$$

Intuitively, in MWC, each walker has a probability of α to follow the transition probabilities in \mathbf{P} , and a probability of $(1 - \alpha)$ to jump to the set of influential nodes of other walkers. In this way, each walker is pulled back by all other walkers when deciding the next steps, which helps keep the walkers trapped in the local cluster.

3.4 Inhomogeneity of MWC

In RWR, let $\mathbb{P} = \alpha\mathbf{P} + (1 - \alpha)\mathbf{1}\mathbf{v}^\top$, then we can rewrite Eq. (1) as

$$\mathbf{x}^{(t+1)} = \mathbb{P}\mathbf{x}^{(t)} \tag{4}$$

Thus, \mathbb{P} can be treated as a transition matrix modified from the original transition matrix \mathbf{P} . The modification results from the influence of the query node whose effect is represented by the vector \mathbf{v} . In RWR, the modified transition matrix \mathbb{P} does not change over time. A random walk model with such a stationary transition matrix is usually referred to as a *homogeneous* Markov chain [14,23]. Most existing random walk models [8,11,15,24,25,36] are homogeneous.

In MWC, for walker W_k , let $\mathbb{P}_k^{(\tau+1)} = \alpha\mathbf{P} + (1 - \alpha)\mathbf{1}(\mathbf{v}_k^{(\tau)})^\top$, then we can rewrite Eq. (2) as

$$\mathbf{x}_k^{(\tau+1)} = (\mathbb{P}_k^{(\tau+1)})^\top \mathbf{x}_k^{(\tau)} \tag{5}$$

In Eq. (5), since $\mathbf{v}_k^{(\tau)}$ changes in different group iterations due to the change of the influential nodes, we have $\mathbb{P}_k^{(\tau+1)} \neq \mathbb{P}_k^{(\tau)}$. A model whose transition matrix dynamically changes is referred to as an *inhomogeneous* Markov chain [14,23].

Note that $\mathbb{P}_k^{(\tau+1)}$ in Eq. (5) is stochastic for all $\tau \geq 0$, since $\mathbb{P}_k^{(\tau+1)}\mathbf{1} = \alpha\mathbf{P}\mathbf{1} + (1 - \alpha)\mathbf{1}(\mathbf{v}_k^{(\tau)})^\top\mathbf{1} = \mathbf{1}$.

While the convergence property of homogeneous random walk models is well studied, the convergence property of the inhomogeneous MWC is more complicated and will be discussed in the next section.

4 Convergence and score vectors

In the experiments, we observe that after certain number of group iterations, the influential nodes of the walkers repeat periodically. This observation motivates us to study the convergence properties of MWC assuming the recurrence of the influential nodes. Based on these properties, we develop two score vectors that can be used for local clustering. Later, we also discuss the convergence property for the general case without assuming the recurrence of the influential nodes.

4.1 Convergence within a period

We first discuss the case in which the influential nodes repeat with a period of T group iterations after certain number of iterations. The following theorem shows that the probability distribution of any walker will also repeat with a period of T .

Theorem 1 *If the recurrence of the influential nodes occurs after τ_p group iterations and has a period of T , then for any W_k , there exists a constant τ_c ($\tau_p \leq \tau_c \leq \tau_p + \lfloor \frac{\log \epsilon}{\log \alpha} \rfloor$) for a computational tolerance ϵ such that when $\tau \geq \tau_c$, $\mathbf{x}_k^{(\tau+T)} = \mathbf{x}_k^{(\tau)}$.*

Proof See Sect. 1. □

Let $\mathbf{h}_k^{(\tau)} = \frac{1}{T} \sum_{\mu=1}^T \mathbf{x}_k^{(\tau+\mu)}$ be the average probability distribution of walker W_k over a period. The following corollary shows that the $\mathbf{h}_k^{(\tau)}$ also converges.

Corollary 1 $\mathbf{h}_k^{(\tau+1)} = \mathbf{h}_k^{(\tau)}$ when $\tau \geq \tau_c$.

In single-walker methods, the stationary probability distribution is usually used as a score vector to measure the proximity between different nodes. For example, in RWR, a node with a large stationary probability usually indicates it has high proximity to the query node and should be included in the local cluster [3,33].

In MWC, we define the *mean-score* vector as follows, which can be used to measure the proximity between the query node and other nodes.

Definition 1 When the period T exists, the *mean-score* vector ϕ is defined as

$$\phi = \sum_{k=1}^K \mathbf{x}_k / K \tag{6}$$

where $\mathbf{x}_k = \mathbf{h}_k^{(\tau_c)} = \frac{1}{T} \sum_{\mu=1}^T \mathbf{x}_k^{(\tau_c+\mu)}$ represents the average of the probability distribution of walker W_k over a period after convergence.

Next, we introduce the *std-score* (standard deviation score) vector, which can help determine whether a node is on the boundary of a cluster or not.

Definition 2 When the period T exists, the *std-score* vector ψ is defined as

$$\psi = \max_{1 \leq \mu \leq T} \sigma([\mathbf{x}_1^{(\tau_c+\mu)}, \mathbf{x}_2^{(\tau_c+\mu)}, \dots, \mathbf{x}_K^{(\tau_c+\mu)}]) \tag{7}$$

The $\sigma(\mathbf{A})$ operation in the above definition returns a column vector with each entry representing the standard deviation of its corresponding row in matrix \mathbf{A} . Thus, each entry $\psi(u)$ corresponds to node u and its value represents the maximum dispersion among the visiting probabilities of the K walkers over a period. Intuitively, a node u on the boundary of the local cluster will have high $\psi(u)$ value since the visiting probabilities of different walkers will be quite different.

4.2 The general case

Next, we study the convergence property of MWC without assuming the recurrence of the influential nodes.

In RWR (Eq. 4), it is known that $\mathbf{x}^{(t)}$ will converge to a stationary vector \mathbf{x} and $\lim_{n \rightarrow \infty} \|\mathbb{P}^n - \mathbf{1}\mathbf{x}^\top\|_\infty = 0$ [10,14]. That is, the power limit of \mathbb{P} will converge to the *constant matrix* $\mathbf{1}\mathbf{x}^\top$ whose rows are all identical.

For a non-negative stochastic matrix \mathbf{A} , the following measure is commonly used to measure how different the row vectors are [10,14]:

$$\delta(\mathbf{A}) = \max_w \max_{v,v'} |\mathbf{A}(v, w) - \mathbf{A}(v', w)|$$

The δ value measures the maximum difference between two entries in every column. By definition, for any non-negative stochastic matrix \mathbf{A} , $0 \leq \delta(\mathbf{A}) \leq 1$. Intuitively, if all rows in \mathbf{A} are similar, δ will be small (with $\delta = 0$ if all rows are identical).

Let $\mathbb{P}_k^{(i,j)} = \mathbb{P}_k^{(i)}\mathbb{P}_k^{(i+1)} \dots \mathbb{P}_k^{(j)}$ ($j \geq i$). Then Eq. (5) can be rewritten as

$$\mathbf{x}_k^{(\tau+1)} = (\mathbb{P}_k^{(1,\tau+1)})^\top \mathbf{x}_k^{(0)} \tag{8}$$

Theorem 2 *If the transition matrix \mathbf{P} is stochastic, irreducible and aperiodic,² then for W_k , for any $\epsilon > 0$, there exists an integer $\nu(\epsilon)$ such that, when $\tau_s \geq \nu(\epsilon)$, $\delta(\mathbb{P}_k^{(1,\tau_s)}) < \epsilon$. And $\nu(\epsilon) = \log \epsilon / \log(\alpha \lambda(\mathbf{P}))$, where $\lambda(\mathbf{P}) = 1 - \min_{v,v'} \sum_w \min(\mathbf{P}(v, w), \mathbf{P}(v', w))$*

Proof See Sect. 1. □

Theorem 2 shows after certain number of group iterations, the difference between row vectors of $\mathbb{P}_k^{(1,\tau_s)}$ will become sufficiently small. This is different from $\lim_{\tau_s \rightarrow \infty} \|\mathbb{P}_k^{(1,\tau_s+1)} - \mathbb{P}_k^{(1,\tau_s)}\|_\infty = 0$, which implies the convergence of $\mathbf{x}_k^{(\tau_s)}$. We refer to Theorem 2 as the *weak convergence* property for the general case of MWC.

Since there is no guarantee that $\mathbf{x}_k^{(\tau)}$ will converge when the period does not exist, we modify the mean-score vector and std-score vector definitions as follows for the general case.

Definition 3 When the period does not exist, the *mean-score* vector ϕ is defined as

$$\phi = \sum_{k=1}^K \mathbf{x}_k^{(\tau_s)} / K \tag{9}$$

Without loss of generality, τ_s is assumed to be large enough so that $\delta(\mathbb{P}_k^{(1,\tau_s)})$ is sufficiently small for all W_k ($1 \leq k \leq K$).

Definition 4 When the period does not exist, the *std-score* vector ψ is defined as

$$\psi = \sigma([\mathbf{x}_1^{(\tau_s)}, \mathbf{x}_2^{(\tau_s)}, \dots, \mathbf{x}_K^{(\tau_s)}]) \tag{10}$$

5 Algorithm

Computing the score vectors Algorithm 1 shows the overall algorithm, MWC, to compute the mean-score and std-score vectors. Within each group iteration τ (Lines 2–10), for each walker, the algorithm updates the node visiting probability vector by calling the UPDATE procedure outlined in Algorithm 2 [which is based on Eq. (2)], stores its influential nodes in

² This is the same set of conditions used to prove the convergence of RWR by the Perron–Frobenius theorem [23].

inodes, and detects the periodic recurrence of the influential nodes. If a period T is detected (Line 11), it calls the MWC-Period procedure outlined in Algorithm 3 to compute the two vectors based on Eqs. (6) and (7). Otherwise (Line 12), it computes the two vectors based on Eqs. (9) and (10). The default value of τ_s is set to 20 since in practice the algorithm breaks the *while* loop very fast (usually within 10 iterations).

Once the periodic recurrence of the influential nodes is detected, the MWC-Period procedure finds the converged mean-score and std-score vectors. From Line 4–12, the algorithm computes \mathbf{x}_k , which represents the average of the probability distribution vectors of walker W_k over a period, and checks whether it is the same as that in the last period. If \mathbf{x}_k converges for all the walkers, the final mean-score and std-score vectors are computed.

Complexity Given a graph $G(V, E)$, if the period does not exist, the time complexity of MWC is $O(\tau_s K(|E| + |V|))$. If the period T exists, the complexity is $O(\tau_c T K(|E| + |V|))$. In practice, we usually have $\tau_c < 20$, and $T < 10$.

Local clustering To find the local cluster, we first find the top- L nodes with the largest mean-scores, where L is the largest possible cluster size with the default value 200. Let $\{s_i\} (1 \leq i \leq L)$ represent the list of top- L nodes sorted in descending order. Then for each i ($1 \leq i \leq L$), we compute the conductance of the subgraph induced by node set $\{s_1, \dots, s_i\}$. The node set with the smallest conductance will be returned as the local cluster.

Complexity Generating the list of the top- L sorted nodes takes time $O(|V| + L \log L)$. Computing the conductances and finding the smallest one need $O(Ld_{S_L})$, where d_{S_L} is the average degree of the top- L nodes.

Algorithm 1: MWC

```

Input:  $P, K, \alpha, q, \tau_s$ 
Output:  $\phi, \psi$ 
1  $T = 0; \tau = 1; \mathbf{x}_k = \mathbf{0}; \mathbf{x}_k(q) = 1; \mathbf{e}_k^{(0)} = \mathbf{x}_k (1 \leq k \leq K)$ 
2 while  $\tau < \tau_s$  do
3   for  $k = 1$  to  $K$  do
4     Compute  $\mathbf{v}_k$  based on Equation (3);
5      $\mathbf{x}_k = \text{UPDATE}(P, K, \alpha, \mathbf{x}_k, \mathbf{v}_k)$ ;
6     Store the new influen. nodes of  $\mathbf{x}_k$  into inodes;
7     Compute  $\mathbf{e}_k$ ;
8   Detect  $T$  in inodes;
9   if  $T$  is detected then  $\Phi = [\mathbf{x}_1, \dots, \mathbf{x}_K]$ ; break;
10   $\tau = \tau + 1$ ;
11 if  $T$  is detected then  $[\phi, \psi] = \text{MWC-Period}(P, K, \alpha, T, \Phi)$ ;
12 else  $\phi = \sum_{k=1}^K \mathbf{x}_k / K; \psi = \sigma([\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_K])$ ;
13 return  $\phi$  and  $\psi$ ;

```

6 Computational speed up

In this section, we discuss how to further reduce the computational cost of MWC. The key idea is to only update a small subset of the nodes that are near the influential nodes.

Recall that in iteration τ , for walker W_k , the influential nodes are the set of nodes with the largest visiting probability. Instead of updating all nodes, we only update a subset of nodes

Algorithm 2: UPDATE

Input: $\mathbf{P}, K, \alpha, \mathbf{x}_k^{(\tau)}, \mathbf{v}_k^{(\tau)}$ **Output:** $\mathbf{x}_k^{(\tau+1)}$

- 1 $\mathbf{x}_k^{(\tau+1)} = \alpha \mathbf{P} \mathbf{T} \mathbf{x}_k^{(\tau)} + (1 - \alpha) \mathbf{v}_k^{(\tau)}$;
- 2 **return** $\mathbf{x}_k^{(\tau+1)}$;

Algorithm 3: MWC-Period

Input: $\mathbf{P}, K, \alpha, T, \Phi$
Output: ϕ, ψ

- 1 $\mathbf{x}_k^{(\mu)} = \mathbf{0}$ ($1 \leq k \leq K, 1 \leq \mu \leq T$);
- 2 $[\mathbf{x}_1 \cdots \mathbf{x}_K] = [\mathbf{x}_1^{(1)} \cdots \mathbf{x}_K^{(1)}] = \Phi$;
- 3 Initialize the \mathbf{e}_k ($1 \leq k \leq K$) according to Φ ;
- 4 **while** any \mathbf{x}_k ($1 \leq k \leq K$) has not converged **do**
- 5 **for** $\mu = 1$ to T **do**
- 6 **for** $k = 1$ to K **do**
- 7 Compute \mathbf{v}_k based on Equation (3);
- 8 $\mathbf{x}_k^{(\mu+1)} = \text{UPDATE}(\mathbf{P}, K, \alpha, \mathbf{x}_k^{(\mu)}, \mathbf{v}_k)$;
- 9 Update \mathbf{e}_k according to $\mathbf{x}_k^{(\mu+1)}$;
- 10 **for** $k = 1$ to K **do**
- 11 $\mathbf{x}_k = \sum_{\mu=1}^T \mathbf{x}_k^{(\mu)} / T$;
- 12 $\mathbf{x}_k^{(1)} = \mathbf{x}_k^{(T+1)}$;
- 13 $\phi = \sum_{k=1}^K \mathbf{x}_k / K$; $\psi = \max_{1 \leq \mu \leq T} \sigma([\mathbf{x}_1^{(\mu)}, \mathbf{x}_2^{(\mu)}, \dots, \mathbf{x}_K^{(\mu)}])$;
- 14 **return** ϕ and ψ ;

that (1) are in the neighborhood of the influential nodes, and (2) cover a significant amount of probabilities.

In group iteration τ , for walker W_k , let $Y_l^{(\tau)}$ represent the l -hop neighbors of the influential nodes of W_k , and $D^{(\tau)}$ represent the set of influential nodes of all other walkers. The *core node set* of W_k is defined as

$$C_k^{(\tau)} = D^{(\tau)} \cup Y_l^{(\tau)},$$

where l is the smallest hop number such that the sum of the visiting probabilities of the nodes in $C_k^{(\tau)}$ is above certain threshold. More specifically, we require $Pr(C_k^{(\tau)}) = \sum_{i \in C_k^{(\tau)}} \mathbf{x}_k^{(\tau)}(i) \geq \theta$, where $\mathbf{x}_k^{(\tau)}(i)$ is the i th entry in $\mathbf{x}_k^{(\tau)}$, and θ is the threshold representing the proportion of the probabilities that the core node set should cover.

Intuitively, the nodes that are not in the core node set are less important for local clustering, since they are far away from the influential nodes and only have small visiting probabilities. Thus, it is unlikely that these nodes are parts of the local cluster.

For example, suppose that we are interested in finding the core node set for W_1 in a 2-walker MWC, and node visiting probabilities of W_1 are shown as in Fig. 4. The influential nodes of W_1 and W_2 are $\{\textcircled{1}\}$ and $\{\textcircled{7}\}$, respectively, and $\theta = 0.9$. We have $D^{(\tau)} = \{\textcircled{7}\}$,

Fig. 4 An example of partial nodes updating

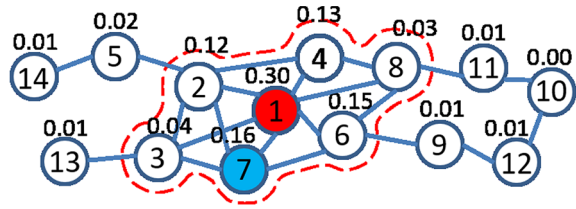
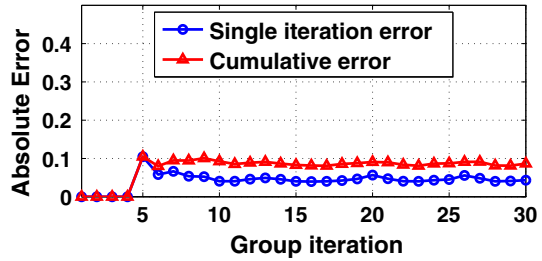


Fig. 5 The single iteration and cumulative errors



$Y_{l=1}^{(\tau)} = \{①, ②, ③, ④, ⑥, ⑦, ⑧\}$, and $Pr(D^{(\tau)} \cup Y_{l=1}^{(\tau)}) = 0.93 > \theta$. Thus, the core node set is $\{①, ②, ③, ④, ⑥, ⑦, ⑧\}$ as highlighted by the red dotted curve in the figure.

Instead of updating all the nodes, we only update the nodes in the core node set and their direct neighbors. Let Δ represent the difference between the exact and estimated probability vectors. The following theorem gives a loose error bound on Δ . Its proof and a tighter bound can be found in “Appendix A.3.”

Theorem 3

$$\Delta \leq 2(1 - \theta)$$

Figure 5 shows the estimation errors on the real-world Live Journal network, which has about 4 million nodes and 34 million edges. The parameter settings are $K = 5, \alpha = 0.6,$ and $\theta = 0.6$. The blue line shows the error in each group iteration and the red line shows the cumulative error over all iterations. The results are averaged on 200 randomly selected query nodes. As can be seen from the figure, both single iteration and cumulative errors are small. In practice, we only need to update less than 0.02% of the nodes to achieve more than 0.98 correlation between the estimated and the exact score vectors. The detailed evaluation on the core node set updating strategy can be found in Sect. 7.1.6.

7 Experimental results

We conduct extensive experiments to evaluate the performance of the proposed method using a variety of real-world and synthetic networks. All experiments are performed on a server with 64 G memory, Intel Xeon 2.6 GHz CPU, and Redhat OS. The source code of MWC is available at <http://sites.psu.edu/yuchenbian/mwc/>.

Table 1 Statistics of the real networks

Name	AZ	DB	YT	LJ	OT
$ V $	334,863	317,080	1,134,890	3,997,962	3,072,441
$ E $	925,872	1,049,866	2,987,624	34,681,189	117,185,083
Density	2.76	3.31	2.63	8.67	38.14
# Cluster	75,149	13,477	8385	287,512	6,288,363

7.1 Evaluation on real networks

7.1.1 Datasets and state-of-the-art methods

The statistics of the real networks are shown in Table 1 (AZ: Amazon; DB: DBLP; YT: YouTube; LJ: LiveJournal; OT: Orkut). These datasets are provided with ground-truth cluster labels and are publicly available at <http://snap.stanford.edu>.

We compare MWC with several state-of-the-art local clustering methods. PageRank–Nibble (PRN) adopts the degree-normalized RWR score to rank the nodes and finds the local cluster that minimizes the external conductance [3]. MARW uses multiple agents with a rope of fixed length to find the local cluster [1]. The query-biased dense connected subgraph detection (QDC) method uses RWR to assign weights to the nodes and find the query-biased densest subgraph [33]. LEMON finds the local cluster by seeking a sparse vector in the span of the local spectra such that the query is in its support [22]. The heat kernel (HK) method uses the heat kernel diffusion process instead of random walk and adopts the PageRank–Nibble framework to find local clusters [16].

7.1.2 Accuracy evaluation

We first evaluate the accuracy of the selected methods. We use *F-score* and *consistency* to measure the accuracy of the detected clusters. Given the discovered local cluster S' and the ground-truth cluster S , *F-score* is defined as:

$$F(S', S) = 2 \cdot \frac{\text{prec}(S', S) \times \text{rec}(S', S)}{\text{prec}(S', S) + \text{rec}(S', S)},$$

where $\text{prec}(S', S) = \frac{|S' \cap S|}{|S'|}$ is the precision and $\text{rec}(S', S) = \frac{|S' \cap S|}{|S|}$ is the recall.

Consistency measures the variation of the *F-scores* of the identified local clusters given different query nodes from the same cluster. It is defined as

$$1 - \sqrt{\frac{1}{|S|} \sum_{s' \in S} (F(S', S) - F_{\text{mean}})^2},$$

where s' is a node in the detected cluster S , S' is the detected cluster when s' is the query, and $F(S', S)$ is the *F-score* of S' when S is used as the ground-truth, and $F_{\text{mean}} = \frac{1}{|S|} \sum_{s' \in S} F(S', S)$. If a method has both high *F-score* and high consistency, it suggests that the method is accurate and robust to different choices of the query nodes. On the other hand, if a method has a low *F-score* and a high consistency value, this does not mean the method is robust to different choices of the query nodes. The high consistency value is merely due to the consistently low *F-scores*.

Table 2 *F*-scores of the selected methods

Methods	MWC	PRN	QDC	MARW	LEMON	HK
AZ	0.9010	0.7774	0.8083	0.8444	0.8195	0.5209
DB	0.4929	0.3962	0.4368	0.4010	0.4551	0.3021
YT	0.1943	0.1299	0.1129	0.0862	0.1412	0.0920
LJ	0.7033	0.6378	0.6586	0.3895	0.6635	0.5056
OT	0.3363	0.2597	0.2850	0.1333	0.2880	0.1787

The best performers are shown in bold

Table 3 Consistency of the selected methods

Methods	MWC	PRN	QDC	MARW	LEMON	HK
AZ	0.9803	0.8746	0.9532	0.9273	0.8277	0.6869
DB	0.8441	0.7021	0.7634	0.7938	0.6265	0.7863
YT	0.6793	0.6018	0.6434	0.6346	0.5582	0.9162
LJ	0.8417	0.8086	0.8021	0.8223	0.5602	0.7784
OT	0.7472	0.7303	0.7056	0.6149	0.7452	0.8632

The best performers are shown in bold

Unless otherwise mentioned, in MWC, *K* and α are set to 5 and 0.6, respectively. For all other methods, we tune their parameters to achieve their best performance. All results are averaged on 200 randomly selected query nodes.

Table 2 shows the *F*-scores of different local clustering methods. We can see that MWC achieves the highest *F*-scores on all datasets. The performances of two recently proposed methods, QDC and LEMON, are the second best. Let *a* be the *F*-score of MWC on one dataset and *a'* be the highest *F*-score of all other methods on the same dataset. We use $\frac{a-a'}{a'}$ represent the ratio of how much MWC outperforms other methods. The ratio ranges from 6 to 38% on the 5 datasets, with the average being 16%. This shows that MWC outperforms other methods by a large margin.

Table 3 shows the results on consistency. MWC has the highest consistency on datasets AZ, DB, and LJ, and the second highest consistency on datasets YT and OT. HK has the highest consistency on YT and OT. This is due to the low *F*-scores of HK on these two datasets. As shown in Table 2, the *F*-scores of HK on YT and OT are 0.092 and 0.1787, respectively. A high consistency value combined with a lower *F*-score value suggests that the method consistently gets relatively low *F*-scores on these two datasets.

7.1.3 Cluster quality evaluation

Next, we use *conductance*, *density*, *relative density*, and *cohesiveness* as metrics to evaluate the quality of the identified clusters [26,34].

The conductance measures how well the cluster is separated from the remaining part of the network. It is defined as $\frac{e(S, V \setminus S)}{\min\{e(S), e(V \setminus S)\}}$, where *e*(*S*, *V* \ *S*) is the cut between *S* and *V* \ *S*, and *e*(*S*) is the number of edges in *S*.

The density of *S* is the ratio between the number of edges *e*(*S*) and the number of nodes in *S*, i.e., $\frac{e(S)}{|S|}$. The relative density is defined as $\frac{e(S)}{e(S)+e(S, V \setminus S)}$.

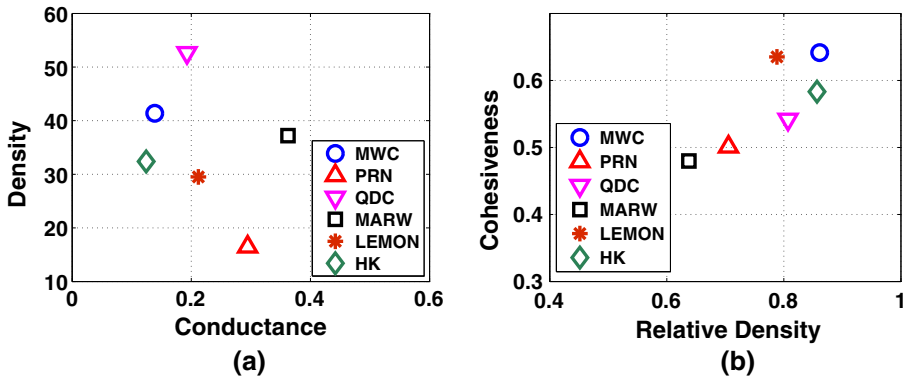


Fig. 6 Conductance, density, relative density, and cohesiveness of the identified clusters in the LJ dataset

The cohesiveness of S is defined as the minimum internal conductance in the subgraph induced by S , i.e.,

$$\min_{S' \subset S} \frac{e(S', S \setminus S')}{\min\{e(S'), e(S \setminus S')\}}.$$

A good cluster usually has low conductance, high density, high relative density, and high cohesiveness.

Figure 6 shows the average value of the identified clusters in the LJ dataset when using these four metrics. As can be seen from the figure, MWC achieves the highest relative density, the highest cohesiveness, the second lowest conductance, and the second highest density.

7.1.4 Comparing random walk methods

Next, we compare MWC with other random walk methods. The goal is to see whether the ranking results generated by these random walk methods can effectively capture the local clustering structure. We compare MWC to the state-of-the-art random walk methods including RWR, SimRank [15] and Panther [37]. The SimRank value between two nodes measures the expected number of steps required before two walkers, one starting from each node, meet at the same node if they walk in lock-step [15]. Panther is a recently proposed proximity measure based on both common neighbors and structural contexts of the nodes [37].

We use the LJ (LiveJournal) dataset as an example, and randomly select query nodes from clusters with sizes between 20 and 40.³ Figure 7 shows the F -scores of different random walk methods when the top-ranked nodes are considered as the local clusters. We can see that MWC clearly outperforms other methods. This shows that the top-ranked nodes are more likely to be in the local cluster of the query node when using MWC. If we simply use the top-ranked nodes as the local cluster, MWC will achieve the highest accuracy.

We further evaluate how similar the ranking results are if we use different query nodes in the same cluster. We randomly select a pair of nodes from the same cluster, and examine the correlation between the two ranking lists generated for the two query nodes. To capture

³ For the well-structured top-5000 clusters in LiveJournal that are suggested to use in the original paper [34], the averaged community size is 28.

Fig. 7 *F*-scores of the top-ranked nodes of different random walk methods

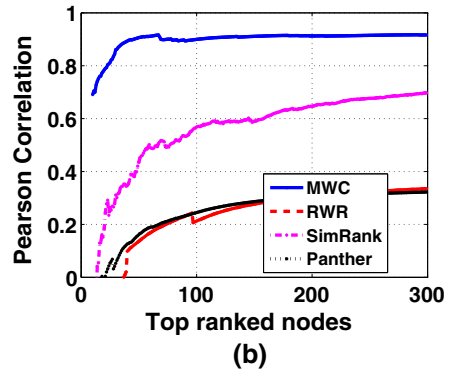
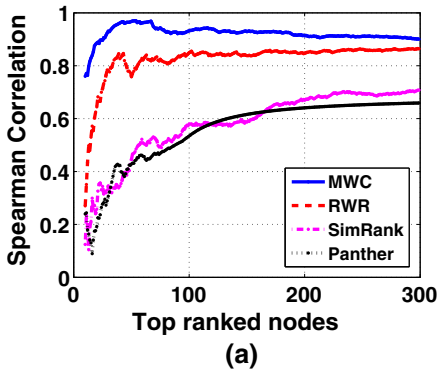
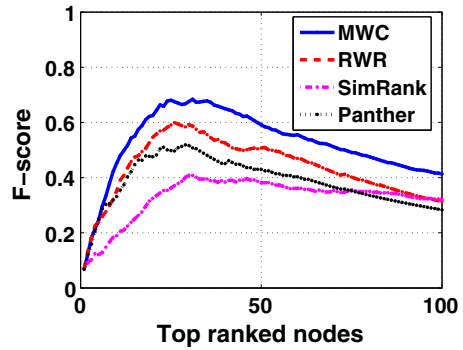


Fig. 8 Correlation between the lists of top-ranked nodes when two nodes from the same cluster are used as the query nodes

the local cluster effectively, an ideal method should return highly correlated ranking results since the two nodes are taken from the same cluster.

Figure 8a, b shows the Spearman’s rank correlation and Pearson correlation of the top-ranked nodes using different random walk methods. The results are averaged on 1000 query pairs. As we can see, MWC achieves much higher correlations compared to other methods. The reason is that the ranking results of RWR, SimRank and Panther are biased toward the query node. Thus, if the query node is not at the center of the local cluster, the nodes outside the cluster but close to the query node will also be included in the result. On the other hand, MWC gives consistent ranking results suggesting that it is robust to the choice of the query nodes.

7.1.5 Detecting the cluster center and boundary

We evaluate the capacity of MWC to detect the center nodes and boundary nodes of the local cluster. We adopt the *betweenness centrality* [6] to quantify a node’s role in the cluster. Intuitively, the center nodes should have high betweenness centrality within the subgraph induced by the cluster. On the other hand, the boundary nodes should have high betweenness centrality when considering nodes from different clusters since they serve as the bridges connecting these clusters.

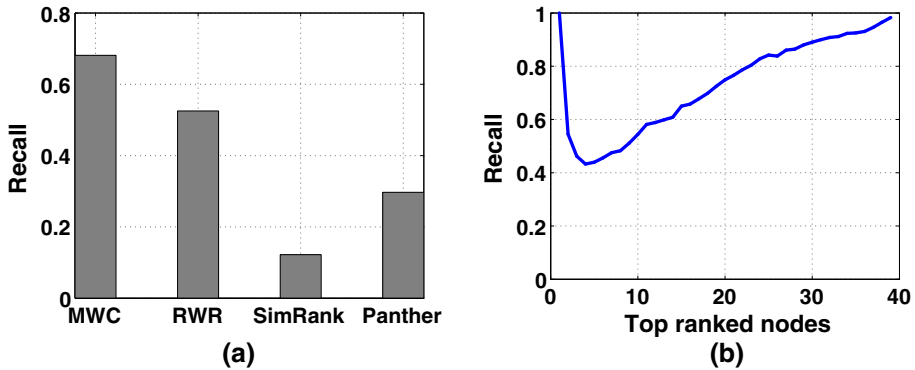


Fig. 9 Detecting the center nodes and boundary nodes of the local cluster

We first evaluate MWC's ability to detect the center nodes. We consider the top-5 nodes with the highest betweenness centrality values within the subgraph induced by the local cluster as the true cluster center nodes. For each ranking method, we report the recall of the top-5 ranked nodes, i.e., among the top-5 ranked nodes, how many are true cluster center nodes.

Figure 9a shows the results on the LJ dataset. We can see that the recall of MWC is around 70%. This indicates that MWC can consistently rank the cluster center nodes among the top nodes. Other random walk methods are biased toward the query nodes.

Next we evaluate MWC's ability to detect the boundary nodes based on their std-scores. Using the ground-truth cluster labels, we find the true boundary nodes, which are in the target cluster but have connections to the outside nodes. The betweenness centrality value of a boundary node is the number of shortest paths that connect the nodes in the target cluster and nodes in the neighborhood clusters and go through the boundary node.

We generate two ranking lists and examine the recall. One list includes the true boundary nodes arranged in descending order based on their betweenness centrality values. Another list includes the top-ranked nodes according to their std-scores identified by MWC. Figure 9b shows the recall when we use the top-ranked nodes in the first list as ground truth. Note that the average number of true boundary nodes is about 22 in our experiments. From the figure, we can see that the recall is about 0.80 when comparing the lists of the top-22 ranked nodes. This demonstrates that the std-score is a good indicator of the boundary nodes. Other random walk methods cannot provide any information about the boundary nodes since they are based on the single-walker models.

7.1.6 The core node set updating strategy

We evaluate the core node set updating strategy introduced in Sect. 6. Figure 10 shows the results on the LJ datasets. The blue line shows the Spearman's rank correlation between the exact and estimated mean-score vectors for the top-200 ranked nodes when varying θ , and the red line shows the percentage of the nodes need to be updated. We can see that the high rank correlation can be achieved by only updating a very small portion of the nodes. For example, by updating 0.02% nodes, we can still archive 0.98 rank correlation between the exact and estimated vectors. This demonstrates the effectiveness of the core node set updating strategy.

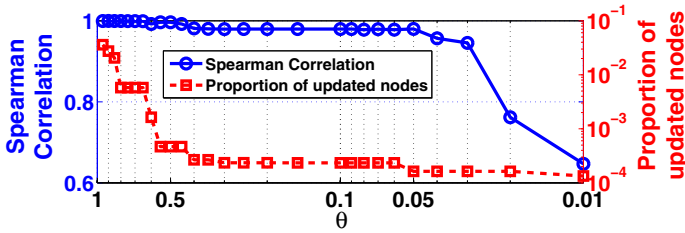


Fig. 10 Core node set updating evaluation

Fig. 11 Parameter sensitivity evaluation

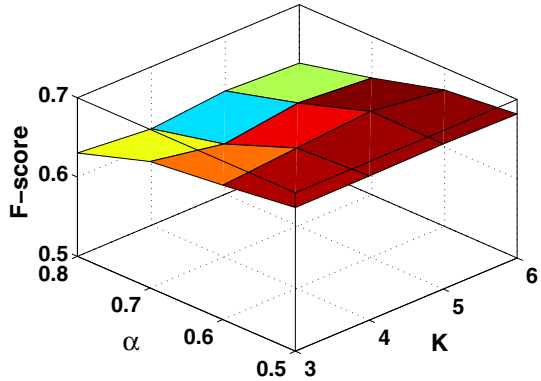
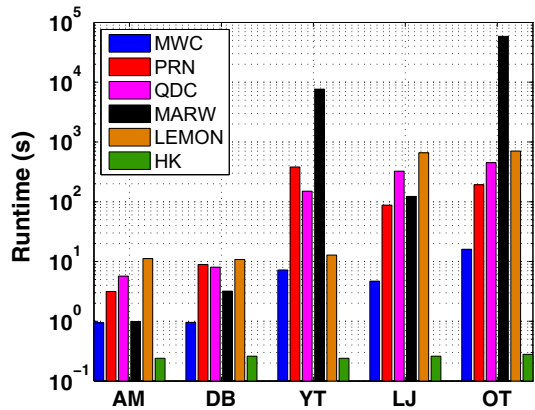


Fig. 12 Runtime evaluation



7.1.7 Parameter sensitivity and runtime

MWC has two parameters, K , the number of walkers, and α , the weight of the influence from other walkers. Figure 11 shows the F -scores of MWC when varying K and α on the LJ dataset. We can see that MWC is robust to a wide range of parameter settings. The F -score slightly decreases for larger α values. Intuitively, a larger α value means that each walker is more independent, while a smaller α value allows stronger influence among the group of walkers.

Figure 12 shows the running time of the selected local clustering methods on the real networks. On all datasets, MWC is the second fastest method. The HK method runs the fastest

Table 4 Synthetic datasets with different sizes (SDI)

Name	SD11	SD12	SD13	SD14
$ V $	10^5	5×10^5	10^6	1.5×10^6
$ E $	10^6	5×10^6	10^7	1.5×10^7
Density	10	10	10	10
μ	0.30	0.30	0.30	0.30

Table 5 Synthetic datasets with different mixing parameters μ (SDII)

Name	SD21	SD22	SD23	SD24
$ V $	10^5	10^5	10^5	10^5
$ E $	10^6	10^6	10^6	10^6
Density	10	10	10	10
μ	0.15	0.30	0.45	0.60

due to its relaxation strategy used to estimate the heat kernel diffusion process. However, as discussed in Sect. 7.1.2, the accuracy of HK is not as high as other local clustering methods. The exact MARW method is very time consuming due to its exponential search space. Here, we adopt the simulation process as suggested in the original paper. The results are approximate, and there is no theoretical guarantee on how accurate the estimations are.

7.2 Evaluation on synthetic datasets

To gain more insights about the performance of MWC, we use the benchmark graph generator [18] to generate a collection of synthetic datasets to evaluate the performance of MWC. These synthetic datasets can be tuned with different parameters and have accurate cluster labels.

Tables 4 and 5 summarize two groups of synthetic datasets generated by the benchmark graph generator [18]. We denote them as SDI and SDII, respectively. To evaluate the scalability of proposed algorithm, the generated synthetic datasets contains millions of nodes and tens of millions of edges. The four datasets in SDI have different sizes and the same density and mixing parameter μ . The four datasets in SDII have different μ and the same size and density. In the network generating model, μ indicates the proportion of a node's neighbors that are outside the target cluster. The boundaries between different clusters become less clear for larger μ values. By tuning μ , we can vary the clearness of the clustering structure. We use the default settings for all other parameters.

7.2.1 Accuracy evaluation

Figure 13a shows the F -scores of the selected methods with different dataset sizes (with fixed $\mu = 0.3$). As can be seen from the figure, MWC achieves the highest accuracy compared to other methods. Specifically, the F -score of MWC is above 0.95. Figure 13b shows the F -scores when varying μ (with fixed $|V| = 10^5$ and $|E| = 10^6$ in Table 5). As we can see, the F -scores of all methods decrease when increasing μ , i.e., lowering the clearness of the clustering structure. MWC has the highest F -score, and clearly outperforms other methods. Moreover, the performance gap between MWC and other methods tends to become bigger for larger μ values, which demonstrates that MWC is more robust to μ .

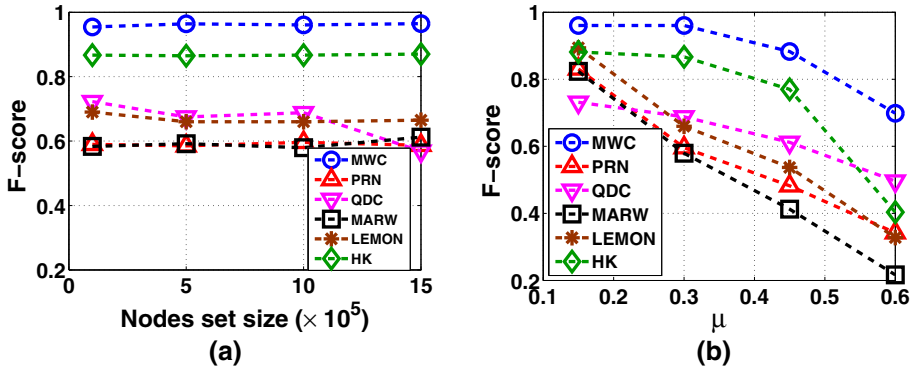
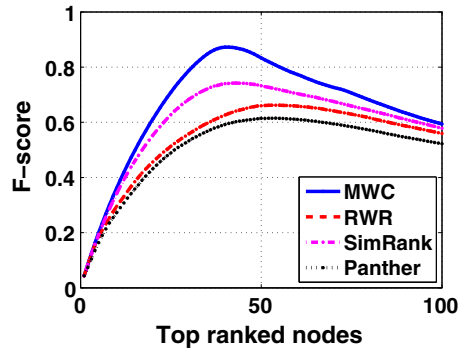


Fig. 13 Accuracy on the synthetic datasets

Fig. 14 *F*-scores of the top-ranked nodes of different proximity measures



7.2.2 Comparison with other proximity measures

In synthetic networks, we use dataset SD23 as an example to study the ranking results of MWC, RWR, SimRank [15] and Panther [37] to see how well they capture the local clustering structure for a given query node. We randomly select 200 query nodes from the ground-truth clusters. Figure 14 shows the *F*-scores of different proximity measures when the top-ranked nodes are considered as the local clusters. The results show that MWC consistently outperforms other proximity measures.

Applying the same experimental setting in Sect. 7.1.4, we further evaluate how similar the ranking results are given two different query nodes in the *same* cluster for each proximity measure. We randomly select 1000 pairs of nodes from the same cluster. Figure 15a, b shows the averaged Spearman’s rank correlation and Pearson correlation of top-ranked nodes for different proximity measures. As we can see, MWC achieves much higher correlations compared to other measures.

7.2.3 Detecting the cluster center and boundary

Next, we evaluate the capacity of MWC to detect the center nodes and boundary nodes of the local cluster on synthetic networks with the same experimental setting of real networks.

Figure 16a shows the cluster center nodes detection results on dataset SD23. We can see that the recall of MWC can reach as high as 90%. This shows that MWC can consistently

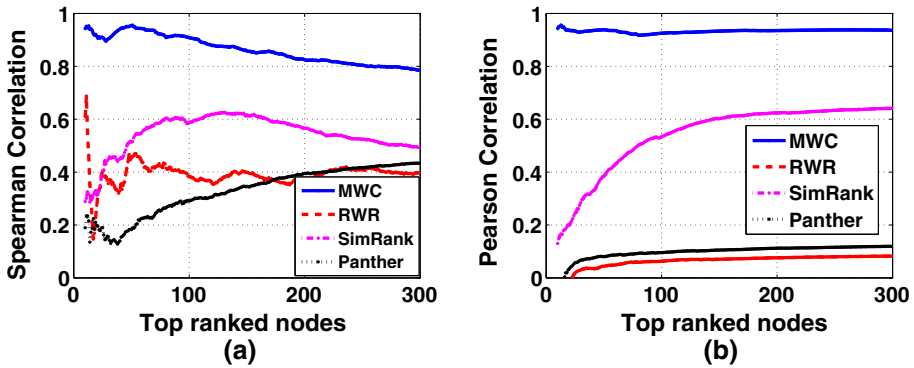


Fig. 15 Correlation between the lists of top-ranked nodes when two nodes from the same cluster are used as the query nodes

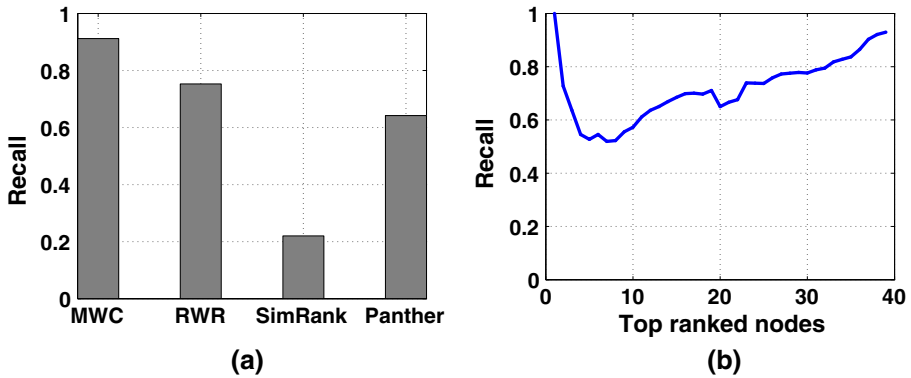


Fig. 16 Detecting the center nodes and boundary nodes on synthetic networks

rank the cluster center nodes among the top nodes. Other proximity measures are all biased toward the query nodes.

Figure 16b shows the recall of cluster boundary nodes detection using std-score vector. Note that the average number of true boundary nodes is about 35 in our experiments. From the figure, we can see that the recall is about 0.85 when comparing the lists of the top-35 ranked nodes. This demonstrates that the std-scores are good indicators of the boundary nodes of the local cluster.

7.2.4 Influential nodes selection

We compare two different approaches to select the influential nodes. The first is that for each walker, we use the r -hop neighbors of the nodes with the largest visiting probability as its influential nodes. Another approach is to use the nodes with the top- l percent largest visiting probabilities as the influential nodes. With larger r or l , more nodes will be selected as the influential nodes.

We run 200 trials on dataset SD23 using a 5-walker chain to evaluate the performance of different r and l . From Fig. 17, we can see the F -score becomes smaller for larger r and l .

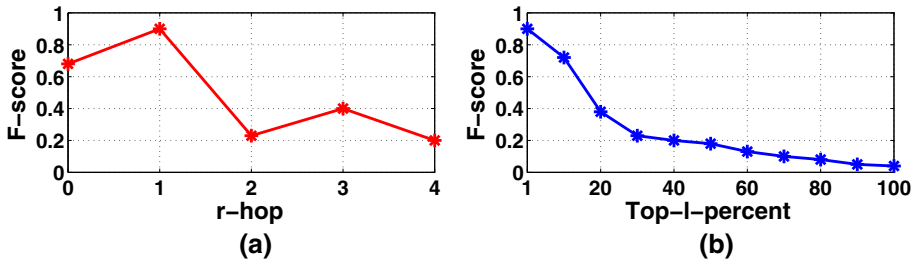


Fig. 17 Influential nodes selection evaluation

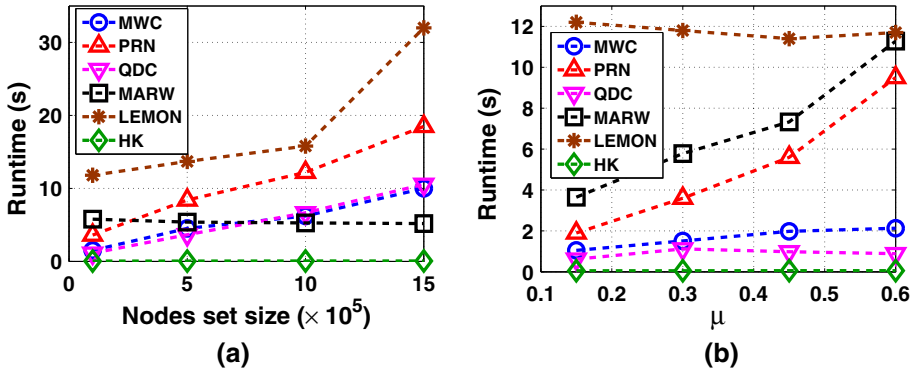


Fig. 18 Running time on synthetic datasets

This is intuitive since the influential nodes should represent the key positions visited by each walker. Including more irrelevant nodes will decrease the performance.

7.2.5 Running time evaluation

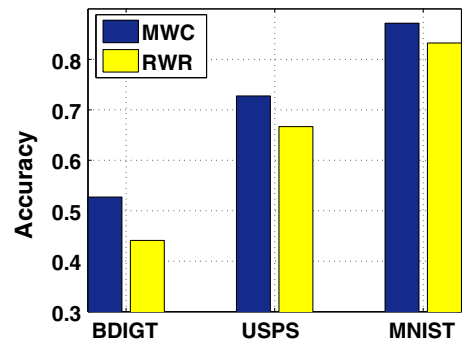
Figure 18a shows the runtime of the selected local clustering methods on synthetic networks with different sizes. We can see that MWC is among the top-3 fastest algorithms. The runtime of all methods except MARW and LEMON is linear to the size of network. Figure 18b shows the runtime for different μ values. MWC, HK, LEMON and QDC are not sensitive to μ . MARW is not sensitive to the network size but sensitive to μ . This is because in the simulation process, it becomes harder to accept a valid step when the clustering structure is less clear. For PRN, both the approximation and sweep processes take longer time for networks with larger μ values because of the blurred cluster boundaries.

7.3 Additional applications of MWC

MWC can also be used in other local clustering related applications. In this section, we use graph-based semi-supervised learning and image auto-annotation as examples to illustrate the effectiveness of MWC. Our intention is not to show that MWC outperforms the methods specifically designed for these applications. We only use the datasets generated in these applications to show the potential applicability of MWC, since RWR has also been proposed to solve these problems.

Table 6 The handwritten image datasets

Name	BDIGT	USPS	MNIST
# Cluster	36	10	10
Cluster size	39	1000	5000
Labeled images	72	20	20
k of k NN	10	20	30
$ V $	1404	1×10^4	5×10^4
$ E $	14,040	2×10^5	1.5×10^6

Fig. 19 Handwritten image prediction accuracy

7.3.1 Graph-based semi-supervised learning

In this application, graphs are utilized for classification [39]. An edge represents the similarity between two connected data objects. The goal is to classify the unlabeled data objects based on the partially labeled objects. We compare the performances of MWC and RWR in this application.

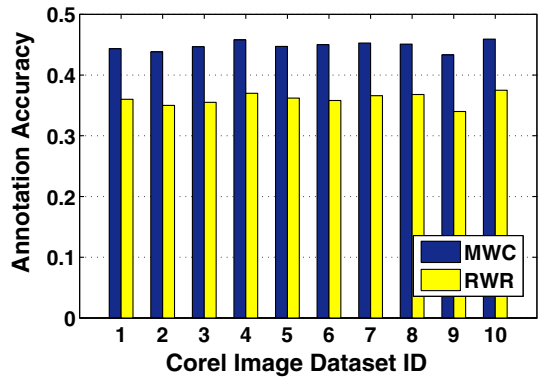
We use three handwritten image datasets publicly available at <http://www.cs.nyu.edu/~roweis/data.html>. Table 6 shows the detailed information of these datasets. The weighted k NN graphs are constructed following the strategy in [32]. For each query image, we compute the scores with the labeled images and use the label of the nearest labeled neighbor as the predicted label. Figure 19 shows the classification accuracy of MWC and RWR. We can see that MWC achieves higher accuracy than RWR on all 3 datasets.

7.3.2 Image auto-annotation

Image auto-annotation is an important problem in computer vision. The goal is to automatically annotate related keywords to a newly added image from the keyword list based on an annotated image dataset [5,9,24]. RWR has been proposed for this application [24].

We use the 10 Corel image datasets in [5,24]. Each image dataset includes about 5200 annotated images, 1740 test images, and 160 keywords. Each annotated image is captioned with 3–4 keywords. A mixed media graph (MMG) is constructed following the strategy in [24]. For each dataset, the graph consists of about 55,000 nodes and 170,000 edges. We run one trial for each test image and calculate the annotation accuracy, which is the percentage of correctly predicted keywords [24]. Figure 20 shows the average accuracy for all test images in the 10 datasets. We can see that MWC outperforms RWR by about 10% in terms of the accuracy.

Fig. 20 Corel image auto-annotation accuracy



8 Conclusion

In this paper, we propose the multi-walker chain (MWC) model for effective local community detection in large networks. Different from the traditional single-walker models, MWC uses multiple walkers to explore the network. The walkers influence each other; thus, it is less likely for the entire group of walkers to walk out of the cluster. We develop two measures based on the mean and standard deviation of the node visiting probabilities of the walkers. The mean-scores can be used to detect the local cluster and the std-scores can provide insight about the cluster boundary nodes. In our experiments, MWC can identify local clusters more accurately than alternatives. Moreover, MWC is not sensitive to the choice of different query nodes in the same cluster. Our partial node updating strategy allows to update only a very small portion of the nodes without compromising the accuracy.

Acknowledgements This work was partially supported by the National Science Foundation Grants IIS-1664629 and CAREER.

Appendix A

A.1 The Proof of Theorem 1

Before proving Theorem 1, we first prove the following lemma. Without loss of generality, we use W_k as an example and omit the subscript k in our discussion.

Lemma 1 *For any walker in MWC, if \mathbf{P} is stochastic, irreducible and aperiodic (SIA), then for any $\tau \geq 1$, $\mathbb{P}^{(1,\tau)} = \mathbb{P}^{(1)}\mathbb{P}^{(2)} \dots \mathbb{P}^{(\tau)}$ is also SIA.*

Proof Based on Eq. (5), for any $\tau \geq 1$, $\mathbb{P}^{(\tau)}$ is stochastic, so is $\mathbb{P}^{(1,\tau)}$ since the product of stochastic matrices is also stochastic.

Furthermore, we have

$$\begin{aligned} \mathbb{P}^{(1,\tau)} &= \mathbb{P}^{(1,\tau-1)}\mathbb{P}^{(\tau)} \\ &= \mathbb{P}^{(1,\tau-1)}[\alpha\mathbf{P} + (1 - \alpha)\mathbf{1}(\mathbf{v}^{(\tau-1)})^\top] \end{aligned}$$

$$\begin{aligned}
 &= \alpha \mathbb{P}^{(1, \tau-1)} \mathbf{P} + (1 - \alpha) \mathbf{1}(\mathbf{v}^{(\tau-1)})^\top \\
 &= \alpha^\tau \mathbf{P}^\tau + \sum_{m=1}^{\tau-1} \alpha^m (1 - \alpha) \mathbf{1}(\mathbf{v}^{(\tau-1-m)})^\top \mathbf{P}^m + (1 - \alpha) \mathbf{1}(\mathbf{v}^{(\tau-1)})^\top
 \end{aligned}$$

Due to the last term, $(1 - \alpha) \mathbf{1}(\mathbf{v}^{(\tau-1)})^\top$, we know that there is at least one column of $\mathbb{P}^{(1, \tau)}$ whose entries are all positive. Thus, $\mathbb{P}^{(1, \tau)}$ is irreducible (Please refer to Corollary 4 in [31]).

The self-loop represented by the diagonal of $(1 - \alpha) \mathbf{1}(\mathbf{v}^{(\tau-1)})^\top$ can guarantee that $\mathbb{P}^{(1, \tau)}$ is aperiodic. □

If T exists, $\mathbb{P}^{(\tau+1)} = \mathbb{P}^{(\tau+T+1)}$ ($\tau \geq \tau_p$), since $\mathbf{v}^{(\tau)} = \mathbf{v}^{(\tau+T)}$ in Eq. (3). Thus, $\mathbb{P}^{(\tau+1, \tau+T)} = \mathbb{P}^{(\tau+T+1, \tau+2T)}$. Now we only discuss the sequence $\{\mathbb{P}^{(\tau)}\}_{\tau=\tau_p+1}^\infty$ after τ_p , and define

$$\mathcal{P} = \mathbb{P}^{(\tau_p+1, \tau_p+T)} = \mathbb{P}^{(\tau_p+T+1, \tau_p+2T)}$$

Then we know that the chain $\{\mathcal{P}\}$ is homogeneous. Suppose the graph has finite nodes set, and it's undirected, connected, and its corresponding \mathbf{P} is SIA. Then based on Lemma 1, in MWC, the modified transition matrices $\mathbb{P}^{(\tau)}$ and \mathcal{P} are also SIA. Then we have $\lim_{n \rightarrow \infty} \|\mathcal{P}^{n+1} - \mathcal{P}^n\|_\infty = 0$ [10,14].

Proof of Theorem 1 For $1 \leq \mu \leq T$, we have

$$\begin{aligned}
 &\| \mathbf{x}^{(\tau_p+(n+1)T+\mu)} - \mathbf{x}^{(\tau_p+nT+\mu)} \|_1 \\
 &= \| (\mathcal{P}^{n+1})^\top \mathbf{x}^{(\tau_p+\mu)} - (\mathcal{P}^n)^\top \mathbf{x}^{(\tau_p+\mu)} \|_1 \\
 &\leq \| \mathcal{P}^{n+1} - \mathcal{P}^n \|_\infty \| \mathbf{x}^{(\tau_p+\mu)} \|_1 \\
 &\leq \| \mathcal{P}^{n+1} - \mathcal{P}^n \|_\infty
 \end{aligned}$$

For a sufficient large n such that $\mathcal{P}^{n+1} = \mathcal{P}^n = \mathcal{Q}$, let $\tau'_c = \tau_p + nT$.

Then we have $\mathbf{x}^{(\tau'_c+T+\mu)} = \mathbf{x}^{(\tau'_c+\mu)}$. That is, for any $\tau \geq \tau'_c$, $\mathbf{x}^{(\tau+T)} = \mathbf{x}^{(\tau)}$.

Next, we estimate τ'_c . To reach a computational tolerance $\epsilon > 0$, i.e., $\|\mathcal{P}^{n+1} - \mathcal{P}^n\|_\infty < \epsilon$, a rough estimate of n is $\log \epsilon / \log(\alpha^T)$ [19], because

$$\begin{aligned}
 \mathcal{P} &= \mathbb{P}^{(\tau_p+1, \tau_p+T)} \\
 &= \alpha^T \mathbf{P}^T + \sum_{m=0}^{T-1} \alpha^m (1 - \alpha) \mathbf{1}(\mathbf{v}^{(\tau_p+T-1-m)})^\top \mathbf{P}^m
 \end{aligned}$$

Then $\tau'_c = \tau_p + nT \leq \tau_p + \lceil \log \epsilon / \log \alpha \rceil$.

For the entire group of K walkers, we set τ_c as the largest τ'_c of all walkers and we have $\tau_p \leq \tau_c \leq \tau_p + \lceil \log \epsilon / \log \alpha \rceil$. □

A.2 The Proof of Theorem 2

Theorem 2 directly follows from the following lemma.

Lemma 2 [30] *Let $\Omega = \{\mathbb{A}^{(1)}, \dots, \mathbb{A}^{(\tau)}\}$ be a set of square stochastic matrices of the same order such that for any $m \geq 1$, the product $\mathbb{B} = \mathbb{A}^{(i_1)} \dots \mathbb{A}^{(i_m)}$ ($1 \leq i_j \leq \tau$ for $1 \leq j \leq m$) is SIA. Then for any $\epsilon > 0$, there exists an integer $\nu(\epsilon)$ such that any \mathbb{B} of length $n \geq \nu(\epsilon)$ satisfies $\delta(\mathbb{B}) < \epsilon$.*

Proof of Theorem 2 Let Ω be the set that contains all possible modified transition matrices in MWC. From Lemma 1, we know that the product of matrices from Ω with any length and order is SIA. Based on Lemma 2, we know that for any $\epsilon > 0$, there exists an integer $\nu(\epsilon)$ such that when $\tau_s \geq \nu(\epsilon)$, $\delta(\mathbb{P}^{(1, \tau_s)}) < \epsilon$.

For $\nu(\epsilon)$, based on Lemma 2 in [30], we know that $\delta(\mathbb{P}^{(1, \tau_s)}) \leq \prod_{\tau=1}^{\tau_s} \lambda(\mathbb{P}^{(\tau)})$. Because the rows in $\mathbf{1}(\mathbf{v}^{(\tau-1)})^\top$ are identical, then

$$\begin{aligned} \lambda(\mathbb{P}^{(\tau)}) &= 1 - \min_{v, v'} \sum_w \min\{\mathbb{P}^{(\tau)}(v, w), \mathbb{P}^{(\tau)}(v', w)\} \\ &= 1 - \min_{v, v'} \sum_w \min\{[\alpha\mathbf{P} + (1 - \alpha)\mathbf{1}(\mathbf{v}^{(\tau-1)})^\top](v, w), \\ &\quad [\alpha\mathbf{P} + (1 - \alpha)\mathbf{1}(\mathbf{v}^{(\tau-1)})^\top](v', w)\} \\ &= 1 - \alpha \min_{v, v'} \sum_w \min\{\mathbf{P}(v, w), \mathbf{P}(v', w)\} - (1 - \alpha) \\ &= \alpha(1 - \min_{v, v'} \sum_w \min\{\mathbf{P}(v, w), \mathbf{P}(v', w)\}) \\ &= \alpha\lambda(\mathbf{P}) \end{aligned}$$

So $\delta(\mathbb{P}^{(1, \tau_s)}) \leq (\alpha\lambda(\mathbf{P}))^{\tau_s}$. To satisfy the tolerance ϵ , we can set $\nu(\epsilon) = \log \epsilon / \log(\alpha\lambda(\mathbf{P}))$. □

For a stochastic matrix \mathbf{P} , the measure $\lambda(\mathbf{P})$ shows the difference of its row vectors and we have $0 \leq \lambda(\mathbf{P}) \leq 1$. In practice, the transition matrix \mathbf{P} is sparse and has block structure. We can select two rows from \mathbb{P} that have no overlapping nonzero entries to make $\lambda(\mathbf{P})$ reach the maximum value 1. So in the worst case, we can set $\tau_s = \lfloor \log \epsilon / \log \alpha \rfloor$ to stop the algorithm.

A.3 The Proof of Theorem 3

In this section, we analyze the error bound on the difference between the exact and estimated probability vectors for W_k in the $(\tau + 1)$ th group iteration. We first define the *probability flow*, and provide a tight error bound in Theorem 4. The proof of Theorem 3 then follows. For simplicity, we omit the subscript “ k ” and use $\mathbf{x}^{(\tau+1)}$ and $\hat{\mathbf{x}}^{(\tau+1)}$ to represent the exact and estimated probability vectors, respectively.

Let $U^{(\tau)}$ represent the set of nodes whose probability will be updated in the $(\tau + 1)$ th iteration, i.e., $U^{(\tau)} = C^{(\tau)} \cup \Gamma(C^{(\tau)})$, where $\Gamma(C)$ represents the direct neighbors of the nodes in C , and let $S^{(\tau)}$ represent the remaining nodes. We only update the scores for nodes in $U^{(\tau)}$ in the $(\tau + 1)$ th iteration, i.e.,

$$\hat{\mathbf{x}}^{(\tau+1)}(i) = \begin{cases} \text{update based on Eq. (2)} & \text{if } i \in U^{(\tau)}; \\ \mathbf{x}^{(\tau)}(i) & \text{otherwise.} \end{cases} \tag{11}$$

The estimation error is thus $\Delta = \|\frac{\hat{\mathbf{x}}^{(\tau+1)}}{\|\hat{\mathbf{x}}^{(\tau+1)}\|_1} - \mathbf{x}^{(\tau+1)}\|_1$

Definition 5 The *probability flow* from a set of nodes B to another set A in the $(\tau + 1)$ th group iteration is defined as

$$F_{B \rightarrow A} = \sum_{g \in A} \sum_{i \in N_g \cap B} \mathbf{P}(i, g)\mathbf{x}^{(\tau)}(i),$$

where N_g represents the direct neighbors of g .

The probability flows during the $(\tau + 1)$ th iteration are (we omit the superscript “ (τ) ” for $U^{(\tau)}$ and $S^{(\tau)}$ below):

$$\begin{cases} a := F_{U \rightarrow S} = \alpha \sum_{g \in S} \sum_{i \in N_g \cap U} \mathbf{P}(i, g) \mathbf{x}^{(\tau)}(i); \\ b := F_{S \rightarrow S} = \alpha \sum_{g \in S} \sum_{j \in N_g \cap S} \mathbf{P}(j, g) \mathbf{x}^{(\tau)}(j); \\ c := F_{S \rightarrow U} = \alpha \sum_{g \in U} \sum_{j \in N_g \cap S} \mathbf{P}(j, g) \mathbf{x}^{(\tau)}(j). \end{cases} \tag{12}$$

Theorem 4

$$\Delta \leq \frac{1}{1 - \gamma} [(\gamma + |\gamma|)(1 - \gamma - \beta) + 2\beta],$$

where $\gamma = 1 - \|\hat{\mathbf{x}}^{(\tau+1)}\|_1 = a + b - \beta$ and $\beta = (c + b)/\alpha$.

Proof According to Eqs. (11) and (12), we have

$$\begin{aligned} \gamma &= 1 - \|\hat{\mathbf{x}}^{(\tau+1)}\|_1 = 1 - \left\| \begin{bmatrix} \mathbf{x}_U^{(\tau+1)} \\ \mathbf{x}_S^{(\tau)} \end{bmatrix} \right\|_1 \\ &= \|\mathbf{x}_S^{(\tau+1)}\|_1 - \|\mathbf{x}_S^{(\tau)}\|_1 \\ &= F_{U \rightarrow S} + F_{S \rightarrow S} - \sum_{g \in S} \mathbf{x}^{(\tau)}(g) \\ &= a + b - (c + b)/\alpha \\ &= a + b - \beta \end{aligned}$$

where \mathbf{x}_U and \mathbf{x}_S are the sub-vector projections of \mathbf{x} on U and S , respectively, and $\beta = (c + b)/\alpha$.

Thus, we have

$$\begin{aligned} \Delta &= \left\| \frac{\hat{\mathbf{x}}^{(\tau+1)}}{\|\hat{\mathbf{x}}^{(\tau+1)}\|_1} - \mathbf{x}^{(\tau+1)} \right\|_1 \\ &= \left\| \begin{bmatrix} \mathbf{x}_U^{(\tau+1)} \\ \mathbf{x}_S^{(\tau)} \end{bmatrix} / (1 - \gamma) - \begin{bmatrix} \mathbf{x}_U^{(\tau+1)} \\ \mathbf{x}_S^{(\tau+1)} \end{bmatrix} \right\|_1 \\ &= \frac{|\gamma|}{1 - \gamma} \|\mathbf{x}_U^{(\tau+1)}\|_1 + \left\| \frac{\mathbf{x}_S^{(\tau)}}{1 - \gamma} - \mathbf{x}_S^{(\tau+1)} \right\|_1 \\ &\leq \frac{|\gamma|}{1 - \gamma} (1 - \|\mathbf{x}_S^{(\tau+1)}\|_1) + \frac{\|\mathbf{x}_S^{(\tau)}\|_1}{1 - \gamma} + \|\mathbf{x}_S^{(\tau+1)}\|_1 \\ &= \frac{|\gamma|}{1 - \gamma} (1 - \gamma - \beta) + \frac{\beta}{1 - \gamma} + (\gamma + \beta) \\ &= \frac{1}{1 - \gamma} [(\gamma + |\gamma|)(1 - \gamma - \beta) + 2\beta] \end{aligned}$$

□

The proof of Theorem 3 is as follows.

Proof (1) If $\gamma \leq 0$, we have

$$\begin{aligned} \Delta &\leq \frac{2\beta}{1 - \gamma} \leq 2\beta = 2 \sum_{g \in S} \mathbf{x}^{(\tau)}(g) \\ &\leq 2 \sum_{g \in V \setminus C^{(\tau)}} \mathbf{x}^{(\tau)}(g) = 2(1 - Pr(C^{(\tau)})) \leq 2(1 - \theta) \end{aligned}$$

Algorithm 4: UPDATE-Core

Input: $P, K, \alpha, \mathbf{x}_k^{(\tau)}, inodes, \theta$ **Output:** $\mathbf{x}_k^{(\tau+1)}$

- 1 Find $C_k^{(\tau)}$ by BFS starting from the influential nodes of W_k ;
- 2 $U_k^{(\tau)} = C_k^{(\tau)} \cup \Gamma(C_k^{(\tau)})$;
- 3 **for** $i \in U_k^{(\tau)}$ **do** Update $\hat{\mathbf{x}}_k^{(\tau+1)}(i)$ according to Eq. (2);
- 4 **for** $i \notin U_k^{(\tau)}$ **do** $\hat{\mathbf{x}}_k^{(\tau+1)}(i) = \mathbf{x}_k^{(\tau)}(i)$;
- 5 **return** $\hat{\mathbf{x}}_k^{(\tau+1)} / \|\hat{\mathbf{x}}_k^{(\tau+1)}\|_1$.

(2) If $\gamma > 0$, we have $\Delta \leq 2(\beta + \gamma) = 2(a + b)$.

Let $\delta U = \{g \in U \mid \exists i \in S, s.t., (i, g) \in E\}$ be the set of boundary nodes of U . We have $a + b \leq F_{\delta U \cup S \rightarrow S} \leq F_{V \setminus C^{(\tau)} \rightarrow V} \leq 1 - \theta$.

Thus, $\Delta \leq 2(1 - \theta)$. □

Algorithm 4 shows the overall process of the partial node set updating strategy for walker W_k , which can be used to replace the UPDATE in Algorithm 2. The algorithm first finds the core node set $C_k^{(\tau)}$ by a breadth first search starting from the influential nodes of walker W_k (Line 1). In Line 2, the updating node set $U_k^{(\tau)}$ can be obtained as the union of the nodes in $C_k^{(\tau)}$ and their direct neighbors. Lines 3 and 4 update the node visiting probabilities according to Eq. (11). Line 5 returns the normalized probability vector.

The breadth first search costs $O(d|U_k^{(\tau)}|)$, where d is the average degree of nodes in $U_k^{(\tau)}$. In each iteration, the updating time is reduced from $O(|V| + |E|)$ to $O(d|U_k^{(\tau)}|)$ for W_k .

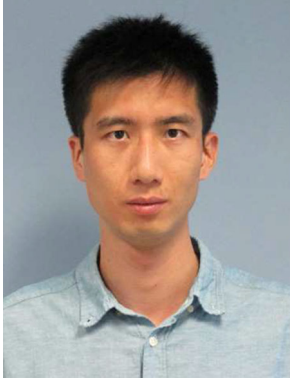
References

1. Alamgir M, Von Luxburg U (2010) Multi-agent random walks for local clustering on graphs. In: *ICDM*
2. Alon N, Avin C, Koucky M, Kozma G, Lotker Z, Tuttle MR (2008) Many random walks are faster than one. In: *SPPA*
3. Andersen R, Chung F, Lang K (2006) Local graph partitioning using PageRank vectors. In: *FOCS*
4. Andersen R, Lang KJ (2006) Communities from seed sets. In: *WWW*
5. Barnard K, Duygulu P, Forsyth D, Freitas Nd, Blei DM, Jordan MI (2003) Matching words and pictures. *J Mach Learn Res* 3:1107–1135
6. Brandes U (2001) A faster algorithm for betweenness centrality. *J Math Sociol* 25(2):163–177
7. Cooper C, Frieze A, Radzik T (2009) Multiple random walks and interacting particle systems. In: *International colloquium on automata, languages and programming*. Springer, New York, pp 399–410
8. Guan Z, Wu J, Zhang Q, Singh A, Yan X (2011) Assessing and ranking structural correlations in graphs. In: *SIGMOD*
9. Guillaumin M, Mensink T, Verbeek J, Schmid C (2009) Tagprop: discriminative metric learning in nearest neighbor models for image auto-annotation. In: *ICCV*
10. Hajnal J, Bartlett M (1958) Weak ergodicity in non-homogeneous Markov chains. In: *Mathematical proceedings of the Cambridge Philosophical Society*
11. Haveliwala TH (2002) Topic-sensitive pagerank. *WWW*
12. He K, Shi P, Hopcroft J, Bindel D (2016) Local spectral diffusion for robust community detection. In: *SIGKDD twelfth workshop on mining and learning with graphs*
13. He K, Sun Y, Bindel D, Hopcroft J, Li Y (2015) Detecting overlapping communities from local spectral subspaces. In: *ICDM*
14. Isaacson DL, Madsen RW (1976) *Markov chains, theory and applications*, vol 4. Wiley, New York
15. Jeh G, Widom J (2002) SimRank: a measure of structural-context similarity. In: *KDD*
16. Kloster K, Gleich DF (2014) Heat kernel based community detection. In: *KDD*
17. Kloumann IM, Kleinberg JM (2014) Community membership identification from small seed sets. In: *KDD*

18. Lancichinetti A, Fortunato S, Radicchi F (2008) Benchmark graphs for testing community detection algorithms. *Phys Rev E* 78(4):046110
19. Langville AN, Meyer CD (2004) Deeper inside PageRank. *Intern Math* 1(3):335–380
20. Lee C, Jang W-D, Sim J-Y, Kim C-S (2015) Multiple random walkers and their application to image cosegmentation. In: *CVPR*
21. Lee S-H, Jang W-D, Park B K, Kim C-S (2016) RGB-D image segmentation based on multiple random walkers. In: *ICIP*
22. Li Y, He K, Bindel D, Hopcroft JE (2015) Uncovering the small community structure in large networks: a local spectral approach. In: *WWW*
23. Lu D, Zhang H (1986) *Stochastic process and applications*. Tsinghua University Press, Beijing
24. Pan J-Y, Yang H-J, Faloutsos C, Duygulu P (2004) Automatic multimedia cross-modal correlation discovery. In: *KDD*
25. Sarkar P, Moore A (2007) A tractable approach to finding closest truncated-commute-time neighbors in large graphs. In: *UAI*
26. Schaeffer SE (2007) Graph clustering. *Comput Sci Rev* 1(1):27–164
27. Spielman DA, Teng S-H (2013) A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *SIAM J Comput* 42(1):1–26
28. Tong H, Faloutsos C, Pan J-Y (2006) Fast random walk with restart and its applications. In: *ICDM*
29. Whang JJ, Gleich DF, Dhillon IS (2013) Overlapping community detection using seed set expansion. In: *CIKM*
30. Wolfowitz J (1963) Products of indecomposable, aperiodic, stochastic matrices. *Linear Algebra Appl* 14(5):733–737
31. Wu CW (2005) On bounds of extremal eigenvalues of irreducible and m-reducible matrices. *Linear Algebra Appl* 402:29–45
32. Wu X-M, Li Z, So AM, Wright J, Chang S-F (2012) Learning with partially absorbing random walks. *NIPS*
33. Wu Y, Jin R, Li J, Zhang X (2015) Robust local community detection: on free rider effect and its elimination. In: *VLDB*
34. Yang J, Leskovec J (2012) Defining and evaluating network communities based on ground-truth. In: *ICDM*
35. Yin H, Benson AR, Leskovec J, Gleich DF (2017) Local higher-order graph clustering. In: *KDD*
36. Yu W, Lin X, Zhang W, Chang L, Pei J (2013) More is simpler: effectively and efficiently assessing node-pair similarities based on hyperlinks. In: *VLDB*
37. Zhang J, Tang J, Ma C, Tong H, Jing Y, Li J (2015) Panther: fast top-k similarity search on large networks. In: *KDD*
38. Zhou D, Zhang S, Yildirim MY, Alcorn S, Tong H, Davulcu H, He J (2017) A local algorithm for structure-preserving graph cut. In: *KDD*
39. Zhu X, Goldberg AB (2009) Introduction to semi-supervised learning. *Synth Lect Artif Intell Mach Learn* 3(1):1–130

Yuchen Bian is a Ph.D. candidate in College of Information Sciences and Technology at The Pennsylvania State University. His research interests cover machine learning and data mining. He is working on projects to design efficient algorithms to detect information in complex networks.

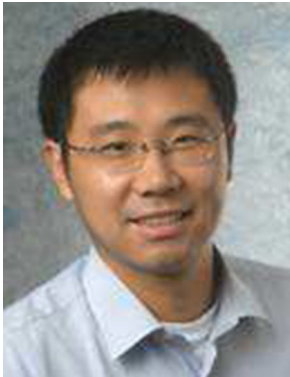




Jingchao Ni received the Ph.D. degree from the Pennsylvania State University. He is a researcher in the Data Science and Systems Research Department of NEC Laboratories America. His research interests are data mining and machine learning.



Wei Cheng received the Ph.D. degree from UNC at Chapel Hill, in 2015. He is a research staff member in the Data Science and System Research Department, NEC Laboratories America. His research interests include data mining, machine learning, and bioinformatics.



Xiang Zhang is an Associate Professor in the College of Information Sciences and Technology at the Pennsylvania State University. His research bridges the areas of big data, data mining, machine learning, and databases. He is particularly interested in developing algorithms and models for analyzing large data sets generated in social, biological, and medical domains.