Directional Component Detection via Markov Clustering in Directed Networks

Yu-Keng Shih
Dept. of Computer Science and Engineering
The Ohio State University
shihy@cse.ohio-state.edu

Sungmin Kim
Dept. of Statistics
The Ohio State University
kim.2774@osu.edu

Tao Shi
Dept. of Statistics
The Ohio State University
taoshi@stat.osu.edu

Srinivasan Parthasarathy
Dept. of Computer Science and Engineering
The Ohio State University
srini@cse.ohio-state.edu

ABSTRACT

Community detection has been one of the fundamental problems in network analysis. Past studies mainly focused on detecting communities in undirected networks. However, several real networks are directed, e.g. World Wide Web, paper citation networks and Twitter's follower-followee network. Although some studies proposed algorithms for directed networks, few of them considered that nodes play two different roles, source and terminal, in a directed network. In this paper, we adopt a novel concept of communities, *directional components*, and propose a new algorithm based on Markov Clustering to detect directional components in a directed network. We then compare our algorithm, Dual R-MCL, on synthetic networks with two recent algorithms also designed for detecting directional components. We show that Dual R-MCL can detect directional components with significantly higher accuracy than the two other algorithms. Additionally, Dual R-MCL is 3x~25x faster than the two other algorithm on networks with ten thousand nodes when achieving high accuracy. The results show that Dual R-MCL is robust with noise and can efficiently detect directional components.

Categories and Subject Descriptors

G.2.2 [**Graph Theory**]: Graph algorithms

Keywords

Directional Component

1. INTRODUCTION

Many real world problems can be effectively modeled as complex relationship in networks (graphs) where nodes represent entities of interest and edges mimic the interactions or relationships among those nodes. The study of such complex relationship networks, recently referred to as network science, can provide insight into their structures and properties [14]. One particularly interesting area in studies of network structures is searching for important sub-networks which are usually called communities. A community in a network is typically characterized by a group of nodes that have more edges (links) connected within the community than connected to out of the community [8]. Community detection is in growing attention not only because it leads to understanding of the complex network structure, but also it allows further analysis such as studies on information flows on in networks, evolution of networks and visualization of networks.

However, this typical definition of communities may not discover real groups of nodes in a directed network. A recent study [19] shows that nodes with similar out-links nodes (nodes which the current nodes point to) and in-link nodes (nodes which point to the current nodes) tend to have similar properties, and using such out-link and in-link similarity might form more meaningful communities than the typical communities. Therefore, different definitions of communities might also discover nodes with similar properties in a directed network.

In many practical applications, there is a large number of networks that are directed in nature, such as the World Wide Web, Tweeter's follower-followee network, and paper citation networks. Even though a few algorithms developed for undirected networks can be extended to apply on directed networks [6, 8, 5], the notion of community in undirected networks cannot be simply translated to the directed ones. A common approach to handle directed networks is symmetrizing the adjacency matrix by removing direction of edges and treating the resulted matrix as from an undirected network. However, it is not uncommon to see that ignoring the direction of edges results in abnormal communities [13].

The aim of the present work is to develop efficient community detection algorithms that explicitly incorporate the direction of links. Compared with in-depth studies of community structures in undirected networks [6, 8, 5], community detection in directed networks has not been as fruitful. One particular difficulty in studying the structure of directed networks is the lack of a clear definition of the connectivity between each pair of nodes. As a result, it is hard to define communities due to the asymmetry nature of the edges. An existing work [10] points out the importance of recognizing

MLG '13 August 11 2013, Chicago, IL, USA
 ACM 978-1-4503-2322-2/13/08 ...$15.00.

the dual roles, source and terminal of edges, which nodes play in a directed network. In this paper, we concentrate on directly incorporating directed edges in analysis and we start with a novel definition of community, directional component, which contains a source part and a terminal part based on the connectivity between nodes following a path of the edges in alternative directions. A recent study [9] has reported that several existing community detection algorithms for a directed network, such as Infomap [16] and DI-SIM [15], cannot effectively detect directional components.

In order to discover dense directional components in a directed network, we begin at focusing on a stochastic-flow based algorithms, Markov Clustering (MCL) [7]. MCL and its variant, R-MCL [17, 20], are a class of simple yet elegant algorithms based on the natural phenomenon of flow or transition probabilities and have been successfully deployed for community discovery in a wide variety of networks [3, 2, 4]. Beyond their simplicity and strong mathematical basis, these algorithms are known to be robust to noise and can handle the unique topological challenges posed by scale-free networks [3]. MCL detects communities by grouping nodes with the same attractor as a cluster. Although the communities MCL detects are according to the traditional definition, with suitable manipulating flows and attractors, MCL can be used to detect directional components. The new algorithm, *Dual R-MCL*, can successfully detect group of nodes play the source role and terminal role separately, and more importantly, match the nodes playing source roles to nodes playing terminal roles to form directional components.

We test Dual R-MCL on synthetic networks with gold standard directional communities. In our preliminary result, Dual R-MCL is compared with two recent algorithms which are also designed for detecting directional components. The experimental results show that Dual R-MCL can achieve higher accuracy in any noise levels. Additionally, Dual R-MCL can more efficiently process a moderately large network than the other two algorithms, achieving 3x~25x speed when the accuracy is high.

2. DIRECTIONAL COMPONENT

The nodes and edges in a directed network are often presented by a graph $G = (V, E)$, where V is the set of nodes, E is the set of directed edges, and $n = |V|$. Each edge from node v_i to v_j is denoted by (v_i, v_j). $w((v_i, v_j))$ is the weight of an edge (v_i, v_j). Let A be the n by n adjacency matrix of the network such that $A_{(j,i)} = w((v_i, v_j))$. Note that here A is the transpose of the usual definition of an adjacency matrix in order to be consistent with the definition of the flow matrix in Markov Clustering, which is introduced in Section 3.1.

Two types of connectivity between nodes have been studied in the literature of directed networks. *Weak connectivity* defines two nodes s and t ($s, t \in V$) as weakly connected if they reach each other through a path e_1 , e_2,...,e_l, regardless of the directions of edges in the path. Meanwhile, *strong connectivity* takes in account of the directions of the edges in the path and claims nodes s and t are strongly connected only if the path (e_1, e_2,..., e_l) also satisfies $v^s(e_1) = s$, $v^t(e_l) = t$, $v^t(e_k) = v^s(e_{k+1})$, $k = 1, ..., l-1$, where $v^s(e)$ and $v^t(e)$ are the source node and destination node of the edge e respectively.

DEFINITION 1. *Two nodes s and t are **D-connected**, denoted by $s \to t$, if there exists a path of edges (e_1,...,e_{2m1}), m is a positive integer, satisfying $v^s(e_1) = s, v^t(e_{2m1}) = t$ and*

$$\begin{cases} v^t(e_{2k-1}) = v^t(e_{2k}) & \textit{(common terminal nodes)} \\ v^s(e_{2k}) = v^s(e_{2k+1}) & \textit{(common source nodes)} \end{cases}$$

for $k = 1, 2, ..., \max\{m-1, 1\}$.

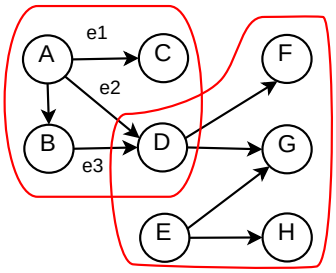

Figure 1: A toy example of directional components. The red circles are two maximal directional components.

D-connectivity follows the edges in alternating directions, one forward and then backward. We call this sequence of edges a D-connected path. Figure 1 provides an illustration of D-connectivity. B $\to$ C via the sequence of edges (e_3 , e_2 , e_1). D-connectivity recognizes the two different roles of nodes, sources and terminals, which leads to a new type of community structure, directional component:

DEFINITION 2. *A **directional component** (DC) consists a source node set S and a terminal node set T ($S, T \subset V$) and $\forall s \in S$ and $\forall t \in T$, $s \to t$. We call S and T the source part and terminal part of the directional component and denote $DC \equiv (S, T)$.*

DEFINITION 3. *A **maximal directional component** is a directional component in which S and T are the maximal subsets of nodes such that any pair of nodes (s, t), $s \in S$, $t \in T$, are D-connected ($s \to t$).*

The concept of directional components provides a potential way to partition a directed network into smaller communities. First, a node v in a directional component may belong to either the source part S or the terminal part T or both. Second, in a directed network that contains multiple directional components $DC_1, DC_2, ..., DC_k$ any node $v \in V$ can only belong to one of the source set S. In other words, the source parts $S_1, ..., S_k$ are disjoint and the same holds for $T_1, ..., T_k$. However, it is possible that $v \in S_i$ and $v \in T_j$ and $i \neq j$. Figure 1 indicates a directed network with two (maximal) directional components, $DC_1 = (S = A, B, T = B, C, D)$ and $DC_2 = (S = D, E, T = F, G, H)$. Note that node D belong to two directional component and node B belong to both source part and terminal part of DC_1. This two-way partition of nodes respects the asymmetric property of the directed network.

Finding maximal directional components can be achieved through a simple searching algorithm of computational complexity $O(|V| + |E|)$. Identification of one maximal directional component is done by iterations of adding nodes into the source part and the terminal part. Starting with picking one node with positive out-degree as a member of a source part, the algorithm iterates between steps:

1. For each node in source part, add all nodes pointed to by this node to the terminal part;
2. For each node in terminal part, add all nodes pointing to this node to the source part;

The iteration runs until no more nodes need be added in either source part or terminal part.

Once the iteration is terminated, one may remove all edges of the identified directional component from the original network and repeat the iteration to identify another directional component on the reduced network. One drawback of this direct searching algorithm is that it usually detects only a few large but sparse directional components, even when the network has much finer community structures. This phenomenon is due to fact that it is unrealistic to expect absolutely no edges between those small but dense directional

Algorithm 1 R-MCL

Input: The canonical flow matrix M_G, the balance parameter b, the inflation parameter r.
Output: A set of clusters $\mathcal{C}$ and their corresponding attractors.
1: $M = M_G$ //Initialize M
2: **repeat**
3: $M_R = RegularizationMatrix(M, M_G, b)$
4: $M = M * M_R$ //Regularize operation
5: $M = M.^r$ //Inflate: Raise each entry to the power of r.
6: $M = Prune(M)$ //Remove insignificant values.
7: **until** M converges //$iter$-time execution of R-MCL
8: Interpreting M as clustering result.

components, as it is unrealistic to expect no edges between traditional communities in an undirected network. For example, a direct search on the Cora citation network[1], which presents bibliographic citation between papers in computer science, leads to one giant directional component that covers all nodes. However, a close look at the dataset suggests that there are much more citations between papers in a similar field and much less between papers in different fields. In other words, many smaller sized communities exist, but the algorithm is too strict to detect them. Therefore, we need consider more flexible algorithms that may detect dense directional components with existence of a small number of external edges. In such directional components, nodes in the source part share similar out-link nodes and nodes in the terminal part share similar in-link nodes. Thus, we propose an algorithm which can efficiently cluster nodes based on out-link similarity and in-link similarity separately, and very efficiently match the found source part and terminal part.

3. METHODS

3.1 Terminology of MCL

A column stochastic matrix M is a n by n matrix which can be interpreted as the transition probabilities of a random walk (or a Markov chain) defined on the network. Specifically, $M_{(i,j)}$ represents the probability of a transition from v_j to v_i. We also refer to the transition probability from v_i to v_j as the flow from v_i to v_j. Given an adjacency matrix A, a corresponding canonical flow matrix $flow(A) = M_G$ is column-normalize A, i.e.,

$$M_{G(i,j)} = \begin{cases} \frac{A_{(i,j)}}{\sum_{x=1}^{n} A_{(x,j)}} & \text{if } \sum_{x=1}^{n} A_{(x,j)} > 0 \\ 0 & \text{otherwise} \end{cases}$$

3.2 Prior work: R-MCL

MCL and R-MCL are community detection algorithms based on a simulation of stochastic flows on the network. MCL consists of two operations on a stochastic matrix: *Expand* and *Inflate*. The Expand operation is simply $M = M * M$, and the Inflate operation raises each entry in the matrix M to the inflation parameter r ($r > 1$, and typically set to 2) followed by re-normalizing the sum of each column to 1. These two operations are applied in alternation iteratively, starting with $M = M_G$, where $M_G = flow(A + I)$. The addition of self-loops to the adjacency matrix avoids dependence of the flow distribution on the length of the random walk simulated in MCL, besides ensuring at least one non-zero entry per column. In R-MCL, Expand is replaced by *Regularize*, which is $M = M * M_G$. The Regularize step updates a node flow to the weighted average of the node's neighbors' (or the node's out-link nodes' in a directed network) current flow M. $M_G(i, j)$ can be considered as the weight of node j when updating node i's flow. The Expand and Regularize operations spread the flow out of a vertex to potentially new nodes. This has the effect of enhancing within-cluster flows as there are more paths between two nodes that are in the same cluster than between those in different clusters. At the start of this process, the distribution of flows out of a node is relatively smooth and uniform; as more iterations are executed, all the nodes within a tightly-linked group of nodes will start to flow to one node within the group. This allows us to identify all the nodes that flow to the same *attractor node* as belonging to one cluster. Note that it is not necessary for an attractor to be contained by its representing cluster. [20] additionally introduce a balance parameter into R-MCL. When b is larger, the flow to a node which already attracts a large number of nodes would be inhibited, so the resulting clusters are more balanced in size. The pseudo code is shown in Algorithm 1. For a complete description of MCL and R-MCL, the reader is referred elsewhere [7, 17, 20].

[1]http://www.cs.umass.edu/~mccallum/code-data.html

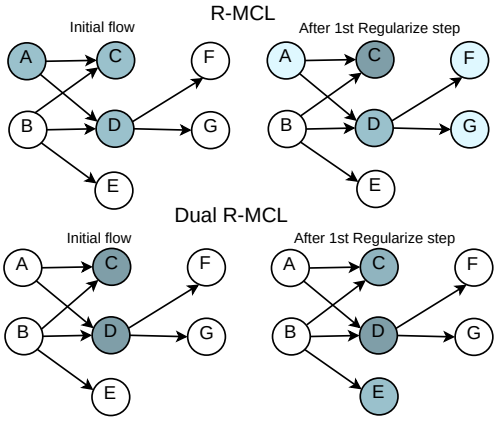

Figure 2: An example illustrates how the flow in R-MCL and Dual R-MCL goes in a directed network. The color of each node represent node A's current flow: White represents 0 and darker blue represent a larger value.

3.3 Dual R-MCL

If R-MCL is executed on directed network, the flow follows the direction of each edge. The attractor tends to be a node without out-link, and the attractor's representing cluster mainly contain nodes strongly connecting to the attractor, no matter how far the node is from the attractor. However, as stated earlier, nodes play the role of source and terminal in a directed network, and in some directed networks, a node shares similar property only with other nodes in a dense directional component. In order to detect dense directional components in a directed network, we propose Dual R-MCL.

The main intuition of Dual R-MCL is (1) to cluster nodes separately based out-link similarity and in-link similarity to obtain source parts and terminal parts respectively and (2) to efficiently match the clusters from each clustering result to form directional components. We first note that our previous study propose an algorithm which can cluster nodes in a directed network based on their neighborhood similarity [18]. In order to detect source parts and terminal parts in a network, we can simply modify that algorithm to construct an out-link similarity matrix and an in-link similarity matrix, and then cluster nodes separately based on the two matrices by any clustering algorithm. However, this procedure cannot effectively match these two parts into directional components. We

Algorithm 2 Dual R-MCL

Input: The adjacency matrix A, the balance parameter b, the inflation parameter r.
Output: A set of directional components.

$M_G^{out} = flow(A)$
$M_G^{in} = flow(A^T)$
$M_G^{outSim} = flow(A^T * A)$
$\mathbb{S} = RMCL(M_G^{outSim}, b, r)$ but initialize M to M_G^{out} //obtain the set of source parts
$M_G^{inSim} = flow(A * A^T)$
$\mathbb{T} = RMCL(M_G^{inSim}, b, r)$ but initialize M to M_G^{in} //obtain the set of terminal parts
for all $S \in \mathbb{S}$ **do**
  Find $T \in \mathbb{T}$ such that $attractor(S) \in T$.
  if $attractor(T) \in S$ **then**
    Add a new directional component $DC = (S, T)$ to the result.

find that, in R-MCL, if the flow is manipulated via D-connected paths, it is possible to easily match the source parts and terminal parts according to the attractors of them. Specifically, when clustering nodes based on out-link similarity to find source parts, we can manipulate a node A's flow, letting it always point to nodes which A is D-connected to. Therefore, when the flow matrix is converged, node A's attractor is likely to be a node in the corresponding terminal part. Similarly, when clustering nodes based on in-link similarity to find terminal parts, we can manipulate a node B's flow, letting it always point to nodes which are D-connected to B. Therefore, when the flow matrix is converged, node B's attractor is likely to be a node in the corresponding source part.

In order to manipulate the flow as above, when clustering nodes based on out-link similarity, the initial flow matrix is set to $flow(A)$, so the initial flow directly points to each node's out-link nodes. When iteratively regularize the flow matrix, a node's flow should not be updated to the weighted average of the node's out-link nodes' flow as in R-MCL, since this step causes the flow to follow the strong connectivity. Instead, in order to let the flow follow the D-connectivity, a node's flow should be updated to the weighted average of the node's out-link-similar nodes' flow, and the weight is proportion to the out-link similarity, $A^T * A$. Since the sum of weight should be 1, the new Regularize step becomes $M = M * flow(A^T * A)$. Note that here the initial flow matrix does not contain self-loop, so a node's flow does not point to itself and thus after regularize step, a node's flow always points to nodes which this node is D-connected to. Figure 2 shows an example. In R-MCL the initial flow and the canonical flow both contain self-loops, and the flow simply follows the strong connectivity. As a result, the flow will spread to nodes which should not be in the same directional component but strongly connected to. However, in Dual R-MCL, A is out-link-similar to B since they share two common out-link nodes, C and D. Therefore, during Regularize step, A's flow is updated to the weighted average of A's current flow and B's current flow, and the resulting flow after the first Regularized step is shown in the right bottom network. Similarly, when clustering nodes based on in-link similarity, the initial flow matrix is set to $flow(A')$ and the new Regularize step should be $M = M * flow(A * A^T)$. After obtaining the set of source parts and the set of terminal parts, we can simply match a source part S to a terminal part T if the attractor of S is in T and also the attractor of T is in S. Algorithm 2 is the pseudo code of Dual R-MCL.

Figure 3 shows a simple example how Dual R-MCL matches source parts and terminal parts. The red rectangle in the bottom

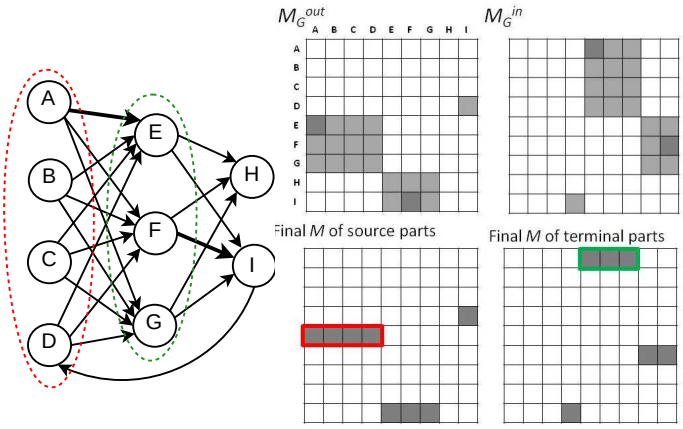

Figure 3: An example illustrates how Dual R-MCL can identify directional components. The wider arrow in the network represents higher weight. White entries in the matrix mean zero flow as a darker entry represents larger flow. The red and green rectangles in the bottom matrices indicates a source part $S = \{A, B, C, D\}$ and a terminal part $T = \{E, F, G\}$ respectively, and they can be matched to form a directional component.

left matrix indicates a source part $\{A, B, C, D\}$ with an attractor E, and the green rectangle in the bottom right matrix indicates a terminal part $\{E, F, G\}$ with an attractor A. Since this source part's attractor is contained by the terminal part and also this terminal part's attractor is contained by the source part, this two parts are matched together to form a directional component. Similarly, we can construct two other directional components $DC_2 = (S = \{E, F, G\}, T = \{H, I\})$ and $DC_3 = (S = \{I\}, T = \{D\})$.

4. PRELIMINARY RESULTS

4.1 Experiment Setting

We performed our experiments on synthetic networks generated by LFR [11]. LFR can randomly construct a power-law network with various tunable parameters. In this study, we set $|V| = 10000$ nodes whose in-degrees follow a power law with decay rate $\tau_1 = -2$ and maximal out-degree 200. The size of the original gold standard communities follow a power law with a decay rate $\tau_2 = -1$ and belong to the range $[40, 200]$. We then tune the proportion of external edges (noise level) μ for all nodes from 0.2 to 0.6 with an interval 0.1 (All algorithms can achieve very high accuracy when $\mu < 0.2$), and set the average out-degree deg_{avg} to 10 or 20.

The gold standard communities of LFR adopts the traditional definition of communities, i.e., nodes in the same community are highly weakly connected, so it ignores the direction of edges. In order to construct gold standard directional component, the original gold standard communities are considered as gold standard source parts. Then, the destination indices of all edges are randomly shuffled, and the original gold standard communities are also shuffled with the same permutation, becoming gold standard terminal parts. Therefore, each node belong to exactly one source part and one terminal part . For example, if LFR originally generates three gold standard communities $\{A, B, C\}$, $\{D, E\}$ and $\{F, G\}$ in a 7-nodes network, and the destination indices are randomly shuffled: $(A, B, C, D, E,$
$F, G)$ is replaced by (C, D, E, A, G, B, H) respectively, then, every edge's destination node should be replaced by the new index,

Table 1: Comparison of Dual R-MCL, L0-harvesting and EN-harvesting. The left value is NMI and the right value is the execution time in seconds. All values are the average among ten networks generated by LFR. $deg_{avg} = 10$

Algorithm	$\mu = 0.2$	$\mu = 0.3$	$\mu = 0.4$	$\mu = 0.5$	$\mu = 0.6$
Dual R-MCL	.9897 / 20	.9710 / 35	.8673 / 53	.2624 / 85*	.0031 / 115*
L0-harvesting	.8658 / 842	.7442 / 1043	.5995 / 1020	.1350 / 943	.0032 / 930
EN-harvesting	.7004 / 234	.5743 / 217	.0865 / 197	.0229 / 160	0 / 185

*$b = 2.0$ is used.

Table 2: Comparison of Dual R-MCL, L0-harvesting and EN-harvesting. The left value is NMI and the right value is the execution time in seconds. All values are the average among ten networks generated by LFR. $deg_{avg} = 20$

Algorithm	$\mu = 0.2$	$\mu = 0.3$	$\mu = 0.4$	$\mu = 0.5$	$\mu = 0.6$
Dual R-MCL	.9997 / 34	.9988 / 51	.9946 / 62	.9876 / 94*	.7536 / 171*
L0-harvesting	.9800 / 896	.9884 / 942	.9786 / 1034	.9664/ 1336	.6635 / 1047
EN-harvesting	.9982 / 270	.9975 / 280	.9840 / 271	.8365 / 282	.1463 / 193

*$b = 2.0$ is used.

and three directional components $DC_1 = (S = \{A, B, C\}, T = \{C, D, E\}), DC_2 = (S = \{D, E\}, T = \{A, G\}), DC_3 = (S = \{F, G\}, T = \{B, H\})$ are constructed. The advantage of this procedure is that the distribution of in-degree and out-degree are kept so the network is still a power-low network. A drawback is that and the size of source part and terminal part of each directional component are the same, but this should not affect the measure of accuracy of execution time much. The experiments were performed on a machine with Intel core i5 650 and 16GB of main memory.

4.2 Comparison against Harvesting algorithms

We compare Dual R-MCL with two recent algorithms *L0-harvesting* and *EN-harvesting* [9], which are also designed for detecting directional component. We have tested other algorithms designed for detecting traditional communities, but none of them can generate meaningful clusters. For example, LinkCommunity [1] is a well-known algorithm to detect overlapping clusters in an undirected network. After ignoring the direction of edges and shuffling the destination indices of all edges, LinkCommunity is unable to detect any significant cluster even if μ is small, showing that the partition density [1] is 0. L0-harvesting and EN-harvesting are regularized Singular Value Decomposition (SVD) based algorithms which sequentially identify dense directional components. The difference between L0-harvesting and EN-harvesting is the penalty type for SVD. The former adopts a L0 norm type of penalty and the latter adopts an Elastic-net type of penalty. In Dual R-MCL, the default parameter values of R-MCL ($b = 0.5$ and $r = 2$) are adopted unless otherwise noted.

The results are measured by the normalized mutual information (NMI) proposed in [12]. Given a set of gold standard communities $\mathbb{Y}$, the NMI of a set of generated clusters $\mathbb{X}$ is

$$NMI(\mathbb{X}|\mathbb{Y}) = 1 - (H(\mathbb{X}|\mathbb{Y}) + H(\mathbb{Y}|\mathbb{X}))/2,$$

where $H(\mathbb{X}|\mathbb{Y})$ is the conditional entropy of $\mathbb{X}$ given $\mathbb{Y}$. NMI is between 0 and 1 and 1 means perfect matching between the generated clusters and gold standard. This NMI is designed for overlapping clusters, as a directional component's source/terminal part might overlap with another directional component's terminal/source part. One assumption of this NMI is that $\mathbb{X}$ and $\mathbb{Y}$ covers all nodes. Although the results of Dual R-MCL, L0-harvesting and EN-harvesting might not contain all nodes, it always cover more than 95% nodes so this NMI can still be adopted. For each LFR parameter setting, ten networks are generated and the average NMI is calculated.

The results of NMI are shown in Table 1 and Table 2. Clearly, Dual R-MCL can obtain higher NMI than the other two algorithms, except the case of $\mu = 0.6$ and $deg_{avg} = 10$, in which all algorithms cannot achieve meaningful average NMI. All algorithms always obtain higher NMI when $deg_{avg} = 20$ than when $deg_{avg} = 10$. This is simply because when the average degree is larger, the information is more abundant, so it is easier for algorithms to correctly detect each directional component. When the noise level is higher, the boundary of each directional component is very hard to detect, so all algorithms tend to merge some directional components and to generate smaller number of directional components. In the extreme case of $\mu = 0.6$ and $deg_{avg} = 10$, all of these algorithms tend to merge all gold standard directional components together and generate only one giant directional component. Dual R-MCL can reduce the case by setting a larger value of balance factor b. A larger b can inhibit the flows going to a node which already attracts a large amount of nodes, so the generated directional components tend not to be all merged together. Therefore, when $\mu \geq 0.5$, $b = 2.0$ is adopted. Moreover, when $\deg_{avg} = 20$ and $\mu \leq 0.4$, EN-harvesting's NMI is higher than L0-harvesting, but when $\deg_{avg} = 10$ or $\mu \geq 0.5$, EN-harvesting's NMI is smaller. This partially follows the observation in [9] that L0-harvesting is can provide better accuracy when the network is sparse and EN-harvesting can generate more accurate directional components when the network is dense. However, when the noise level is very larger (≥ 0.5), L0-harvesting always achieves higher NMI, showing that L0-harvesting is more noise-tolerant.

As can be seen in Table 1 and Table 2, Dual R-MCL is more efficient than the other two algorithms. All algorithm consumes more time when $deg_{avg} = 20$ than when $deg_{avg} = 10$, because more non-zero values would causes matrix multiplication and SVD executing longer. We find that Dual R-MCL consumes longer time when μ is larger, because it needs more iterations for the flow matrix to converge. For L0-harvesting and EN-harvesting, the execution depends on the number of generated directional component and the noise level. When the noise level is increased, the execution time of SVD increased; however, since fewer directional components are generated as mentioned above, the process of sequentially identifying directional components is faster (but the result tends to be incorrect). As generating accurate enough results (NMI$>$ 0.6), Dual R-MCL is much more efficient, achieving 5x~25x and 3x~10x speed than L0-harvesting and EN-harvesting respectively.

5. CONCLUSION

In this study, we adopt a novel definition of communities, di-

rectional component, in a directed network. While the directional component cannot be detected by traditional community detection algorithms, we propose Dual R-MCL, which can separately discover source parts and terminal parts and efficiently match them into directional components. In the preliminary results, we show that Dual R-MCL outperforms other two directional component detection algorithms in terms of accuracy and execution time.

In future works, we intend to enhance the scalability of Dual R-MCL. Since the canonical flow matrix $flow(A*A^T)$ or $flow(A^T*A)$ is mush denser than the traditional canonical flow matrix in R-MCL, Dual R-MCL cannot process a network with more than 1 million nodes and 10 million edges very efficiently. A possible solution is the network sparsification technique analogous to [19] or the multi-level coarsening technique similar to [17].

6. REFERENCES

[1] Y. Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

[2] S. Asur, D. Ucar, and S. Parthasarathy. An ensemble framework for clustering protein–protein interaction networks. *Bioinformatics*, 23(13):i29–i40, 2007.

[3] S. Brohee and J. V. Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC bioinformatics*, 7(1):488, 2006.

[4] F. Chen, A. J. Mackey, C. J. Stoeckert, and D. S. Roos. Orthomcl-db: querying a comprehensive multi-species collection of ortholog groups. *Nucleic acids research*, 34(suppl 1):D363–D368, 2006.

[5] M. Coscia, F. Giannotti, and D. Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining*, 4(5):512–546, 2011.

[6] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.

[7] S. V. Dongen. Graph clustering by flow simulation. *PhD thesis, University of Utrecht*, 2000.

[8] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[9] S. Kim and T. Shi. Scalable spectral algorithms for community detection in directed networks. *arXiv preprint arXiv:1211.6807*, 2012.

[10] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[11] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, 2009.

[12] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

[13] E. A. Leicht and M. E. Newman. Community structure in directed networks. *Physical review letters*, 100(11):118703, 2008.

[14] M. E. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[15] K. Rohe and B. Yu. Co-clustering for directed graphs; the stochastic co-blockmodel and a spectral algorithm. *arXiv preprint arXiv:1204.2296*, 2012.

[16] M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, 2009.

[17] V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: applications to community discovery. In *Proceedings of SIGKDD*, pages 737–746. ACM, 2009.

[18] V. Satuluri and S. Parthasarathy. Symmetrizations for clustering directed graphs. In *Proceedings of the 14th International Conference on Extending Database Technology*, pages 343–354. ACM, 2011.

[19] V. Satuluri, S. Parthasarathy, and Y. Ruan. Local graph sparsification for scalable clustering. In *Proceedings of the 2011 international conference on Management of data*, pages 721–732. ACM, 2011.

[20] V. Satuluri, S. Parthasarathy, and D. Ucar. Markov clustering of protein interaction networks with improved balance and scalability. In *Proceedings of ACM-BCB*, pages 247–256, 2010.