A Novel Algorithm for Community Detection and Influence Ranking in Social Networks

Wenjun Wang and W. Nick Street
Department of Management Sciences
University of Iowa, Iowa City, IA 52242, USA
{wenjun-wang, nick-street}@uiowa.edu

***Abstract*—Community detection and influence analysis are significant notions in social networks. We exploit the implicit knowledge of influence-based connectivity and proximity encoded in the network topology, and propose a novel algorithm for both community detection and influence ranking. Using a new influence cascade model, the algorithm generates an influence vector for each node, which captures in detail how the node's influence is distributed through the network. Similarity in this influence space defines a new, meaningful and refined connectivity measure for the closeness of any pair of nodes. Our approach not only differentiates the influence ranking but also effectively finds communities in both undirected and directed networks, and incorporates these two important tasks into one integrated framework. We demonstrate its superior performance with extensive tests on a set of real-world networks and synthetic benchmarks.**

I. Introduction

Finding community structure is an important but difficult problem in network analysis, and has attracted a great deal of effort from many disciplines; see [1] and [2] for surveys. While many algorithms have been presented, there are still significant areas for improvement. One general problem is the lack of a quantitatively precise definition of community. While most researchers describe a community as a group of nodes with higher internal than external connectivity, this notion of connectivity is ambiguous, leading to many different objective functions and performance metrics. Leskovec et al. [3] present an empirical comparison of a range of community-detection algorithms. They point out that intuitive notions of cluster quality tend to fail as one aggressively optimizes the community score, and conclude that approximate optimization of community score introduces a systematic bias into the extracted clusters. Yang et al. [4] evaluate various commonly-used objective functions and cast doubt on their quality. Modularity-based methods have crucial limits as well in spite of their popularity [5].

Our work is motivated by observations from real-world communities. Community members have individual social roles: leaders, elite members, liaisons to other communities, etc. Some are more influential than others, and some are more susceptible to influence. Individual roles, influence, and susceptibility are encoded in the network structure. In fact, it is influence that not only differentiates individual roles but also acts as the force holding the individuals together to form the community. Shared-Nearest-Neighbor (SNN) similarity [6], used in traditional clustering, indicates that two nodes both being close to a common set of neighbors suggests they are close to each other. This is naturally extended to our influence-based scenario, and can be rephrased as: *that two nodes both influencing a common set of (direct and indirect) neighbors suggests they are close to each other*. We term this idea **Shared-Influence-Neighbor (SIN)** similarity, which captures our intuitive notion of community: that two nodes both influencing a common set of neighbors confirms their closeness and same-community membership.

Having chosen influence as the context for community detection, the next question is how to define a quantitatively precise influence measure. One fundamental step in influence analysis is to differentiate the relative influence of the nodes, which is often characterized using various centrality measures. There are five widely-used measures of centrality: *degree*, *closeness*, *betweenness*, *eigenvector*, and *Katz* centralities. Unfortunately, none of them is fine enough or comprehensive enough to quantify overall influence. *Degree* centrality is simple but very coarse. It has many ties and fails to take into account the influence weight of even the immediate neighbors. The *closeness* and *betweenness* centralities are both based on the shortest path. However, the spread of information does not always go along the shortest path in reality. *Eigenvector* centrality captures an intuitive but important concept: *connecting to a more influential node contributes more influence weight to the node of interest than connecting to a less influential node*. Unfortunately, it fails to capture the fact that influence is attenuated when passing through the network. The well-known PageRank is a variant of eigenvector centrality. For undirected graphs, PageRank degenerates into degree centrality. *Katz* centrality is a good generalization of degree centrality and eigenvector centrality plus an attenuation factor associated with the path length. However, it allows influence to be transmitted in a loop infinitely, which is not realistic. Further, none of these centralities is able to measure the influence-based *proximity* between pairs of nodes, which as indicated could be a desirable metric for community detection.

We use concepts and techniques from the fields of network modeling, influence diffusion, AI, and data mining to decode the implicit but rich information of influence-based connectivity in network topology, and arrive at a novel approach to both community detection and influence ranking in both undirected and directed networks. Experiments on real and simulated networks show the superior performance of our algorithm.

ASONAM 2014, August 17-20, 2014, Beijing, China
978-1-4799-5877-1/14/$31.00 ©2014 IEEE

II. METHODOLOGY

Our approach differs from prior work in many ways. We draw inspiration from the PageRank algorithm in the sense that we cannot rely solely on the node degree. We propose a new influence diffusion model that embeds influence into a node and passes it around in the network. From the point of view of centralities, our model extends *degree* centrality from immediate neighbors to multi-step neighborhoods, includes the shortest path (that the *closeness* and *betweenness* centralities rely on) and non-shortest paths, and takes into account both neighbors' influence weight (like *eigenvector* centrality) and influence attenuation (like *Katz* centrality) but without cycling. Further, we not only find the total influence a node spreads, but also keep track of where and how much its influence is distributed so as to build its influence vector for community detection.

A. Influence Diffusion Model

The influence defined in our diffusion model is different from the influence defined in many other diffusion models, such as the popular *independent cascade* model and the *linear threshold* model in [7], in which the influence of a node is quantified by the number of inactive nodes it can activate. We assume the influence decays with path distance, and measure a node's influence weight by the total amount of influence it spreads through the network. In addition, the influence in our model is realized and transmitted through *out-links* step by step. Therefore, for a specific real-life network, we need to understand what the link direction represents. For example, in a citation network, if paper *i* cites paper *j*, then the network contains a directed link from node *i* to node *j*. However, this directed link does not reflect the direction of the influence propagation since it is actually the cited paper *j* that influences the citing paper *i*. So, we need to reverse the citation network to fit into our influence diffusion model. Finally, our model can be applied to both directed and undirected (or mixed) networks. For any undirected link, we allow influence to be transmitted through it in either direction. In other words, if the link between nodes *i* and *j* is undirected, we replace it with a directed link from *i* to *j* and another directed link from *j* to *i*.

Our influence diffusion model can be regarded as a simple branching process, in which influence originates from a root node and propagates step by step to its offsprings following out-links. We have three important rules implemented in this model. First, **cycling is prohibited**. It makes sense since no one should repeatedly exert influence in cycles in the same round of an influence diffusion process. This distinguishes our model from *Katz* centrality and most random-walk-based community-detection algorithms. Second, **revisits along different routes are allowed and independent**. This is a realistic imitation in the sense that the influence originating from the root node may be delivered to the same person via many different routes independently. This distinguishes our centrality from the closeness/betweenness centralities which only focus on the shortest path. Further, to capture the *influence locality*, it is reasonable to assume that **the farther away from the root, the less influence the message exerts on arrival**. We draw inspiration from the small-world phenomenon and the concentric scales of resolution around a particular node depicted in [8]. It is claimed that the probability of a center node linking to a node at a fixed distance d of the ring is proportional to d^{-2}. This idea fits our influence scenario, and we define a **depth-associated coefficient** to quantitatively model the attenuation of influence, which is the inverse square of the current depth from the root node.

Input : Directed graph $G(V, E)$ with $n = |V|$
Maximum depth $depthLimit$
Output : Influence-based connectivity matrix M

1: Set the influenceWeight of each node to 1
2: **for** node i = 1 to n **do**
3: Empty open/close list
4: Set all nodes to be unexplored
5: OpenList-PushStack ($Node(i)$)
6: Set $Node(i).depth = 0$
7: **while** OpenList is not empty **do**
8: $currNode$ = OpenList-PopStack ( )
9: $Node(i).influenceVector(currNode)$ += $(currNode.depth)^{-2}$
10: Pop all nodes in CloseList with $depth \geq currNode.depth$
11: Set those nodes to be unexplored
12: **if** $currNode.depth < depthLimit$ **then**
13: **for** each out-link neighbor j of $currNode$ **do**
14: **if** $Node(j)$ is unexplored **then**
15: OpenList-PushStack($Node(j)$)
16: Set $Node(j).depth = currNode.depth + 1$
17: **end if**
18: **end for**
19: Set $currNode.isExplored = True$
20: CloseList-PushStack ($currNode$)
21: **end if**
22: **end while**
23: $M(i)$ = $Node(i).influenceVector$
24: **end for**
25: Return M

Fig. 1. Pseudocode of InfluenceMatrix-Builder.

B. Influence Matrix

We employ a modified depth-limited search algorithm to generate an influence vector for each node. The pseudocode in Figure 1 shows how we sweep over all the nodes to build the influence matrix consisting of the influence vectors of all the nodes in the network.

As discussed above, both undirected graphs and mixed graphs can be converted into directed graphs by simply replacing undirected links with a pair of directed links. Without loss of generality, our algorithm takes a directed graph and a pre-specified maximum depth limit as input. It maintains an open list of to-be-explored nodes and a close list of already-explored nodes, both implemented as a stack (Last-In-First-Out). Each

node in the open/close list contains an *integer* variable that indicates its depth in the influence rings of the root node, and a *Boolean* variable that indicates whether it has been explored so as to avoid cycling. The algorithm assigns an initial influence weight of 1 to each node (Line 1). Starting with a root node in the open list and an empty close list (Lines 3-6), we do the depth-limited search until the open list is empty. Whenever a node (*currNode*) is popped from the open list (Line 8), we calculate the attenuated influence and accumulate it in the root node's influence vector accordingly (Line 9). Then we pop from the close list all the nodes whose depth are greater than or equal to the depth of the current node, and set all of those nodes to be unexplored (Lines 10-11). This allows revisits from different routes. Then we check whether the depth of the current node is less than the maximum depth limit. If it is, we explore the current node by pushing to the open list all of its *out-link neighbors* that have not been explored, set their depth, denote the current node as explored, and push it to the close list (Lines 12-20). Each iteration of the *For* loop generates an influence vector of a specific node, which contains all the nodes it affects associated with the corresponding influence value distributed from that node. After sweeping over all the nodes, the algorithm builds up an influence matrix as a whole of the network.

We also develop a closed form for the influence matrix up to a depth limit of 3. Let A denote the adjacency matrix of the network (without self-loops). The matrix A^n (i.e., the matrix product of n copies of A) has an interesting interpretation: the entry in row i and column j gives the number of paths of length n from node i to node j. Let D_n denote the diagonal matrix of A^n. The entry d_{ii} is the number of paths of length n for node i to walk to itself. Then the influence matrix M with different depth limit is given as follows:

$$M_0 = I$$

$$M_1 = M_0 + A$$

$$M_2 = M_1 + \frac{1}{4}(A^2 - D_2)$$

$$M_3 = M_2 + \frac{1}{9}[(A^2 - D_2)A - D_3 - AD_2 + A \otimes A^T]$$

M_0 is the initial assignment of influence weight of 1 to each node at depth 0, where I is the identity matrix. M_1 is simply the first-step influence propagation. In M_2, we avoid the two-step cycling by subtracting D_2 from A^2 and multiply by 2^{-2}, which is the two-step influence attenuation coefficient. In M_3, we first let all the nodes on the second depth propagate to depth 3, which is represented by $(A^2-D_2)A$, then subtract the three-step cycling D_3 of the root node and the two-step cycling of all the first-step nodes. For all those first-step nodes that link to the root node with an undirected link, we remove them twice, one in D_2A and one in AD_2. And so we get one back by adding $A \otimes A^T$, which is the component-wise multiplication of matrix A and its transpose matrix A^T. Finally, we multiply by 3^{-2} to reflect the three-step influence attenuation.

C. Influence Ranking

As described above, the influence weight of a node is quantified by the total influence it spreads throughout the network. Once the influence matrix is built, it is straightforward to compute the influence weight of each node, which is simply the summation of all the elements in the influence vector (a row vector in the influence matrix) corresponding to each individual node.

We refer to our influence ranking as **influence centrality**. It is noted that the pre-specified maximum depth limit is a nice gauge to measure the influence from local to global at different scales. It incorporates degree centrality (when *depthLimit=1*) and extends it to multi-step influence centrality. In addition, an important characteristic is hidden in the influence matrix. Let $Row(i)$ and $Column(i)$ denote the influence matrix's i^{th} row vector and i^{th} column vector, respectively. Then $Row(i)$ is the influence vector of node i that describes where and how much influence node i distributes through the network. Interestingly, the column vector $Column(i)$ is exactly a representation of where and how much influence node i acquires from the network. In other words, $Row(i)$ consists of the set of nodes that are influenced by node i, and $Column(i)$ represents the set of nodes that influence node i. The summation of all the elements in $Column(i)$ is the total influence node i receives from the network, which could be a good indicator of susceptibility among all the nodes in the network.

D. Influence-Guided Spherical K-means (*IGSK*)

From a geometric perspective, our algorithm projects the graph into an n-dimensional influence space, where each node defines one dimension. The position of a node in this space is determined by its influence vector. And we measure the closeness of two nodes with their **Shared-Influence-Neighbor (SIN)** similarity, i.e., the cosine similarity of their influence vectors. We can then apply a variety of well-studied clustering algorithms to find communities. In this paper, we use spherical K-means clustering [9]; hence our algorithm is termed as *Influence-Guided Spherical K-means* (**IGSK**).

We also use the influence information in a heuristic method for initializing the cluster centroids. Intuitively, the most influential member of a community is more likely to be located near the center of the community. We take advantage of the influence ranking, which is already available from the influence matrix. We first choose the node with the highest influence ranking as the centroid of cluster 1. For the next (*K*-1) centroids, we choose the remaining node with greatest influence, and assign it as a centroid of a cluster if its similarity score with the previously-selected centroids is less than a similarity threshold. This mechanism significantly improves both the accuracy and the efficiency.

The remaining parameter is the maximum depth. Remember that the influence diminishes inversely proportional to the square of the depth. We find that setting the maximum depth to 3 is sufficient for community detection. Moreover, for small-size networks or small communities, or when the community structure is fuzzy, setting the maximum depth to 2 may have

advantages over setting it to 3. In practice, we run **IGSK** by setting the maximum depth to 2 and 3. Between the two resultant clusterings, we finalize the cluster assignment with the one of higher modularity score.

III. Experiments

To get preliminary insights and verify the validity of our algorithm, we focus on networks with known communities. Since the ground truth for large-scale real-world networks is rarely available, we test our algorithm on several small real-world datasets and a set of simulated networks using the LFR model [10], and evaluate the performance by comparing with the ground truth and a set of representative algorithms. We also use a large citation network to evaluate our influence centrality.

A. Network Description

The real-world datasets are: Zachary's Karate Club [11], Mexican Political Power [12], Sawmill Communication Network [13], Dolphin Social Network [14], American College Football [15], and arxiv HEP-TH citation network [16]. All of them are commonly-used benchmarks for algorithm evaluation. Moreover, in order to compare with the 12 representative algorithms examined in [17], we generate a set of LFR benchmark graphs using exactly the same parameter settings: *average degree* = 20, *maximum degree* = 50, *degree-distribution exponent* = -2, *community-size-distribution exponent* = -1. There are two different network sizes (1000 and 5000 nodes), and two different ranges for community sizes (S and B). "S" stands for "small", which means *min/max community size* = 10/50. In contrast, "B" stands for "big", which means *min/max community size* = 20/100. In each of the 8 benchmark sets (4 undirected sets and 4 directed sets), we vary the *mixing parameter* from 0.1 to 0.8, and generate 5 realizations for each value of the mixing parameter. This results in a total of 320 simulated networks.

B. Influence Ranking

Figure 2 shows the Karate Club network. Its influence ranking (with *depthLimit=3*) is shown in Figure 3. We use a rating scale of 0 to 10, with 10 meaning "most influential". Our algorithm uncovers the minute influence difference among nodes based on the network topology. It successfully identifies the two leaders (nodes 1 and 34) and a set of core members (nodes 2, 3, 4, 9, 14, 32 and 33). Interestingly, nodes 10 and 17 both have a degree of 2, but node 10 has a much higher influence weight than node 17 since node 10 connects to nodes 34 and 3, which are much more influential than nodes 6 and 7 to which node 17 connects. Moreover, even though node 12 only has a degree of 1, its influence weight is also greater than that of node 17 because node 12 has a direct connection to the leader Node 1. It follows our intuition that connecting to a more influential person contributes more influence to the person of interest than connecting to a less influential one.

The arXiv HEP-TH citation network consists of 27,771 papers and 352,807 citations among them. Those papers are in the field of high energy physics, and were added to the

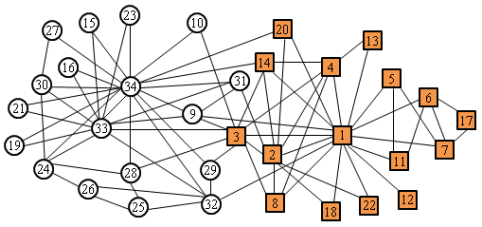

Fig. 2. Zachary's karate club.

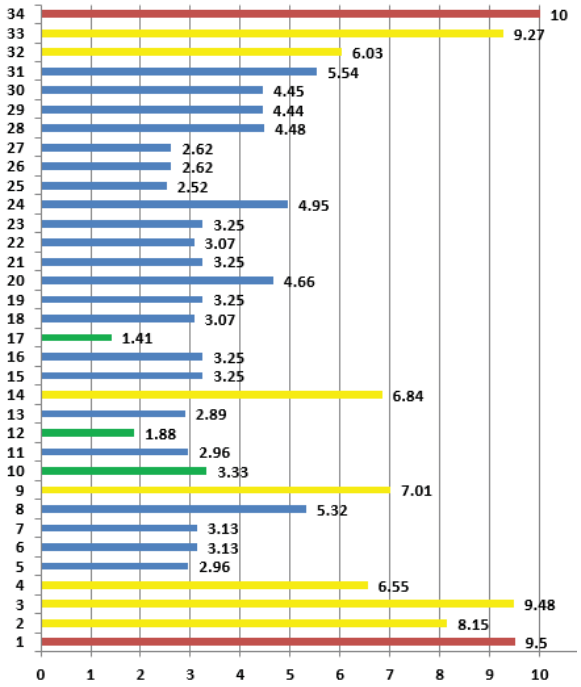

Fig. 3. Influence ranking of Zachary's karate club.

e-print arXiv between 1992-2003. The first 4 digits of each paper ID represent the year and the month when the paper was published online. For instance, paper *9510017* indicates it was published in October of 1995. We list in the *Influence Centrality* column of Table I the top 10 papers identified by our influence-ranking algorithm (*depthLimit=3*), and compare it with the in-degree centrality and PageRank. The in-degree centrality is the number of citations a paper receives, which is listed in parenthesis in the *In-degree centrality* column. The number listed in parenthesis in the *Influence centrality* column is the in-degree ranking of the corresponding paper.

Like most centrality measures, our influence centrality is correlated with degree centrality. All the top-10 influence-centrality papers are of very high in-degrees, which we can tell by their respective in-degree ranking. It includes 7 of the top-10 in-degree centrality papers but ranks them in different order. While we cannot prove that our influence centrality gives the most accurate ranking, it does differentiate the influence ranking in a more meaningful and precise manner than in-degree centrality. As discussed, degree centrality can be regarded as a special case of our influence centrality,

TABLE I
COMPARISON OF DIFFERENT CENTRALITIES.

Rank	Influence centrality	In-degree centrality	PageRank
1	9510017 (6)	9711200 (2414)	9402044
2	9503124 (8)	9802150 (1775)	9205068
3	9711200 (1)	9802109 (1641)	9205027
4	9410167 (15)	9407087 (1299)	9207053
5	9510135 (14)	9610043 (1199)	208020
6	9802150 (2)	9510017 (1155)	9204102
7	9802109 (3)	9908142 (1144)	9301042
8	9610043 (5)	9503124 (1114)	9201019
9	9407087 (4)	9906064 (1032)	9205081
10	9601029 (17)	9408099 (1006)	9209016

TABLE II
COMPARISON ON REAL-LIFE NETWORKS USING RANDINDEX AND NMI.

	RandIndex				NMI		
Algorithm	Karate	Mexican	Sawmill	Algorithm	Football	Dolphin	Karate
RankClus	**1.000**	0.489	0.530	GPSODM	**1.000**	0.723	**1.000**
Walktrap	0.745	0.536	0.560	GGADM	0.910	0.736	**1.000**
KernelBased	0.941	0.536	0.527	HA	0.907	0.707	0.754
LinkComm	0.743	0.536	0.560	MMC	0.885	0.579	**1.000**
SPICi	0.586	0.553	0.629	LPA	0.927	0.710	0.751
Betweenness	0.913	0.605	0.570	InfoMap	0.899	0.695	0.643
IGSK	**1.000**	**0.716**	**0.870**	IGSK	0.924	**0.814**	**1.000**

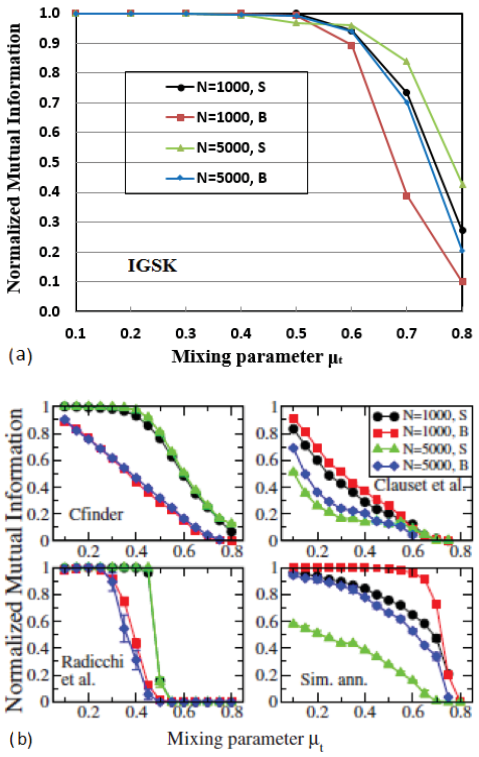

Fig. 4. Performance comparison on undirected LFR benchmark graphs. (a) our **IGSK** and (b) is a plot given in [17].

i.e., *depthLimit=1*, in which it simply counts the number of immediate in-link neighbors. When we set *depthLimit=3* as we do by default, we explore the 3-step neighborhood of each node, in which neighbors' influence weights are incorporated via the influence diffusion process. In addition, it is observed that PageRank fails to rank the influence in this case. All the top-10 PageRank papers are old papers that do not have any out-links (since the papers they cited are not included in the dataset).

C. Community Detection

Table II shows the results of applying our algorithm **IGSK** to 5 real-life datasets. We use *RandIndex* for comparison with the 6 representative algorithms presented in [4], and use *Normalized Mutual Information* (NMI) for comparison with another set of 6 algorithms presented in [18]. It is shown our algorithm has the best performance overall.

We illustrate our results of the tests on the 4 sets of undirected LFR benchmarks (1000-node-S/B and 5000-node-S/B) in Figure 4 (a), in which each curve shows the variation of the averaged NMI score with respect to the mixing parameter. Our algorithm shows excellent performance. Even when the mixing parameter is set to 0.5 (the threshold of defining strong communities), we achieve NMI scores of 0.999, 0.992, 0.968, and 0.99 for 1000-node-S, 1000-node-B, 5000-node-S, and 5000-node-B, respectively. Our algorithm is generally not sensitive to the community size or the network size. We do the performance comparison against a set of 12 representative community-detection algorithms examined in [17]. Due to space limits, we illustrate in Figure 4 (b) the performance of 4 algorithms in [17]. Our algorithm is one of the top-3 performing algorithms among the 12 algorithms.

Finding communities in directed networks is more challenging. Most existing algorithms are not able to deal with directed networks. Among the 12 algorithms Lancichinetti and Fortunato examined in [17], only 5 can be used for directed networks. We generate 4 sets of directed networks using the same parameters as theirs, in which both *degree-distribution exponent* and *mixing parameter* refer to the in-degree of the nodes while the out-degree is kept constant for all nodes. This setting makes the resulting networks similar to the citation networks in terms of in-degree/out-degree distributions. Therefore, as we did with the arXiv HEP citation network, we reverse these LFR directed graphs to reflect the influence flow in our influence diffusion model, and then run our **IGSK** algorithm to find the communities. We illustrate our results in Figure 5 (a), and compare the performance of our algorithm against the 2 algorithms Lancichinetti and Fortunato investigate in [17] as seen in Figure 5 (b). Our algorithm demonstrates superior performance in directed networks as well (even better than its performance in undirected networks).

D. Space and Time Complexity Analysis

Let n denote the total number of nodes in the network, b denote the average node out-degree, d denote the depth limit, K denote the number of communities, I denote the number of iterations to converge, and L denote the average length of the influence vectors.

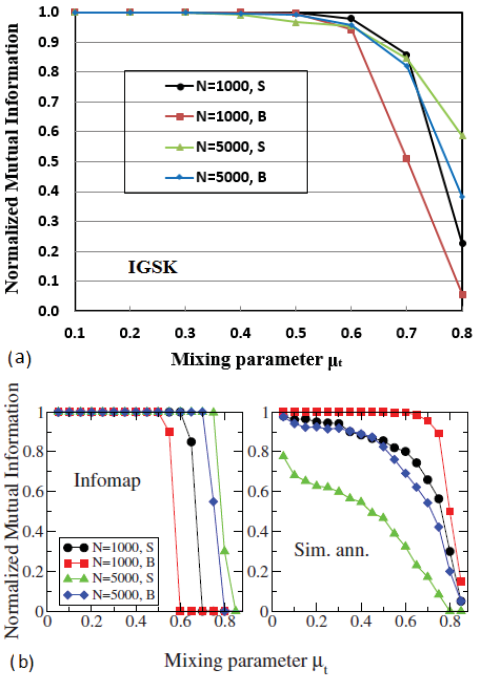

Fig. 5. Performance comparison on directed LFR benchmark graphs. (a) our **IGSK** and (b) is a plot given in [17].

Space complexity is closely related to the influence vectors. Using an array of length n definitely wastes space since most nodes can never spread influence to all other nodes in the network. To improve the space complexity, we store the influence vector in compact arrays that keep only the nodes affected by the root node. Let L denote the average length of the influence vectors, we then have the space complexity of $O(nL)$. L is directly affected by the depth limit d and the average node degree b as well as the community structure.

For time complexity, it may take $O(b^d)$ time for a node to go through the influence branching process to build its influence vector. The worst case is $O(nb^d)$ time to sweep over all the nodes to generate the influence matrix. But this worst case is unlikely to happen since its corresponding network structure is a full tree structure. In fact, in most cases, it is roughly $O(nL)$ time to generate the influence matrix. The next is the spherical K-means clustering. It has a time complexity of $O(nKLI)$. It is demonstrated in our experiments that **IGSK** converges very fast especially when the community structure is clear. For example, for almost all our experimented LFR benchmarks, the clustering converges in 2 iterations when the mixing parameter is 0.3 or less. When the community structure is fuzzy (*mixing parameter>0.6*), we force it to stop if it does not converge after 8 iterations.

IV. Conclusion and Future Work

In this paper, we provide a new perspective on the influence-based connectivity of network graph topology, and define a novel **influence centrality** and **Shared-Influence-Neighbor (SIN)** similarity in an integrated framework. Our *influence centrality* differentiates the influence ranking in networks in a more meaningful and detailed manner, and the **SIN** similarity is well-suited as a refined proximity metric for community detection. Experiments on both real-world and simulated networks show the effectiveness and superior performance of our algorithm (**IGSK**) in both undirected and directed networks.

In future work, we will examine the combination of this influence-based methodology with other clustering techniques to avoid pre-specifying the number of communities, and extend this approach to weighted networks as well as finding overlapping communities. Furthermore, we will combine the algorithm with content analysis, i.e., considering both the network graph topology and the nodes' profile information. Finally, we point out that the **influence ranking** and the **SIN** similarity metric introduced in this paper provide important implications for viral marketing and link prediction in social networks.

References

[1] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2010.
[2] F. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics Reports*, vol. 533, pp. 95–142, 2013.
[3] J. Leskovec, K. J. Lang, and M. W. Mahoney, "Empirical comparison of algorithms for network community detection," in *WWW*, 2010.
[4] Y. Yang, Y. Sun, S. Pandit, N. Chawla, and J. Han, "Is objective function the silver bullet? A case study of community detection algorithms on social networks," in *ASONAM*, 2011, pp. 394–397.
[5] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, no. 1, pp. 36–41, 2007.
[6] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared nearest neighbors," *IEEE Transactions on Computers*, no. 11, pp. 1025–1034, 1973.
[7] D. Kempe, J. M. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *9th ACM SIGKDD*, 2003, pp. 137–146.
[8] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambride University Press, 2010.
[9] I. Dhillon and D. Modha, "Concept decompositions for large sparse text data using clustering," *Machine Learning*, vol. 42, no. 1, pp. 143–175, 2001.
[10] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithm," *Physical Review E*, p. 78:046110, 2008.
[11] W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, pp. 452–473, 1977.
[12] J. Gil-Mendieta and S. Schmidt, "The political network in mexico," *Social Networks*, vol. 18, no. 4, pp. 355–381, 1996.
[13] J. Michael and J. Massey, "Modeling the communication network in sawmill," *Forest Products Journal*, vol. 47, pp. 25–30, 1997.
[14] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behavioral Ecology and Sociobiology*, vol. 54, pp. 396–405, 2003.
[15] M. Girvan and M. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.
[16] J. Gehrke, P. Ginsparg, and J. M. Kleinberg, "Overview of the 2003 kdd cup," *SIGKDD Explorations*, vol. 5, pp. 149–151, 2003.
[17] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *Physical Review E*, vol. 80, pp. 056 117(1–11), 2009.
[18] A. Hajibagheri, A. Hamzeh, and G. Sukthankar, "Modeling information diffusion and community membership using stochastic optimization," in *ASONAM*, 2013, pp. 175–182.