Finding Hierarchical Communities in Complex Networks Using Influence-Guided Label Propagation

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

***Abstract*—Communities play fundamental organizational and functional roles in various complex network systems. Community detection is an important challenge in network analysis. We approach community detection based on a *Shared-Influence-Neighbor* (SIN) similarity metric that measures the closeness of a pair of nodes in terms of their mutual influence and the common set of nodes they both influence. In this paper, we present two novel *influence-guided label propagation* (IGLP) algorithms. One is called *IGLP-Weighted-Ensemble* (IGLP-WE), in which each node adopts the label of the majority of its neighbors, weighted by the SIN similarity. This simple weighting scheme effectively resolves the significant stability issue in conventional label propagation algorithms. The other is called *IGLP-Direct-Passing* (IGLP-DP), in which the label is propagated directly from one node to its *most similar* neighbor step by step. This new label propagation method produces a deterministic partition and requires no convergent iterations. For both IGLP-WE and IGLP-DP, we regard the resultant partitioning as the initial configuration of the community structure. We then perform agglomerative hierarchical clustering to uncover the hierarchical communities at different scales using a new cluster-proximity measure. Extensive tests on a set of real-life networks and synthetic benchmarks demonstrate superior performance of our algorithms in terms of both quality and efficiency in undirected/directed and unweighted/weighted networks. Both IGLP-WE and IGLP-DP manifest promising scalability for large-scale networks.**

***Keywords*-community detection, influence diffusion, label propagation, hierarchical clustering**

I. Introduction

Community is one of the most significant structural properties in networks. From a karate club of tens of members to large-scale online social networks with millions of participants, from functional protein modules in biology to arXiv citation networks, community structures occur in various network systems. Community detection is of great importance in a wide variety of applications. However, finding meaningful communities is a difficult task. Despite the active attention and efforts from many disciplines, there are still many open issues.

Due to ambiguity in the definition of community, a surprisingly large number of algorithms have been presented using many different objective functions and performance metrics. Unfortunately, it is not clear which algorithms are more consistently reliable in practice. Further, numerous networks in diverse domains (such as Twitter and the Web) are directed and/or associated with weights on edges to differentiate the strength of the connections. Nevertheless, most existing algorithms ignore the link direction and/or the weights, which may cause considerable loss of information and lead to unreliable or even misleading results. Moreover, complex network systems often exhibit hierarchical community structure in which small communities successively group together to form larger ones. However, the number of algorithms that can identify hierarchical communities in complex networks is limited, and those that do are usually not scalable to large-scale networks. We propose the following general but essential design principles for new community-detection algorithms, and suggest using them as evaluation criteria for a more comprehensive performance comparison across various algorithms.

- **Applicability:** Many algorithms have been proposed for undirected binary networks. However, directionality and weights are usually essential features of diverse networks. Ignoring the edge direction and weights fails to capture the asymmetric relationship or the strength of the relationship between vertex pairs. An *applicable* algorithm should be able to incorporate the directionality and weights such that it can be applied to not only undirected binary networks but also directed/weighted networks.
- **Quality:** An algorithm should be evaluated using a set of well-known network benchmarks with or without known communities, i.e., ground truth. For a network with known communities, the widely-used *normalized mutual information* (NMI) [1] is suitable for measuring the accuracy of a community partition against the ground truth. For network benchmarks without ground truth, the *modularity* [2] is a well-defined measure for the quality of a community partition. A *reliable* algorithm should demonstrate both high accuracy in terms of NMI score on benchmarks with ground truth, and also find community partitions with high modularity score on benchmarks with or without ground truth.

- **Hierarchy:** Real-life network systems often exhibit some form of hierarchical organization; large communities are generally found to be made up of smaller, tighter ones. We argue that even the "ground truth" of the network benchmark is actually not unique but is subject to the desired granularity level in the hierarchy of the community structure. Moreover, the ground-truth community partition does not have to be the one that maximizes the modularity score. Instead, it could be the one that just (closely) matches the hierarchical community structure at some specific scale, which partially explains why exhaustive modularity maximization algorithms often fail to arrive at the ground truth of real-life network benchmarks. Therefore, a *robust* algorithm should be able to find hierarchical communities and enable us to zoom into the community hierarchy at different scales.
- **Scalability:** In the era of big data, the network may grow unprecedentedly huge in size. However, most existing algorithms are too computationally expensive to be scalable for large-scale networks. A *feasible* algorithm should be not only effective but also efficient enough to be of promising scalability, especially if we require the discovery of hierarchical community structure.

Guided by these four criteria, we develop two novel *influence-guided label propagation* (IGLP) algorithms, *IGLP-Weighted-Ensemble* (IGLP-WE) and *IGLP-Direct-Passing* (IGLP-DP), and present both of them in this paper. Our work on IGLP-WE is motivated by the well-known *label propagation algorithm* (LPA) [3] and a recently proposed *Shared-Influence-Neighbor* (SIN) similarity [4], [5]. LPA is basically an ensemble-based label association method, in which each node adopts the label that the majority of its neighbors have in an iterative process until a global consensus is reached. Thanks to its nearly linear time complexity, this algorithm has received much attention. Unfortunately, it comes with a significant stability issue due to random tie breaking. The problem exists in its majority voting rule, which simply counts the number of neighbors that carry the same label. This is a coarse technique that implicitly assumes the votes of all neighbors have the same weight, resulting in many ties. An intuitive solution is to weigh the vote of each neighbor by its similarity (or closeness) to the node of interest. But how can we differentiate the similarity between a node and its neighbors based on the network graph topology? We address this issue using the SIN similarity, which measures the closeness of any pair of nodes in terms of their mutual influence and the common set of nodes they both influence.

Further, we develop IGLP-DP inspired by the following intuition: *given only the network graph topology, any node should belong to the same community as its most structurally-similar neighbor*. We apply this idea to a new label-propagation framework: once a node identifies its most similar neighbor using SIN similarity, it passes its community label to that neighbor, which in turn passes the label to its most similar neighbor and so on. In the end of the label propagation, the nodes of the same labels are grouped into respective sub-communities. Each sub-community is composed of a group of most similar neighbors chained together. All of these sub-communities constitute the most compact configuration of the community structure.

Finally, for both IGLP-WE and IGLP-DP, we regard the resultant community partitions directly derived from the label-propagation process as the building blocks, and perform agglomerative hierarchical clustering using a novel boundary-node-based cluster proximity measure to reveal the hierarchical community structure. This approach greatly improves the time complexity of hierarchical clustering and maintains the representational advantages. We conduct extensive tests on a set of real-life networks and synthetic benchmarks, and demonstrate their superior performance in terms of both quality and efficiency in both undirected/directed and unweighted/weighted networks.

II. Related Research

Finding meaningful communities in complex networks is an important challenge. Although numerous methods have been presented, there is still significant area for improvement. An in-depth survey is beyond the scope of this paper. Here we briefly review the papers that are most relevant to our work.

A. Label Propagation Algorithms

The *label propagation algorithm* (LPA) for community detection is originally proposed by Raghavan et al. [3]. Suppose that a node x has j neighbors and let $C_x(t)$ denote the label of node x at the t^{th} iteration. The main steps of LPA can be described as follows:

1) Initialize the label of each node with its node index at $t = 0$, i.e., $C_x(0) = x$;
2) Set $t = 1$;
3) Randomly choose an order in which all the nodes update their labels;
4) For each node x chosen in that order, update its label using the majority voting rule, i.e., $C_x(t) = f(C_{x_1}(t), ..., C_{x_i}(t), C_{x_{i+1}}(t-1), ..., C_{x_j}(t-1))$, where $x_1, ..., x_i$ are the neighbors of x that have already updated their labels in the t^{th} iteration and $x_{i+1}, ..., x_j$ are those that have not updated their labels yet. f returns the label that the maximum number of neighbors carry and ties are broken randomly;
5) If every node has the same label as the maximum number of its neighbors, group the nodes sharing the same label into one community and stop the algorithm. Otherwise, set $t = t + 1$ and go to (3).

The label-updating process in Step 4 is asynchronous. In synchronous updating, node x at the t^{th} iteration updates its label based on the labels of its neighbors at $(t-1)^{th}$ iteration, i.e., $C_x(t) = f(C_{x_1}(t-1), ..., C_{x_j}(t-1))$. Raghavan et al. propose the use of asynchronous updating to avoid oscillations of labels that may occur in synchronous updating.

This algorithm achieves a time complexity of $O(km)$, where k is the number of iterations and m is the total number of edges in the network. This algorithm has gained much attention due to its simplicity and nearly-linear time complexity. However, it has a significant stability issue. It produces different community partitions in different runs due to both the random tie breaking and the random ordering of nodes in its label-updating process, making the algorithm nondeterministic.

Leung et al. [6] contrast asynchronous with synchronous updating, showing that synchronous updating is more stable on average but converges much more slowly than asynchronous updating. They also introduce hop attenuation and node preference to improve the performance. Xing et al. [7] attempt to avoid the randomness by fixing the node sequence of label updating and changing the label selection mechanism. They improve the stability but the quality of the resultant community partition is not consistently satisfactory. Xie and Szymanski [8] introduce a set of operations to control and stabilize the label propagation, and produce deterministic partitions. However, all the methods discussed above require one or more user-specified parameters, which partially shifts the randomness issue from tie breaking to the selection of those parameters. In this regard, the stability issue of LPA is still open.

B. Shared-Influence-Neighbor (SIN) Similarity

The SIN similarity is a meaningful and refined connectivity-based measure for the closeness of any pair of nodes in the network. It is derived from a simple influence diffusion model proposed by Wang and Street [4], [5]. Three important rules are implemented in the influence-diffusion process: 1) *cycling is prohibited*; 2) *revisits along different routes are allowed and independent*; 3) *the farther away from the root, the less influence on arrival*. To quantitatively model the attenuation of influence when it is transmitted outward from the root node, they define d^{-2} as the depth-associated attenuation coefficient, where d is the depth along the path from the root node to the node of interest.

In this model, any undirected link is replaced with a pair of directed links pointing to each other of the pair of nodes of interest. To capture influence diffusion in weighted networks, they introduce a weight normalization scheme to differentiate the relative susceptibility of a node to the influence of all its in-link neighbors. The *normalized susceptibility weight* is defined as

$$\hat{w}_{ij} = \frac{w_{ij}}{\max_{k \in L} w_{kj}},$$

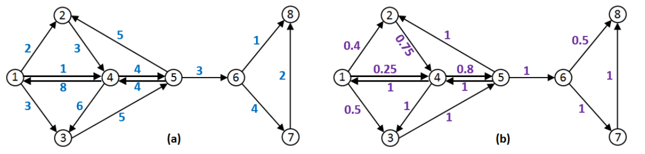

Figure 1. Example of a directed and weighted network (given in [5]): (a) Raw weights; (b) Normalized susceptibility weights.

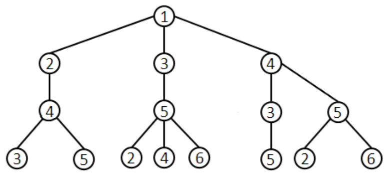

Figure 2. Egocentric influence rings of node 1.

where w_{ij} is the raw weight of a directed link pointing from node *i* to node *j*, and L denotes the set of in-link neighbors of *j*. In unweighted networks, all the raw weights are simply set to 1. As an example, a simple network is illustrated in Fig.1(a) with the corresponding normalized susceptibility weights shown in Fig.1(b). The influence-diffusion process generates the egocentric influence rings of the root node. Fig.2 shows the egocentric influence rings of node 1. Following a diffusion path from root node $i \to j \to k \to l$, the influence nodes *j*, *k*, and *l* acquire from node *i* is $\frac{1}{1^2} \times \hat{w}_{ij}$, $\frac{1}{2^2} \times \hat{w}_{ij} \times \hat{w}_{jk}$, and $\frac{1}{3^2} \times \hat{w}_{ij} \times \hat{w}_{jk} \times \hat{w}_{kl}$, respectively. Those coefficients are depth-associated attenuation.

A depth-limited search algorithm is developed to explore the egocentric influence rings and generate an influence vector for each node, recording where and how much the influence of the root node is distributed in its neighborhood. These influence vectors incorporate rich structural connectivity information and enable differentiation of the vertex-pair similarity in a more precise manner. For a pair of nodes *i* and *j* with their respective normalized influence vectors denoted as $\hat{V}_i$ and $\hat{V}_j$, the SIN similarity of *i* and *j* is defined as

$$S_{ij} = \hat{V}_i(j) \times \hat{V}_j(i) \; + \sum_{k \neq i,\, k \neq j} \hat{V}_i(k) \times \hat{V}_j(k).$$

It can be interpreted as the summation of their mutual influence (the first term) and the dot product of their influence vectors (the second term), which captures their similarity in terms of the set of neighbors they both influence.

Based on this influence diffusion model, Wang and Street [5] define an *influence centrality* and propose an *influence-guided spherical K-means* (IGSK) algorithm that effectively finds communities in undirected/directed and unweighted/weighted networks. It is demonstrated that the SIN similarity is a refined vertex-pair proximity metric. The main drawback of IGSK is that it requires the pre-specified number of communities. In addition, it does not scale well on large-scale networks.

C. Hierarchical Community Detection

Finding hierarchical communities in complex networks is desirable but even more challenging. Only a few algorithms in the literature attempt to unfold the community hierarchy. Blondel et al. [9] propose a heuristic algorithm (known as the *Louvain method*), in which a local-modularity-optimization phase and a community-aggregation phase alternate repeatedly until the maximum modularity is attained. It is well known for its high efficiency (roughly $O(n \log n)$) and high modularity. An implicit issue of this method is that the order of visiting each node may affect not only the the computation time but also the resultant community partitions. In addition, the resultant hierarchy of communities is not complete, as many intermediate levels are skipped. Instead of greedily maximizing the modularity, Huang et al. [10] improve the *Louvain method* by introducing a structural similarity used in density-based clustering and guiding the community-aggregation phase with a similarity-based modularity gain. Lancichinetti et al. [11] present an algorithm that is based on local optimization of a simple fitness function. The community structure is revealed by peaks in the fitness histogram, and different hierarchical levels can be investigated by tuning a resolution parameter.

III. Methodology

In this section, we elaborate in detail how we apply the SIN similarity to the label-propagation framework and arrive at *IGLP-Weighted-Ensemble* (IGLP-WE) and *IGLP-Direct-Passing* (IGLP-DP) algorithms.

A. IGLP-Weighted-Ensemble

As discussed above, the traditional LPA has a significant stability issue due to random tie breaking and random ordering of nodes in the label-updating process. The key problem exists in its majority voting rule, which simply counts the number of neighbors of the same labels. This implicitly assumes that all neighbors have the same weight applied to their votes, i.e.,

$$c_i = \underset{l}{\operatorname{argmax}} \sum_{j \in N_i^l} 1,$$

where c_i is the new label of node i and N_i^l is the set of neighbors of node i with label l. However, the similarity of a node to its different neighbors is not exactly the same. The SIN similarity is such a metric that enables us to differentiate the structural proximity of a node to its neighbors in detailed manner. Therefore, we propose a straightforward extension to the traditional LPA by weighing the vote of each neighbor using its SIN similarity to the node of interest. With S_{ij} denoting the SIN similarity of nodes i to j, our weighted majority voting rule can be written as

$$c_i = \underset{l}{\operatorname{argmax}} \sum_{j \in N_i^l} S_{ij}.$$

This simple weighting scheme not only directly resolves the random tie-breaking issue but also implicitly addresses the random node-order issue. Our experiments demonstrate that IGLP-WE always produces the same community partition regardless of the order of nodes in the label-updating process. Hence, we ignore the step of randomly choosing a node-updating order. Our IGLP-WE algorithm can be described in the following steps:

1) Generate the influence vector of each node based on the influence diffusion model;
2) For each node, calculate its SIN similarity to each of its neighbors;
3) Initialize the label of each node with its node index;
4) For each node, implement the asynchronous label updating using the *weighted* majority voting rule;
5) If no node changes its label, group the nodes into respective communities indicated by their labels and stop. Otherwise, go to (4).

B. IGLP-Direct-Passing

Given only the network graph topology, it is sensible to assume that: *any node belongs to the same community as its closest (most similar) neighbor*. Inspired by this intuition, we leverage the SIN similarity further to arrive at a new label-propagation framework, in which the community label is directly passed from a node to its most similar neighbor iteratively. At the end of this process, we are left with a set of most compact sub-communities. Each sub-community is composed of a group of most similar neighbors chained up from one to another.

The pseudocode of IGLP-DP is shown in Algorithm 1. Each node contains three integer variables: 1) *msn* denotes the node index of its most similar neighbor; 2) *label* denotes its community index; 3) *parent* denotes the node index of the node that passes the community label to the node of interest. The algorithm starts with the generation of the influence vector for each node based on the influence diffusion model. Then it finds the most similar neighbor of each node using the SIN similarity, and sets the community label of all nodes to 0 to indicate they are all unlabeled (Lines 3-6). In the following *For* loop, we first check if node i (*root node*) is unlabeled (Line 8). If yes, we assign its node index as its community label, and set its *parent* to be 0 to indicate its parent node is null (Line 9-10). We then assign its node index to *pIndex* and the node index of its most similar neighbor to *cIndex* (Lines 11-12). In the following *while* loop (Lines 13-18), we propagate *forward* the community label from *Node(pIndex)* to its most similar node *Node(cIndex)*, and denote *pIndex* as the node index of *Node(cIndex)*'s parent node until reaching a node that already has a community label, which is denoted as *newLabel*. If *newLabel* is the same as the root node's label i (i.e., the label we are propagating forward), the propagation stops, and we skip to the next node in the *For* loop. Otherwise, we propagate *backward* *newLabel* to *Node(pIndex)* and its parent node iteratively in a *while* loop until reaching the *root node* (Lines 20-25). After sweeping over all the nodes, IGLP-DP generates a set of sub-communities, in which each node acquires the same community label as its most similar neighbor.

Algorithm 1 IGLP-Direct-Passing

```
1:  procedure IGLP-DP($G(V, E)$)
2:      Generate the influence vector of each node
3:      for node $i \leftarrow 1, n$ do
4:          Find $Node(i).msn$ using the SIN similarity
5:          $Node(i).label \leftarrow 0$
6:      end for
7:      for node $i \leftarrow 1, n$ do
8:          if $Node(i).label = 0$ then
9:              $Node(i).label \leftarrow i$
10:             $Node(i).parent \leftarrow 0$
11:             $pIndex \leftarrow i$
12:             $cIndex \leftarrow Node(i).msn$
13:             while $Node(cIndex).label = 0$ do
14:                 $Node(cIndex).label \leftarrow i$
15:                 $Node(cIndex).parent \leftarrow pIndex$
16:                 $pIndex \leftarrow cIndex$
17:                 $cIndex \leftarrow Node(pIndex).msn$
18:             end while
19:             $newLabel \leftarrow Node(cIndex).label$
20:             if $newLabel \neq i$ then
21:                 while $pIndex > 0$ do
22:                     $Node(pIndex).label \leftarrow newLabel$
23:                     $pIndex \leftarrow Node(pIndex).parent$
24:                 end while
25:             end if
26:         end if
27:     end for
28: end procedure
```

IGLP-DP is easy to implement and efficient. It resolves the stability issue and produces a deterministic partition. Another distinguishing feature is that it requires no convergent iteration. It is worth pointing out that the backward propagation is essential to maintain the crucial property in which IGLP-DP is rooted, that is, *each* node should acquire the same community label as its most similar neighbor; meanwhile, it addresses the stability issue caused by node ordering in the label-updating process.

C. Hierarchical Clustering

The resultant community partitioning from IGLP-WE and IGLP-DP provides an initial configuration of the community structure. Especially for IGLP-DP, the initial configuration contains many small, tight sub-communities. We consider those communities in the initial configuration to be the building blocks for agglomerative hierarchical clustering to explore the community hierarchy at different scales.

The key issue here is how to define a cluster-proximity measure that is quantitatively accurate and computationally cheap. Since neighboring communities are connected by the boundary nodes, it makes sense to measure the cluster proximity using the SIN similarity of the boundary nodes. More precisely, we focus on the out-link-based boundary nodes of each community. We refer to a node as an *out-link-based boundary node* if it has at least one out-link neighbor that resides in a different community. We measure the cluster proximity between a pair of neighboring communities based on their out-link-based boundary nodes' SIN similarity. Considering that large communities tends to have more boundary nodes and more neighboring communities, we scale the similarity by the number of community members and the number of neighboring communities to eliminate the bias caused by the community size.

Let P_{ij} denote the cluster proximity between community *i* and community *j*, B_{ij} denote the set of boundary nodes in community *i* with out-link neighbors in community *j*, D_{kj} denote the set of node *k*'s out-link neighbors in community *j*, S_{kl} denote the SIN similarity of nodes *k* to *l*, N_i denote the number of nodes in community *i*, and C_i denote the number of neighboring communities that community *i* has. Similarly, let F_{ji} denote the set of boundary nodes in community *j* with out-link neighbors in community *i*, H_{mi} denote the set of node *m*'s out-link neighbors in community *i*, S_{mn} denote the SIN similarity of nodes *m* to *n*, N_j denote the number of nodes in community *j*, and C_j denote the number of neighboring communities that community *j* has. Then we define our cluster-proximity measure as

$$P_{ij} = \frac{1}{N_i \times C_i} \sum_{k \in B_{ij}} \sum_{l \in D_{kj}} S_{kl} + \frac{1}{N_j \times C_j} \sum_{m \in F_{ji}} \sum_{n \in H_{mi}} S_{mn}.$$

A major weakness of traditional hierarchical clustering is its high computational cost. A straightforward implementation has time complexity $O(n^3)$, which makes it too slow to be scalable for large-scale datasets. We take advantage of our boundary-node-based cluster-proximity measure, and arrive at a novel and highly efficient hierarchical-clustering algorithm. A high-level description of this algorithm reads as follows:

1) For each cluster, sweep over each cluster member's out-link neighbors to find the boundary nodes and construct a neighboring-cluster list associated with the corresponding cluster proximity;
2) Sweep over each cluster's neighboring-cluster list to find the closest pair of clusters;
3) Merge the two closest clusters and relabel them;
4) For each cluster, sweep over each boundary node's out-link neighbors to reconstruct its neighboring-cluster list associated with the updated cluster proximity;
5) Repeat Steps 2 to 4 until only one cluster remains.

Our hierarchical clustering takes the initial community partition given by the label propagation as input. This is a significant improvement in efficiency compared to conventional agglomerative clustering that starts by assigning each node to a separate cluster, since the number of clusters in our initial community partition is much smaller than the number of nodes in the network. The efficiency is further improved by constructing a neighboring-cluster list for each cluster which greatly reduces the time complexity when searching for the closest pair of cluster in Step [2]. Moreover, each node contains a *Boolean* variable that indicates whether it is a boundary node, which helps efficiently reconstruct the neighboring-cluster list and update the cluster proximity in Step [4]. This is another considerable improvement in time complexity.

Finally, we note that we incorporate our hierarchical-clustering algorithm as an integral part of the IGLP-WE and IGLP-DP algorithms. Hereafter whenever we refer to IGLP-WE or IGLP-DP, it means we perform the respective label propagation followed by the hierarchical clustering, and deliver a hierarchy of communities.

IV. Experiments

To evaluate the performance of IGLP-WE and IGLP-DP, we extensively test them on a large set of networks, which includes 6 widely-used real-life networks and more than 500 LFR benchmarks [12]. For networks with ground truth, we split the resultant hierarchy dendrogram at the level at which the number of separated communities is the same as the number of communities of ground truth, and use *normalized mutual information* (NMI) [1] to evaluate the accuracy of our community partition. For networks without ground truth, we adopt *modularity* [2] to evaluate the quality of a community partition. We present the maximum modularity we find in the community hierarchy of real-life networks.

A. Real-life Networks

The 6 real-life networks are: *Zachary's Karate Club* [13], *Dolphin Social Network* [14], *Political Books* [15], *American College Football* [16], *Email* [17], and *PGP Network* [18]. Only the first 4 have the ground truth of known communities. Their basic information is listed in Table I. Fig.3 is an illustration of Zachary's karate club network. The community dendrograms IGLP-WE and IGLP-DP generate are shown in Fig.4. As we can see, IGLP-WE produces an initial configuration of 6 sub-communities of different sizes, and IGLP-DP delivers 8 sub-communities by splitting 2 sub-communities detected by IGLP-WE into 4 smaller ones. In fact, IGLP-DP always produces more smaller and tighter sub-communities in the initial configuration than IGLP-WE. Both IGLP-WE and IGLP-DP find communities that match the ground truth perfectly when we split their respective hierarchy dendrogram at the level of two separate communities.

Table I
Real-life Networks

Networks	Nodes	Edges	Communities
Karate	34	78	2
Dolphins	62	159	2
PolBooks	105	441	3
Football	115	613	12
Email	1133	5451	—
PGP	10680	24316	—

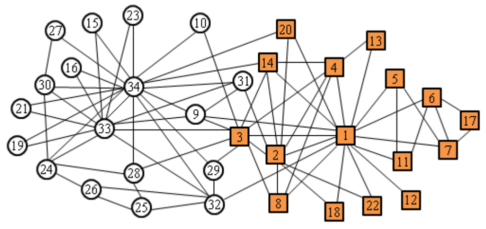

Figure 3. Zachary's karate club.

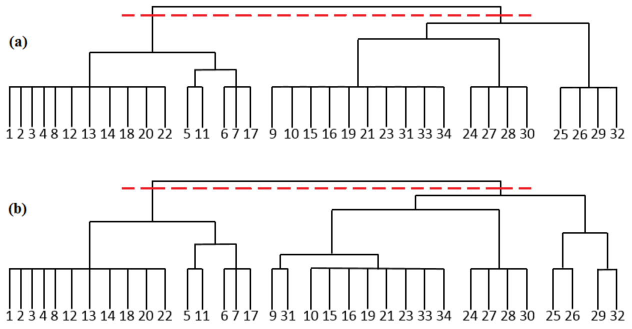

Figure 4. Dendrograms generated by: (a) IGLP-WE; (b) IGLP-DP.

The NMI and modularity scores of applying IGLP-WE and IGLP-DP to all the 6 real-life networks are listed in Table II. We compare the performance with a set of 9 representative algorithms examined in the literature [5], [7], [11], [19]. As we can see, both IGLP-WE and IGLP-DP demonstrate excellent performance. They clearly outperform *Fitness*, *CPM*, *Fastgreedy*, and *LPA* in terms of higher NMI scores on all networks with ground truth. Both of them achieve perfect NMI score on *Karate Club* and the highest NMI score on *Dolphins*. Moreover, when using the modularity as the metric, they both achieve superior performance. The only exception is the score of IGLP-WE on *Email*. It groups all the nodes in one community in its initial configuration. This is actually a common feature of most LPAs. One may notice that the traditional LPA is closely related to the *strong-community* definition, which requires *each* node to have higher internal connections than external connections. When the community structure is sufficiently fuzzy, LPA arrives at one or several *monster* communities. IGLP-DP exhibits its advantages in this regard by zooming into the community hierarchy in a much more detailed manner. IGLP-DP achieves higher modularity scores on both *Email* and *PGP* networks than *Fastgreedy*, which is a representative algorithm that explicitly maximizes modularity. In addition, it is worth pointing out that: for some algorithms (like *Fitness* and *NIBLPA*), one has to tune a user-specified parameter to find the best result for *each* network individually; the only user-specified parameter in IGLP-WE and IGLP-DP is the depth limit that specifies the maximum depth from the root node in the influence diffusion model. We set it to 3 (by default) for *all* networks examined in this paper. In fact, the depth limit of 3 can be regarded as a fixed, built-in parameter in both IGLP-WE and IGLP-DP, which matches the well-known 3-degree-of-influence phenomenon [20].

B. LFR Benchmarks

Since large real-life networks with reliable ground truth are rarely available (especially for directed and/or weighted networks), we further test our algorithms on LFR benchmarks. To compare with the algorithms examined in [12], we generate a set of LFR benchmark graphs using 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 size (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 unweighted benchmark sets (4 undirected and 4 directed), we vary the *topological mixing parameter* μ_t from 0.1 to 0.8. Similarly, we generate 4 sets of LFR weighted networks using the above parameters plus another 2 parameters: *weight-strength-distribution exponent* β = 1.5 and *weight mixing parameter* μ_w. As done in [12], while μ_w varies from 0.1 to 0.8, we fix μ_t to 0.5 and 0.8, respectively. We generate 5 realizations for each value of the mixing parameters and average the results in each experiment.

We illustrate in Fig.5 the results on the 4 sets of undirected and unweighted LFR benchmarks, in which each curve shows the variation of the averaged NMI score with respect to the topological mixing parameter μ_t. We compare the performance against 9 state-of-the-art algorithms including *IGSK* [5] and the 8 algorithms examined in [12]. They are referred to as: *Blondel et al.* [9], *MCL* [25], *Infomod* [26], *Infomap* [22], *Cfinder* [21], *Clauset et al.* [23], *Radicchi et al.* [27], and *Sim. ann.* [28]. Note that *Blondel et al.* is exactly the *Louvain method* discussed in Section II.

As we can see, IGLP-WE and IGLP-DP achieve almost perfect NMI scores on all datasets when μ_t is less than or equal to 0.5. IGLP-WE maintains its superior performance on 5000-S datasets with μ_t up to 0.7. IGLP-DP exhibits even better performance than IGLP-WE. IGLP-DP outperforms almost all the other algorithms except *Infomap*, which

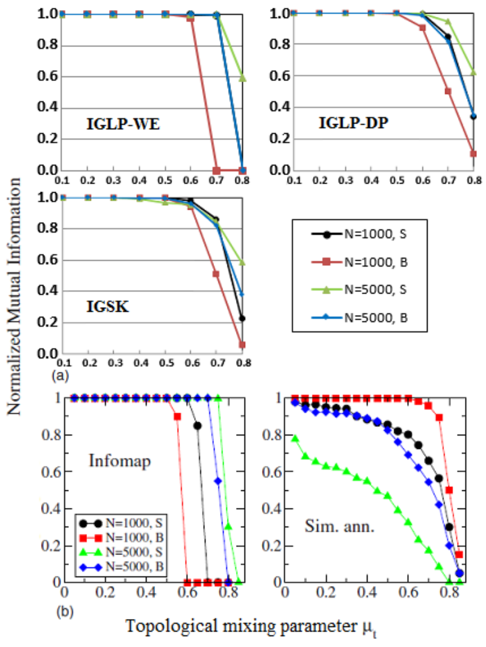

Figure 6. Performance comparison on directed and unweighted LFR benchmarks. Plot of *IGSK* is from [5], and plot (b) from [12].

achieves higher NMI scores than IGLP-DP when μ_t is 0.6 and 0.7. But as shown in Table II, IGLP-DP beats *Infomap* on all real-life networks except *Football* dataset. Both IGLP-WE and IGLP-DP perform better on larger networks and smaller community size.

Community detection in directed networks is more challenging. Most existing algorithms are not able to find communities in directed networks, and extension of an algorithm to directed networks is nontrivial. We illustrate the results of IGLP-WE and IGLP-DP in Fig.6 and compare them against *IGSK*, *Infomap*, and *Sim. ann.* Once again, both IGLP-WE and IGLP-DP demonstrate outstanding performance in directed networks (even better than their performance in undirected networks). IGLP-WE achieves perfect NMI scores on both 1000-S and 5000-S/B datasets when μ_t is less than or equal to 0.7. Both of them outperform *IGSK* on almost all datasets. They are better than *Infomap* on 1000-S/B datasets, and clearly beat *Sim. ann.* on both 1000-S and 5000-S/B datasets. It is noted that both IGLP-WE and IGLP-DP exhibit better performance on larger networks and smaller community size in directed networks as well.

For weighted networks, the perform comparison against *IGSK* and the 3 algorithms examined in [12] is shown in Fig.7. Interestingly, while both IGLP-WE and IGLP-DP consistently show excellent performance and maintain their preference on smaller community size, they show different performance on the topological mixing parameter. IGLP-WE works better on 5000-S/B-0.5 datasets, but IGLP-DP performs better on 5000-S/B-0.8 datasets. As discussed and shown in [5], when μ_t is 0.8 (fuzzy community structure),

Table II
PERFORMANCE COMPARISON ON REAL-LIFE NETWORKS

	NMI				Modularity					
Algorithm	**Karate**	**Dolphins**	**PolBooks**	**Football**	**Karate**	**Dolphins**	**PolBooks**	**Football**	**Email**	**PGP**
Fitness [11]	0.690	0.781	—	0.754	—	—	—	—	—	—
CPM [21]	0.170	0.254	—	0.697	—	—	—	—	—	—
DCBNT [19]	**1.000**	0.762	0.578	0.949	—	—	—	—	0.537	0.819
Infomap [22]	0.700	0.537	0.494	**0.972**	—	—	—	—	0.526	0.801
Fastgreedy [23]	0.693	0.557	0.531	0.753	—	—	—	—	0.507	0.853
Walktrap [24]	0.504	0.582	0.543	0.937	—	—	—	—	0.531	0.789
LPA [3]	0.583	0.516	0.572	0.863	0.296	0.465	0.489	0.582	0.38	0.806
NIBLPA [7]	**1.000**	0.622	**0.656**	0.872	0.423	0.521	0.497	0.582	0.427	0.783
IGSK [5]	**1.000**	0.814	0.541	0.924	0.421	0.406	0.513	0.609	—	—
IGLP-WE	**1.000**	**0.889**	0.482	0.927	**0.450**	**0.546**	0.463	**0.613**	0.002	**0.868**
IGLP-DP	**1.000**	**0.889**	0.576	0.918	**0.450**	0.528	**0.521**	0.612	**0.547**	0.863

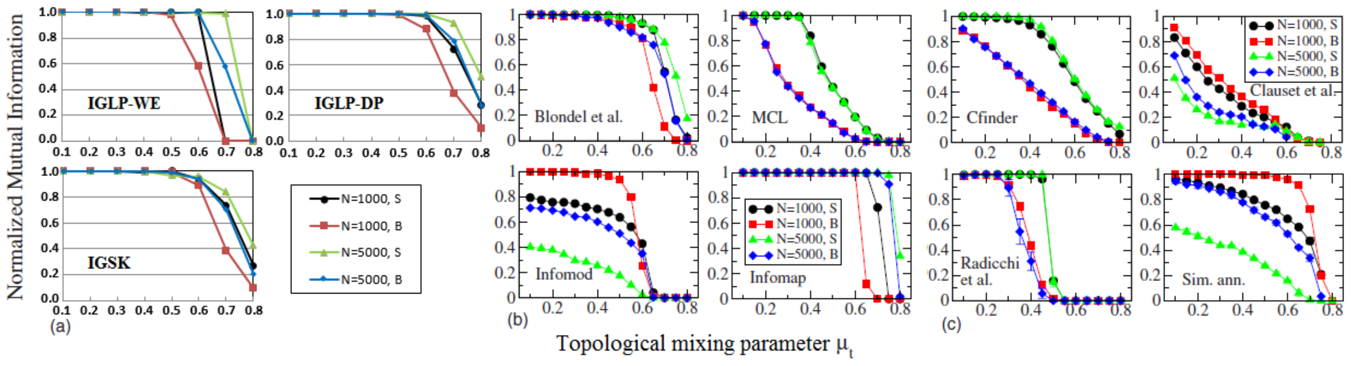

Figure 5. Performance comparison on undirected and unweighted LFR benchmarks. Plot of *IGSK* is from [5], and plots (b) and (c) from [12].

the weight distribution always reinforces the community structure while μ_w is less than 0.8. However, if μ_t is set to 0.5 (relatively clear community structure), the weight distribution undermines the community structure when $\mu_w > 0.5$. Therefore, we can infer that IGLP-DP is able to exploit the weight information more than IGLP-WE, and so, is more sensitive to the weight distribution. Except for IGLP-WE on 5000-B-0.8 dataset, IGLP-DP and IGLP-WE clearly outperforms all the other algorithms in the comparison.

Our extensive tests on both real-life networks and LFR benchmarks demonstrate IGLP-WE and IGLP-DP are the best performing algorithms overall. Moreover, unlike most existing algorithms in the literature that deliver a single community partition, IGLP-WE and IGLP-DP reveal the complete community hierarchy, which enables us not only to achieve high accuracy in terms of NMI scores but also to examine different levels of granularity to find the community partition of the highest modularity. They show superior performance consistently on undirected/directed and unweighted/weighted networks.

C. Space and Time Complexity Analysis

The space complexity is $O(Ln)$, where n is the number of nodes in the network, and L is the average length of influence vectors. L is determined by the average node out-degree b, depth limit d_{max} and and the community structure. Generally, the more cohesive community structure, the shorter influence vectors. In fact, the space complexity can be easily improved. Instead of simply following the node-index order to generate the influence vector for each node, we can implement the *breath-first* search algorithm to generate the influence vector of a node followed by generating the influence vectors of its neighbors. Once that node finds its SIN similarity to each of its neighbors, it is no need to keep the influence vector of that node. We can delete it and reclaim the space immediately.

To examine time complexity, we experiment on a set of undirected and unweighted LFR benchmarks with $\mu_t = 0.5$ and *min/max community size* = 20/100 (all other parameters are the same as described before). We vary the number of nodes n from 2,500 to 25,000 in an increment of 2,500, and generate 5 realizations for each value of n. All the experiments are carried out on a regular desktop PC with Intel(R) Core(TM) i5-4670 CPU @ 3.40 GHz and 8.0 GB memory under Windows 7 64-bit OS. Let m denote the number of edges in the network, K_{gt} denote the number of communities of ground truth, and K_{we} and K_{dp} denote the number of sub-communities in the initial community configuration given by IGLP-WE and IGLP-DP, respectively. We

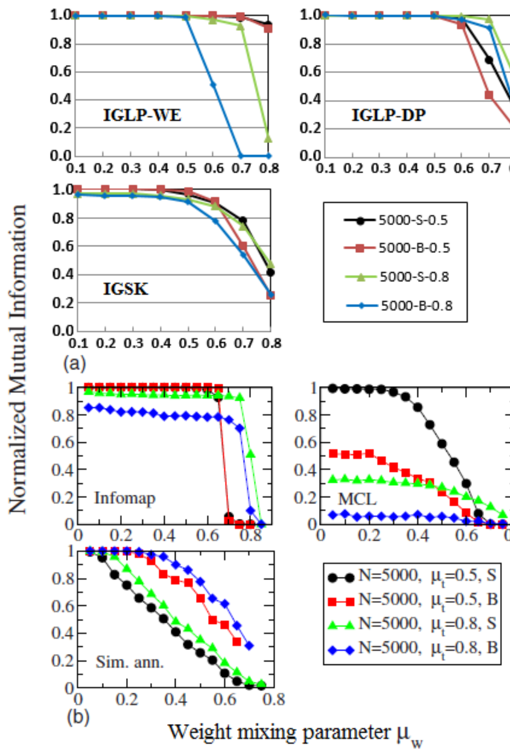

Figure 7. Performance comparison on undirected and weighted LFR benchmarks. Plot of *IGSK* is from [5], and plot (b) is from [12].

Table III
NETWORK INFORMATION AND EXPERIMENTAL RESULTS

n	m	K_{gt}	K_{we}	K_{dp}	NMI$_{we}$	NMI$_{dp}$
2,500	24,630	50	50	161	1	1
5,000	48,959	100	101	267	1	1
7,500	73,535	150	151	372	1	1
10,000	97,985	201	203	494	1	1
12,500	121,891	249	252	565	1	1
15,000	146,859	303	305	665	1	1
17,500	171,520	355	357	761	1	1
20,000	195,839	401	407	842	1	1
22,500	219,820	455	461	931	1	1
25,000	244,688	502	510	1,031	1	1

also include their respective NMI scores denoted by NMI$_{we}$ and NMI$_{dp}$. The experimental results are shown in Table III and Fig.8, in which we present not only the total time but also the time spent on generating the influence vectors, label propagation and hierarchical clustering, respectively.

As we can see, both IGLP-WE and IGLP-DP not only consistently achieve perfect NMI scores for all datasets, but also run fast. For the 25,000-node ($\sim$245,000 edges) dataset, IGLP-WE produces a community hierarchy of 510 levels in less than 250 seconds, and IGLP-DP delivers a more detailed hierarchical community structure of 1,031 levels within 300 seconds. They are actually faster than many existing algorithms such as *IGSK*, which provide only a single community partition. IGLP-WE and IGLP-DP incorporate the same depth-limited search algorithm [5] that generates the influence vectors efficiently. Its time complexity is $O(m)$ as shown in Fig.8. For both IGLP-WE and IGLP-DP, the time complexity is dominated by label propagation. IGLP-DP runs slightly faster than IGLP-WE w.r.t label propagation since IGLP-DP requires no convergent iteration. Actually, the label-propagation process itself is fast with both weighted ensemble and direct passing. Most of the time is spent on calculating the SIN similarity of each node to each of its neighbors, which has a time complexity of $O(Lm)$. Recall L is the average length of influence vectors, which is determined by the average node out-degree b, depth limit d_{max} and and the community structure. It is independent of n and m in general, which explains why the time complexity of label propagation is linear in m as shown in Fig.8.

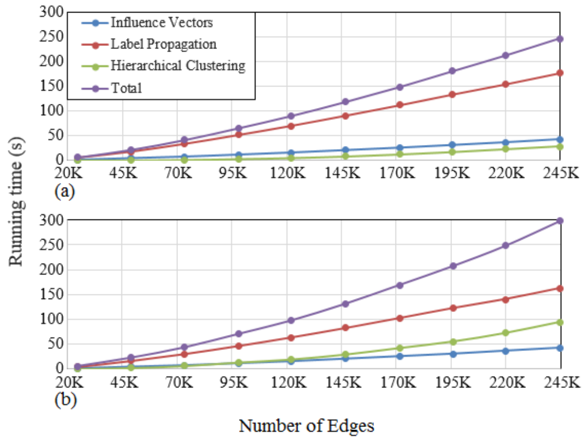

Figure 8. Time complexity: (a) IGLP-WE; (b) IGLP-DP.

The time complexity of hierarchical clustering, however, is slightly superlinear. For each iteration, we need to visit those boundary nodes and their neighbors to calculate the cluster proximity, which leads to a time complexity of $O(m)$. This is done $(K-1)$ times to build the hierarchy of communities agglomeratively, where K is the number of communities in the initial configuration given by label propagation. Hence, our hierarchical clustering has a time complexity of $O(K_{we}m)$ and $O(K_{dp}m)$ for IGLP-WE and IGLP-DP, respectively. Note that: K_{we} and K_{dp} are much smaller than m, and the number of boundary nodes is a monotonically decreasing fraction of n. K_{we} and K_{dp} are closely related to K_{gt}. As shown in Table III, K_{gt} increases monotonically as the network size increases when we fix both the topological mixing parameter and the min/max community size. While IGLP-WE consistently arrives at an initial community configuration that matches the ground truth closely, IGLP-DP zooms into deeper scales of the community structure and renders an initial configuration that includes many small sub-communities (roughly 2$\sim$3 times of the number of communities of the ground truth). And so IGLP-DP takes more time on hierarchical clustering than IGLP-WE. It is worth pointing out that: both IGLP-WE and IGLP-DP deliver a *complete* hierarchy of communities in the sense that no intermediate scale is missing above the initial configuration, as opposed to the well-known *Louvain method* [9]. From this point of view, $O(Km)$ is the best time complexity one can achieve to generate a community hierarchy of K levels.

V. Conclusion

In this paper, we present two novel algorithms, IGLP-WE and IGLP-DP, for finding hierarchical communities in complex networks. IGLP-WE and IGLP-DP naturally integrate undirected/directed and unweighted/weighted networks into one unified framework and consistently uncover a hierarchy of the community structure with excellent quality and high efficiency. In future work, one interesting direction is to extend them to overlapping community detection. Another advantage of IGLP-WE and IGLP-DP is that they can be easily parallelized. It is desirable to parallelize them for large-scale networks of millions of nodes and potential online community detection.

References

[1] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification." *J of Stat. Mech.*, p. P09008, 2005.

[2] M. Newman, "Analysis of weighted networks." *Phys. Rev. E*, vol. 70, p. 056131, 2004.

[3] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks." *Phys. Rev. E*, vol. 76, p. 03106, 2007.

[4] W. Wang and W. N. Street, "A novel algorithm for community detection and influence ranking in social networks." in *ASONAM*, 2014, pp. 555–560.

[5] W. Wang and W. N. Street, "Modeling influence diffusion to uncover influence centrality and community structure in social networks." *J of Social Network Analysis and Mining*, vol. 5, no. 1, 2015.

[6] I. Leung, P. Hui, P. Liò, and J. Crowcroft, "Towards real-time community detection in large networks." *Phys. Rev. E*, vol. 79, p. 066107, 2009.

[7] Y. Xing, F. Meng, Y. Zhou, M. Zhu, M. Shi, and G. Sun, "A node influence based label propagation algorithm for community detection in networks." *The Scientific World Journal*, 2014, 627581.

[8] J. Xie and B. K. Szymanski, "Labelrank: A stablized label propagation algorithm for community detection in networks." in *IEEE Network Science Workshop*, 2013, pp. 138–143.

[9] V. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "The louvain method for community detection in large networks." *J of Statistical Mechanics: Theory and Experiment*, vol. 10, p. P10008, 2008.

[10] J. Huang, H. Sun, J. Han, H. Deng, Y. Sun, and Y.Liu, "Shrink: A structural clustering algorithm for detecting hierarchical communities in networks." in *CIKM*, 2004.

[11] A. Lancichinetti, S. Fortunato, and J. Kertesz, "Detecting the overlapping and hierarchical community structure in complex networks." *New J of Physics*, vol. 11, 2009, 033015.

[12] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis." *Phys. Rev. E*, vol. 80, pp. 056 117(1–11), 2009.

[13] W. Zachary, "An information flow model for conflict and fission in small groups." *J of Anthropological Research*, vol. 33, pp. 452–473, 1977.

[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. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, p. 026113, 2004.

[16] M. Girvan and M. Newman, "Community structure in social and biological networks." *Proceedings of the National Academy of Sciences of USA*, vol. 99, no. 12, pp. 7821–7826, 2002.

[17] R. Guimera, L. Danon, D.-G. A., F. Giralt, and A. Arenas, "Self-similar community structure in a network of human interactions," *Phys. Rev. E*, vol. 68, p. 065103, 2003.

[18] M. Boguna, R. Pastor-Satorras, A. Diaz-Guilera, and A. Arenas, "Models of social networks based on social distance attachment," *Phys. Rev. E*, vol. 70, p. 056122, 2004.

[19] W. Liu, M. Pellegrini, and W. X., "Detecting communities based on network topology," *Scientific Report*, vol. 4, p. 5739, 2014.

[20] N. A. Christakis and J. H. Fowler, "The spread of obesity in a large social network over 32 years." *New England J of Medicine*, vol. 357, pp. 370–379, 2007.

[21] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society." *Nature*, vol. 435, pp. 814–818, 2005.

[22] M. Rosvall and C. Bergstrom, "Maps of random walks on complex networks reveal community structure." *Proceedings of the National Academy of Sciences of USA*, vol. 105, pp. 1118–1123, 2008.

[23] A. Clauset, M. Newman, and C. Moore, "Finding community structure in very large networks." *Phys. Rev. E*, vol. 70, p. 066111, 2004.

[24] P. Pons and M. Latapy, "Computing communities in large networks using random walks." *J of Graph Algorithms Applications*, vol. 10, no. 2, pp. 191–218, 2006.

[25] S. van Dongen, "Graph clustering by flow simulation." Ph.D. dissertation, University of Utrecht, 2000.

[26] M. Rosvall and C. Bergstrom, "An information-theoretic framework for resolving community structure in complex networks." *Proceedings of National Academy of Sciences*, vol. 104, pp. 7327–7331, 2007.

[27] R. Radicchi, C. Castellano, F. Cecconi, and D. Parisi, "Defining and identifying communities in networks." *Proceedings of National Academy of Sciences of USA*, vol. 101, pp. 2658–2663, 2004.

[28] R. Guimera and L. Amaral, "Functional cartography of complex metabolic networks," *Nature*, vol. 433, pp. 895–900, 2005.