Enhanced Distance Dynamics Model for Community Detection via Ego-Leader

Cai LiJun[1], Zhang Jing[2], Chen Lei[2*] and He TingQin[1]
[1] College of Information Science and Engineering, Hunan University
Changsha 410082, China
[e-mail: lijuncai_hnu@126.com, hetingqin@hnu.edu.cn]
[2] College of Electrical and Information Engineering, Hunan University
Changsha 410082, China
[e-mail: zhangj_hnu@163.com, chenleixyz123@hnu.edu.cn]
*Corresponding author: Chen Lei

Received June 26, 2017; accepted November 16, 2017; published May 31, 2018

Abstract

Distance dynamics model is an excellent model for uncovering the community structure of a complex network. However, the model has poor robustness. To improve the robustness, we design an enhanced distance dynamics model based on Ego-Leader and propose a corresponding community detection algorithm, called E-Attractor. The main contributions of E-Attractor are as follows. First, to get rid of sensitive parameter λ, Ego-Leader is introduced into the distance dynamics model to determine the influence of an exclusive neighbor on the distance. Second, based on top-k Ego-Leader, we design an enhanced distance dynamics model. In contrast to the traditional model, enhanced model has better robustness for all networks. Extensive experiments show that E-Attractor has good performance relative to several state-of-the-art algorithms.

Keywords: community detection, interaction model, complex network, Ego Network, Leader

1. Introduction

Most real-world systems, such as social networks and electric networks, can be modeled as networks and have some common features (also called community structure [1]). For example, these networks are scale-free, modular, small-world, and so on. Generally, a community is a set of nodes that is densely connected internally and loosely connected externally. Community detection is used to find the community structure of a network and to help us analyze the organizational structure, functional behavior, and evolution dynamics of the network. Recently, community detection techniques have been widely used in various domains, such as epidemiology networks, biological networks, metabolic networks, ecological webs, and particularly online social networks [2].

Over the years, many community detection algorithms have been developed to reveal the hidden community structure of networks, including modularity optimization algorithms, spectral clustering algorithms, graph partition algorithms, and dynamics algorithms [3]. Under the category of dynamics algorithms, there is a sub-category based on the synchronization phenomena; we call them synchronization-inspired dynamics algorithms. In 2010, inspired by the nature of synchronization phenomena, Böhm C [4] proposed a natural clustering method for any given dataset based on exploiting the synchronization dynamics. To adapt to various different datasets (text data or high-dimensional data), Böhm C selected and extended the Kuramoto Model [5-6] to simulate the synchronization dynamics. The Kuramoto Model can work well in various application environments (text data or high-dimensional data), especially in the case of a scattered dataset, where any two objects have no obvious association. Based on the clustering algorithm, many extended methods are proposed for different domains, such as high-dimensional data [5], data streams [6], graph clustering, anomaly mining [7], image segmentation [8], and biological information [9].

In the context of community detection, the dataset is a network graph. In the graph, edges indicate the obvious associations between data objects, and the properties of data objects are often ignored. In 2015, driven by network topology, Shao [10] addressed community detection by developing distance dynamics model to simulate the synchronization dynamics from a new view, namely, the edge, instead of traditional Kuramoto Model. In the new model, each edge of a network is associated with an initial distance. As time evolves, driven by network topology, each distance changes gradually. Finally, those objects with high similarity will synchronize together, and dissimilar objects will be far away from each other. The new model has several attractive benefits, such as “intuitive community detection,” “small community detection,” and “anomaly detection.” These benefits are very important and desirable in community detection. The model is described in more detail in Section 2.

A drawback of the distance dynamics model is that it is sensitive to the value of cohesion parameter λ, and deriving the best value of λ is difficult. The cohesion parameter λ is introduced as a threshold determining the negative or positive influence of an exclusive neighbor on the distance. With a high value of λ, nodes are more likely to be mutually exclusive during the synchronization dynamics process, and the distance dynamics model yields more communities. Conversely, when λ is small, nodes are more likely to synchronize together, and the model produces larger communities. By modulating cohesion parameter λ, the distance dynamics model allows analyzing the community structure from coarse to fine. However, different networks are very sensitive to parameter λ. There are three main difficulties associated with the importance of parameter λ.

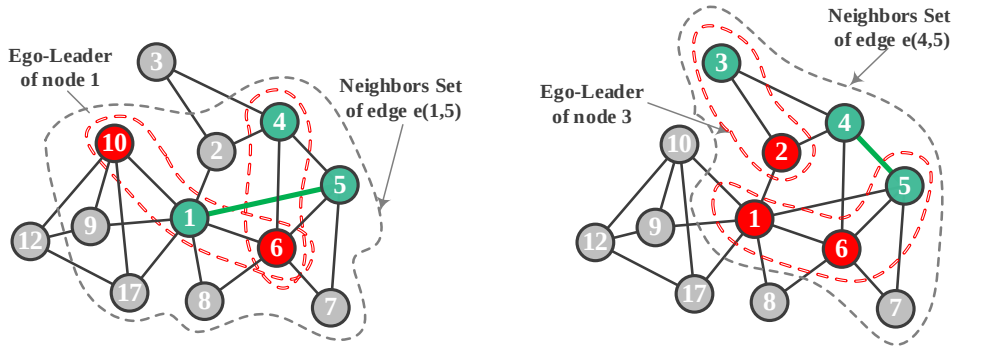
- Each network has its own density and properties, so each network needs a different value of λ. Although the region [0.4, 0.6] is recommended for the cohesion parameter λ in the

model, this range is only suitable for a small number of networks and does not work well in most networks.

- There is no powerful or robust method to obtain the best value of λ for different networks. To achieve a desired or perfect partitioning requires a long period of constant adjustment of λ with observation of the resulting community structure. Moreover, class labels (ground truth) are unknown for most real-world networks, making obtaining the best value of λ for a network very difficult.
- Minor changes to parameter λ may cause great differences in the resulting community structure.

1.1 Basic Idea

The motivation of this paper is to further optimize the robustness of the distance dynamics model. Toward that aim, we introduce Ego-Leader to replace the sensitive cohesion parameter λ in the distance dynamics model.



(a) Ego-Leader for exclusive neighbor pair ***en(1,4)*** on edge ***e(1,5)***

(b) Ego-Leader for exclusive neighbor pair ***en(3,5)*** on edge ***e(4,5)***

Fig. 1. Illustration of Ego-Leader.

By carefully analyzing synchronization process phenomena, we find that there are some leaders hiding in the neighbor set of each node. For each node, these leaders determine the movement of the node in the synchronization process. The leaders of a node remain constant throughout the synchronization process. Moreover, these leaders are local rather than global. That is, two nearby nodes may have completely different leaders. In this paper, we call these node leaders the Ego-Leaders. We introduce Ego-Leader to replace the cohesion parameter λ in the distance dynamics model to determine whether two indirectly connected nodes are similar and to estimate the negative or positive influence of one exclusive neighbor on the distance. When two indirectly connected nodes have common objects in their respective Ego-Leaders, we assume these two nodes are similar, and these common leaders will attract the two nodes to move towards themselves in the synchronization dynamics process. Thus, a node as the exclusive neighbor of another node will yield a positive influence on the distance, reducing the value of the distance. Conversely, when two indirectly connected nodes do not have common objects in an Ego-Leader, these two nodes are dissimilar, and the two nodes will keep away from each other in the synchronization process. Thus, a node as the exclusive neighbor of another node will yield a negative influence on the distance, increasing the value of the distance. By using the Ego-Leader, we can overcome the imperfections of the distance dynamics model caused by the cohesion parameter λ. There are two reasons for this improvement. First, Ego-leader is easily obtained. Second, Ego-Leader is local and common to all networks.

To describe the Ego-Leader more clearly, let us take a simple example. In **Fig. 1**, node 1 and node 4 are two indirectly connected nodes, node 4 is an exclusive neighbor for edge *e(1,5)*, and we call node 1 and node 4 an exclusive neighbor pair, denoted *en(1,4)*. Similarly, node 3 and node 5 are two indirectly connected nodes, node 3 is an exclusive neighbor for edge *e(4,5)*, and we call node 3 and node 5 the exclusive neighbor pair *en(3,5)*. For the pair *en(1,4)* in **Fig. 1(a)**, we find the Ego-Leader of node 1 and node 4 from the neighbor set (the gray dashed circle) of edge e(1,5). The Ego-Leader of node 1 encompasses node 10 and node 6 (two red nodes), while the Ego-Leader of node 4 is only node 6. Because node 6 is common to the Ego-Leaders of node 1 and node 4, we think node 1 is similar to node 4, and common leader node 6 will attract node 1 and node 4 to move towards itself in the synchronization process. For the edge *e(1,5)*, exclusive neighbor 4 will produce a positive influence and reduce the distance of *e(1,5)*. For contrast, let us look at the exclusive neighbor pair *en(3,5)* in **Fig. 1(b)**. The Ego-Leader of node 3 is node 2, and the Ego-Leader of node 5 comprises node 1 and node 6. Because there is not a common object in the Ego-Leaders of node 3 and node 5, we consider node 3 dissimilar to node 5, and node 3 keeps away from node 5 in the synchronization process. For the edge *e(4,5)*, exclusive neighbor node 3 will yield a negative influence, increasing the distance of *e(4,5)*.

1.2 Key contributions

Some significant contributions of this paper are as follows:

- A top-k level Ego-Leader discovery algorithm is developed to find the Ego-Leader of each node from the network topology.
- An enhanced distance dynamics model is designed, where Ego-Leader is used to replace the traditional cohesion parameter λ. Based on the enhanced model, a corresponding community detection algorithm is described, called E-Attractor.

The remainder of this paper is organized as follows. The traditional distance dynamics model is described in Section 2. Section 3 shows the Ego-Leader. Section 4 demonstrates our enhanced model and corresponding community detection algorithm (E-Attractor). Extensive experimental evaluation is presented in Section 5. Section 6 concludes this paper.

2. Traditional Distance Dynamics Model

Distance dynamics model is a typical dynamics model for community detection proposed in 2015 [10]. The process of the model is as follows: to start, each edge is associated with an initial distance; as time evolves, three different interaction models are used to expand or shrink each distance gradually; finally, all distances converge and the community structure of the network is naturally formed by removing the edges with the long distance value. In the following, we introduce the background and three interaction patterns of the distance dynamics model in detail.

2.1 Related Background

Definition 1 (undirected graph). Let $G=(V,E,W)$ be an undirected graph, where V is the node set, E is the edge set, and W is the corresponding weight set of all edges. Each edge $e(u,v)\epsilon E$ implies a communication connection between node u and v. $w(u,v)$ is the weight of corresponding edge $e(u,v)$.

Definition 2 (neighbors of node u). Given an undirected graph $G=(V,E,W)$, the neighbors of node u $N(u)$ is a node set that consists of node u and its connected nodes and is defined as follows:

$$N(u)=\{v\in V|\{u,v\}\in E\}\ \cup\ \{u\} \tag{1}$$

Definition 3 (Jaccard distance). Given an undirected graph *G=(V,E,W)*, the Jaccard distance [20] between node *u* and node *v* is defined as:

$$d(u,v)=1-\frac{|N(u)\cap N(v)|}{|N(u)\cup N(v)|} \tag{2}$$

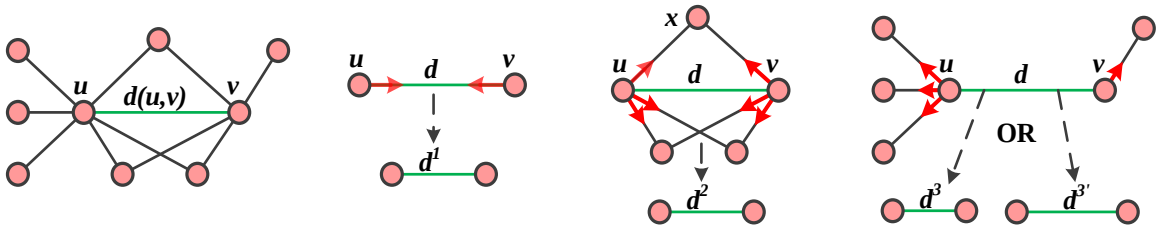
In the above equation, |*| indicates the size of set *, and *N(u)* comprises the neighbors of node *u*. The Jaccard distance measures the similarity of the neighbor sets of node *u* and node *v*. The more common neighbors the two nodes have, the greater similarity the two nodes have and the less the value of the Jaccard distance will be, and vice versa.

For the weighted undirected graph, because each edge has a different weight, the computational model of the Jaccard distance is different. The new computational model is extended as:

$$d(u,v)=1-\frac{\sum_{x\in N(u)\cap N(v)}\left(w(u,x)+w(v,x)\right)}{\sum_{\{x,y\}\in E;x,y\in N(u)\cup N(v)}w(x,y)} \tag{3}$$

2.2 Interaction Model

In the traditional distance dynamics model, three interaction patterns are proposed to simulate the distance dynamics. **Fig. 2** shows the three interaction patterns.



(a) Graph (b) Influence from direct linked nodes (c) Influence from common neighbors (d) Influence from exclusive neighbors

Fig. 2. Three distinct interaction patterns.

Pattern 1: Influence from directly linked nodes. The distance *d(u,v)* between node u and node *v* is obviously influenced by two directly linked nodes *u* and *v*. Through mutual interactions, one node attracts the other to move towards it, thus the distance *d(u,v)* decreases (see **Fig. 2 (b)**). Therefore, to characterize the change in the distance *d(u, v)*, *DI* is defined to represent the influence from two directly linked nodes, as follows:

$$DI=-\left(\frac{f(1-d(u,v))}{deg(u)}+\frac{f(1-d(u,v))}{deg(v)}\right) \tag{4}$$

In pattern *DI*, *deg(u)* indicates the degree of node *u*, and *f(·)* is a coupling function, where *sin(·)* is the default. The term *1-d(u,v)* implies the similarity of the inherent structure or properties of node *u* and *v*. The term *1/deg(u)* is a normalized factor that is used to account for the different influences between linked nodes with diverse degrees.

Pattern 2: Influence from common neighbors. The distance *d(u,v)* is influenced by common neighbors *CN=(N(u)-u)∩(N(v)-v)* of nodes *u* and *v*. Each common neighbor is a node connected to both node *u* and node *v*. In the dynamic interaction process, since each common

neighbor communicates with nodes *u* and *v*, the common neighbor will gradually attract two nodes to move towards itself, and thus lead to t decrease of the distance *d(u, v)* (see **Fig. 2 (c)**). The second interaction pattern is called *CI* and is defined as follows.

$$CI = -\sum_{x\in CN}\left(\frac{f\left(1-d\left(x,u\right)\right)\cdot\left(1-d\left(x,v\right)\right)}{deg\left(u\right)}\right)-\sum_{x\in CN}\left(\frac{f\left(1-d\left(x,v\right)\right)\cdot\left(1-d\left(x,u\right)\right)}{deg\left(v\right)}\right) \tag{5}$$

In pattern *CI*, for each common neighbor *x*, the two terms *(1-d(x,u))* and *(1-d(x,v))* are used to further measure the difference in influence between pattern *DI* and pattern *CI*.

Pattern 3: Influence from exclusive neighbors. The influence of exclusive neighbors is the third interaction pattern. Each exclusive neighbor in $EN(u)=N(u)-(N(u)\cap N(v))$ or $EN(v)=N(v)-(N(u)\cap N(v))$ connects to only one of nodes *u* or *v*, (see **Fig. 2 (d)**). And each exclusive neighbor and the indirectly connected end node compose an exclusive neighbor pair. In the dynamic interaction process, each exclusive neighbor will attract only one node (node *u* or *v*) to move towards itself. However, we do not know whether another node is close to the exclusive neighbor. To determine the positive or negative influence of an exclusive neighbor on the distance, a cohesion parameter λ is introduced to measure the similarity of two nodes in an exclusive neighbor pair and is defined as follows.

$$\rho\left(x,u\right)=\begin{cases}\left(1-d\left(x,v\right)\right) & \left(1-d\left(x,v\right)\right)\geq\lambda\\ \left(1-d\left(x,v\right)\right)-\lambda & \text{otherwise}\end{cases} \tag{6}$$

In the above equation, $\rho(x,u)$ indicates the positive or negative influence from exclusive neighbors $EN(u)$ on the distance *d(u, v)*. The third interaction pattern, *EI*, based on the cohesion parameter λ, is defined as:

$$EI = -\sum_{x\in EN(u)}\left(\frac{f\left(1-d\left(x,u\right)\right)\cdot\rho\left(x,u\right)}{deg\left(u\right)}\right)-\sum_{y\in EN(v)}\left(\frac{f\left(1-d\left(y,v\right)\right)\cdot\rho\left(y,v\right)}{deg\left(v\right)}\right) \tag{7}$$

Finally, considering the three interaction patterns together, the dynamics of distance *d(u,v)* between nodes *u* and *v* over time is governed by:

$$d\left(u,v,t+1\right)=d\left(u,v,t\right)+DI\left(t\right)+CI\left(t\right)+EI\left(t\right) \tag{8}$$

where *d(u,v,t+1)* is the new distance at time step *t+1*. *DI(t)*, *CI(t)* and *EI(t)* are the three different influences from two directly linked nodes, common neighbors, and exclusive neighbors, respectively.

3. Ego-Leader

In this paper, for any node, Ego-Leader is the set of nodes with the greatest power in the node's neighbor set. To find the Ego-Leader of a node, we must select a way to measure the capacity of each neighbor. Generally, degree centrality, betweenness centrality, closeness centrality, subgraph centrality, and eigenvector centrality are five classical measurements of centrality [11]. However, these five metrics are global metrics. They represent the capacity of a node in the whole network. In this paper, Ego-Leader is a local metric and applies only to one node rather than the whole network. That is, the scope of activity of Ego-Leader is the neighbor set of a node. Therefore, traditional metrics of measuring node capacity cannot work for determining the Ego-Leader. We need to find a new, local metric of measuring node capacity.

3.1 Asymmetric edge clustering coefficient

In synchronization dynamics, an Ego-Leader will attract some nodes to synchronize with itself. That is, for any node, its Ego-Leader has a greater probability of clustering together with the

node (some nodes synchronize together). Hence, we consider the clustering relationship to be a good metric for Ego-Leader. In this paper, the asymmetric edge clustering coefficient is proposed as the metric to measure the capacity of a neighbor. In the following, we describe the concept of an edge clustering coefficient, then present the asymmetric edge clustering coefficient.

The notion of edge clustering coefficient originates in the research on community discovery in complex networks [11]. It characterizes the closeness between the two end nodes of an edge and the other nodes around them. The edges with higher clustering coefficients tend to hide in the community structure of the network. This view has been proved in many works [12]. The edge clustering coefficient is a measure that can both evaluate the importance of edges in the network and describe how close an edge's two end nodes are.

Definition 4 (Edge clustering coefficient): for an edge $e(u,v)$ connecting node u and node v, we observe how many other nodes adjoin both u and v. The edge clustering coefficient of $e(u,v)$ can be defined as

$$ECC(u,v)=\frac{z_{u,v}}{min(deg(u)-1,deg(v)-1)} \tag{9}$$

where $z_{u,v}$ denotes the actual number of triangles in the network topology that include edge $e(u,v)$, $deg(u)$ and $deg(v)$ are degrees of node u and node v, respectively, and $min(deg(u)\text{-}1, deg(v)\text{-}1)$ is the maximum number of triangles containing edge $e(u,v)$. When either end node (u or v) is a leaf node, the edge clustering coefficient defaults to 0.

The traditional edge clustering coefficient is a symmetric local metric, $ECC(u,v)=ECC(v,u)$, but Ego-Leader is asymmetric. That is, node v may be in the Ego-Leader of node u when node u is not in the Ego-Leader of node v. Therefore, the traditional edge clustering coefficient cannot fully address the asymmetry of Ego-Leader. Therefore, we revise the traditional edge clustering coefficient and propose an asymmetric edge clustering coefficient.

Definition 5 (Asymmetric edge clustering coefficient): for an edge $e(u,v)$ connecting node u and node v, the asymmetric edge clustering coefficient can be defined as.

$$AECC(u,v)=\frac{z_{u,v}}{deg(v)-1} \tag{10}$$

The asymmetric edge clustering coefficient indicates the influence of node u on node v and the degree of overlap in the two nodes' neighbor sets. The higher $AECC(u,v)$ is, the greater the influence of node u on node v is. To illustrate *AECC* more clearly, let us take a simple example. In **Fig. 3**, we try to calculate the *AECC* of the thick green edge e(6,0). n6 and n0 are the two end nodes of edge e(6,0), and the degrees of these two nodes are 4 and 6, respectively. From the figure, we can see that there are only two triangles (Δ_{609} and Δ_{607}) including the edge e(6,0). Therefore, $AECC(6,0)=2/(6\text{-}1)=0.4$, and $AECC(0,6)=2/(4\text{-}1)=0.67$. That is, $AECC(6,0)$ is not equal to $AECC(0,6)$, proving the asymmetric edge clustering coefficient is an asymmetric local metric and completely meets the requirements of Ego-Leader.

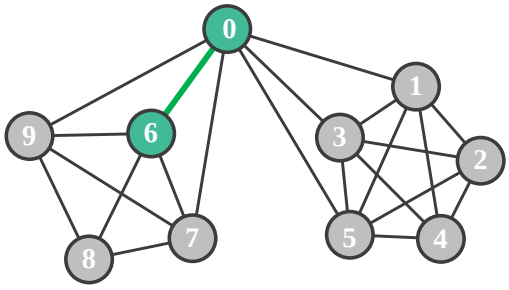


Fig. 3. Example of 10-node and 23-edge graph.

3.2 Ego-Leader and its features

In summary, we use the asymmetric edge clustering coefficient *AECC* to find the Ego-Leader of any node in the network. So far, we have a clear understanding of the Ego-Leader of a node. The Ego-Leader of a node consists of multiple neighbors having the greatest *AECC* value. In addition, the number of nodes in a node's Ego-Leader is never more than the number of neighbors of the node. Hence, the Ego-Leader has the following important features.

(1) Ego-Leader is asymmetric.

This feature is very important and implies the intrinsic Leader-Follower relationship. The importance of a Leader to its Followers is obviously greater than the importance of any Follower to its Leader. Moreover, the proof for this feature is self-evident.

(2) Ego-Leader is local and static.

Proof: as we have shown, the Ego-Leader of a node comprises the nodes of greatest power in its neighbor set. That is, any Ego-Leader only works for one node, and its scope of activity is the neighbor set of the node. Therefore, the Ego-Leader is a local metric and strongly dependent on the network topology. Because the network topology is static, the Ego-Leader of a node is also static and does not incur any change.

In addition, this feature has been proved in many works [15-16].

(3) The centrality of Ego-Leader in its group is positively related to the objective performance of that leader's group.

This feature indicates that the greater the power of a group leader is, the greater the probability that all members of this group cluster together. This feature has also been proved in previous work [17].

3.3 Finding top-k level Ego-Leader

In this paper, the goal of introducing our Ego-Leader is to replace the traditional cohesion parameter λ in determining the influence of an exclusive neighbor on distance. However, in the traditional model, cohesion parameter λ has a useful feature. That is, by modulating the cohesion parameter λ, the distance dynamics model allows analyzing the network's community structure from coarse to fine. With a higher value of λ, the distance dynamics model yields more communities, while larger communities are produced with a lower value of λ. To retain this feature, we use parameter k to dynamically adjust the number of objects in the Ego-Leader of a node. Hence, we select the top-k level objects having greatest power from the neighbor set to form the Ego-Leader of a node. When two neighbors have the same influence (*AECC*) on a node, those two neighbors belong to the same level.

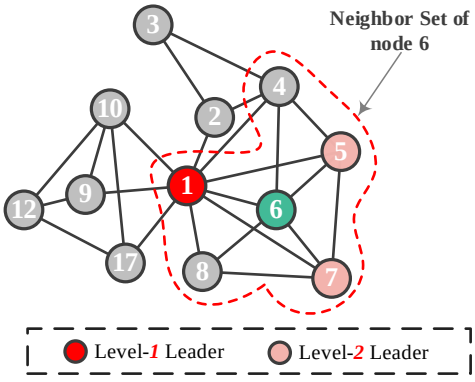


Fig. 4. Top-2 level Ego-Leader.

To describe the top-k level Ego-Leader concept more clearly, let us take a simple example. In **Fig. 4**, we try to find the top-2 level Ego-Leader for green node 6. We search the neighbor set of node 6, as shown in the red dashed circle of **Fig. 4**. From the neighbor set, we find node 1 has the highest *AECC* value because there are 4 triangles (Δ_{146}, Δ_{156}, Δ_{176}, and Δ_{186}) containing edge e(1,6). Thus, the dark red node, 1, is the level-1 Ego-Leader of node 6. Likewise, two light red nodes 5 and 7 have the second highest *AECC* value, *AECC*(5,6)=*AECC*(7,6), because there are 3 triangles between node 5 (node 7) and node 6, corresponding to Δ_{516}, Δ_{546}, Δ_{576} (or Δ_{716}, Δ_{756}, Δ_{786}). Thus, the light red node 5 and node 7 comprise the level-2 Ego-Leader of node 6. That is, for node 6, the top-2 level Ego-Leader includes node 1, node 5, and node 7.

By modulating k, we can easily adjust the scale of node Ego-Leaders, and this modified distance dynamics model also allows analyzing network's community structure from coarse to fine. The higher the value of k is, the bigger the scale of Ego-Leader is, increasing the probability that an exclusive neighbor yields positive influence and simultaneously increasing the size of the community found by the distance dynamics model. Conversely, the model produces more communities with a lower value of k. In contrast to the traditional cohesion parameter λ, we find the top-k Ego-Leader has the following advantages.

- ***It makes the distance dynamics model more robust***. Ego-Leader is local and common for all networks, so it is not sensitive to different networks. Moreover, Ego-Leader is strongly dependent on network topology, so it is more suitable than cohesion parameter λ for the distance dynamics model and makes the model more robust.
- ***It is easier to obtain the best value of k***. To get a perfect network community structure, we need to get the best value of parameter k. For that, we only need to adjust the value of k several times, generally fewer than 6, from top-3 to top-8. By contrast, to get the best value of cohesion parameter λ for different networks requires a long period of constant, manual adjustment of λ.
- ***It is easier to understand and accept***. In contrast to traditional cohesion parameter λ, Ego-Leader is easier to understand and accept in the distance dynamics model because the synchronization dynamics process in the distance dynamics model has been regarded as the Leader-Follower process in previous research [17-18].
- ***It is fast, and no extra time is required***. Because the asymmetric edge clustering coefficient is a local metric, the calculation speed of Ego-Leaders is very fast. Moreover, the calculation of Ego-Leaders can be merged into the initial distance calculation phase of the distance dynamics model, so no extra time is required for calculating the nodes' Ego-Leaders.

In summary, the process of finding top-k level Ego-Leaders is very simple. **First**, each edge of the network is scanned sequentially. For each node, we build a queue to sort the neighbors according to their capacity. For edge $e(u,v)$, the asymmetric edge clustering coefficients $AECC(u,v)$ and $AECC(v,u)$ are calculated using Eq.10 to measure the influence of node u (or node v) to node v (or node u). According to the value of $AECC(u,v)$ (or $AECC(v,u)$), neighbor u (or v) is stored into the queue of node v (or u) in descending order. **Second**, we judge whether the AECCs of all neighbors of node u (node v) have been calculated. When the AECC has been calculated for all neighbors of node u (or node v), the top-k level neighbors are chosen from the queue of node u (or node v) as its Ego-Leader. When the value of k is greater than the degree of node u (or node v), all neighbors are selected to form the Ego-Leader. **Third**, when all edges of the network have been scanned, the process is over.

3.4 Simplify validation

To validate the effectiveness of Ego-Leader, we select two classic networks to evaluate whether Ego-Leader correctly determines the influence of an exclusive neighbor on distance, thus replacing traditional cohesion parameter λ. Two networks, the Zachary's karate club and Dolphins networks, are publicly available from the UCI network data repository (https://network data.ics.uci.edu/index.php). The process of validation is as follows. We randomly sample 30 indirectly connected node pairs from the Zachary's karate club network and 100 indirectly connected node pairs from the Dolphins network. We label each indirectly connected node pair as an exclusive neighbor pair because one node must be the exclusive neighbor of another node in the pair. We use top-2 Ego-Leader and cohesion parameter λ (λ=0.5) as two distinct ways to determine whether the two nodes of an exclusive neighbor pair are similar. If two nodes in the exclusive neighbor pair are similar, the exclusive node will produce a positive influence on distance. If the nodes are not similar, the exclusive node will yield a negative influence on the distance.

Table 1. Validation of Ego-Leader on karate club network.

Exclusive neighbor pair	**Ego-Leader**			**Cohesion parameter λ**		
	first node	second node	similar	Jaccard distance	λ	similar
(23↔30)	[29, 27]	[8, 32]	False	0.77	0.5	False
(23↔15)	[29, 27]	[32]	False	0.71	0.5	False
(26↔9)	[29]	[2]	False	0.8	0.5	False
(7↔13)	[3, 1, 2]	[3, 1, 2]	**True**	0.42	0.5	**True**
(8↔12)	[30, 32, 2]	[3]	False	0.88	0.5	False
(10↔6)	[4, 5]	[5, 16, 4]	**True**	0.5	0.5	**True**
(10↔7)	[4,5]	[3, 1, 2]	False	0.88	0.5	False
(12↔17)	[3]	[1]	False	0.8	0.5	False
(13↔7)	[3, 1, 2]	[3, 1, 2]	**True**	0.43	0.5	**True**
(17↔21)	[0]	[0]	**True**	0.5	0.5	**True**

The first network is the well-known Zachary's karate club network, which consists of 34 vertices and 78 undirected edges. Each node represents a member of the club, and each edge represents a tie between two members. For convenience, **Table 1** shows the results of only 10 exclusive neighbor pairs. From the table, we easily observe that Ego-Leader produces nearly the same choices as cohesion parameter λ.

Table 2. Validation of Ego-Leader on Dolphins network.

Exclusive neighbor pair	**Ego-Leader**			**Cohesion parameter λ**		
	first node	second node	similar	Jaccard distance	λ	similar
(33↔44)	[14, 37, 16]	[38, 20, 2]	False	0.86	0.5	False
(23↔1)	[51, 45]	[27, 26, 54]	False	0.92	0.5	False
(6↔5))	[13, 57, 9]	[9, 13, 57]	**True**	0.5	0.5	**True**
(40↔20)	[14, 0, 37]	[28, 16, 44]	False	0.94	0.5	False

(41↔26)	[54, 13, 57]	[27, 25]	False	0.89	0.5	False
(1↔57)	[27, 26, 54]	[13, 9, 6]	False	0.81	0.5	False
(2↔30)	[10, 42, 61]	[47, 19, 28]	False	0.9	0.5	False
(43↔0)	[33, 38]	[10, 47, 40]	False	0.93	0.5	False
(15↔51)	[24, 18]	[18, 4, 11]	**True**	0.43	0.5	**True**
(10↔20)	[42, 0, 2]	[28, 16, 44]	False	0.93	0.5	False

The second network is the Dolphins network. It is an undirected social network based on the frequent associations between 63 dolphins in a community living off Doubtful Sound (New Zealand). Similar to the karate club network, **Table 2** shows only the results of 10 exclusive neighbor pairs. From the table, we can draw the same conclusion: Ego-Leader produces the same or similar results as cohesion parameter λ.

Based on the above evaluations, we believe Ego-Leader can be introduced into the distance dynamics model and replace traditional cohesion parameter λ to determine the influence of an exclusive neighbor on distance.

4. E-Attractor: Enhanced Distance Dynamics Model for Community Detection via Ego-Leader

In this section, we describe an enhanced distance dynamics model based on Ego-Leader. Then, based on the enhanced model, we present a corresponding community detection algorithm called E-Attractor.

4.1 Enhanced distance dynamics model

In the traditional distance dynamics model, three interaction patterns (DI, CI, and EI) are used to simulate the distance dynamics. The details are provided in Section 2. In contrast to the traditional model, our enhanced distance dynamics model introduces Ego-Leader to replace the cohesion parameter λ. More specifically, the top-k Ego-Leader of each node is used to determine whether two indirectly connected nodes are similar and to decide the influence of an exclusive neighbor on distance. Because the traditional DI and CI patterns do not use cohesion parameter λ, these two patterns are unchanged in our enhanced model. We only improve the EI pattern via Ego-Leader.

New Pattern 3: Influence from exclusive neighbors. The influence from the exclusive neighbors is the third interaction pattern. In the exclusive neighbor sets $EN(u)=N(u)-(N(u)\cap N(v))$ and $EN(v)=N(v)-(N(u)\cap N(v))$, each node only connects to one end node, u or v, of edge e(u,v) (please see **Fig.2 (d)**). Each exclusive neighbor and corresponding indirectly connected end node form an exclusive neighbor pair. In the interaction process, each exclusive neighbor will only attract the node connected with it (node *u* or node *v*). However, we do not know whether another node (indirectly connected node) will be close to the exclusive neighbor. To determine the positive or negative influence of an exclusive neighbor on distance d(u,v), Ego-Leader is used and defined as:

$$\sigma(x,u)=\begin{cases}(1-d(x,v)) & \left|EgoL_k(x)\cap EgoL_k(v)\right|\geq 0\\ -(1-d(x,v)) & \text{otherwise}\end{cases} \tag{11}$$

In the above equation, $EgoL_k(x)$ is the top-k Ego-Leader of node x. The function $\sigma(x,u)$ not only indicates the direction of influence (positive or negative) of exclusive neighbor x on distance but also shows the strength of this influence. Hence, the new third interaction pattern,

NEI, is defined as:

$$NEI = -\sum_{x\in EN(u)}\left(\frac{f\left(1-d\left(x,u\right)\right)\cdot\sigma\left(x,u\right)}{deg\left(u\right)}\right)-\sum_{y\in EN(v)}\left(\frac{f\left(1-d\left(y,v\right)\right)\cdot\sigma\left(y,v\right)}{deg\left(v\right)}\right) \quad (12)$$

Finally, considering the three interaction patterns together, the dynamics of distance *d(u,v)* between nodes *u* and *v* over time is governed by:

$$d\left(u,v,t+1\right)=d\left(u,v,t\right)+DI\left(t\right)+CI\left(t\right)+NEI\left(t\right) \quad (13)$$

where *d(u,v,t+1)* is the new distance at time step *t+1*. *DI(t)*, *CI(t)* and *NEI(t)* indicate three different influences from two directly linked nodes, common neighbors, and exclusive neighbors, respectively.

4.2 E-Attractor algorithm

In this section, we give a comprehensive description of the E-Attractor algorithm. The E-Attractor process is very simple, consisting mainly of the following three steps.

1. At the start time (t=0), without any interaction, each edge is associated with an initial distance using the Jaccard-distance function (Eq.2 or Eq.3). At the same time, the two asymmetric edge clustering coefficients of each edge are calculated and stored into the queues of their respective end nodes. When all neighbors of an end node have been calculated, then the top-k level Ego-Leaders are chosen to form the Ego-Leader set. More detail is provided in Section 3.
2. The dynamic interaction process is initiated. As time evolves, driven by the network topology, each distance changes gradually under the influence of three different interaction patterns (DI, CI, and NEI). In particular, the nodes with higher similarity synchronize faster, and the distances between these nodes decrease faster. At the same time, the nodes with higher dissimilarity separate faster, and the distances between these nodes increase faster. After multiple time steps, all distances converge, either to 0 or 1, ending the dynamic interaction process.
3. After the dynamic interaction process, the community structure of the network is naturally detected by removing all edges with a distance value of 1.

4.3 Time complexity

The time complexity of E-Attractor is two-fold. First, each edge is associated with an initial distance, and the Ego-Leader of each node is calculated. Therefore, the time complexity is *O(|E|)*, where *|E|* is the number of edges. Second, a dynamic interaction process is executed with T time steps. Thus, time complexity of this process is *O(T*U*|E|)*, where *U* is the average number of exclusive neighbors of two linked nodes. In total, the time complexity of E-Attractor is *O(|E|+T*U*|E|)*.

5. Experimental Evaluation

5.1 Evaluation Setup

Comparison Algorithms. To evaluate the performance of the E-Attractor algorithm, we select five representative community detection algorithms as competitors. All comparison algorithms are listed in **Table 3**, where the InfoMap, FastGreedy and Louvain algorithms are considered to be the best algorithms for disjoint community detection [3,21], the LPA algorithm has a high speed, and the Attractor algorithm is a native algorithm built on the distance dynamics model. For all community detection algorithms, recommended parameter

defaults are used to get the best experimental results.

Table 3. Comparison Algorithms.

Algorithm	Full Name	Implement
InfoMap [19]	Maps of random walks on complex networks reveal community structure.	Python
FastGreedy [13]	Finding community structure in very large networks.	Python
LPA [24]	Label propagation through linear neighborhoods.	Python
Louvain [14]	Fast unfolding of communities in large networks.	Python
Attractor [10]	Community Detection based on Distance Dynamics.	Python
E-Attractor	Enhanced distance dynamics model for community detection via Ego-Leader.	Python

Evaluation metrics. To extensively compare different community detection algorithms with respect to effectiveness, we select two widely used metrics to evaluate the cluster quality. (1) The first metric is Normalized Mutual Information (NMI) [23], which is defined in the context of classical clustering to compare two different partitions of one dataset by measuring how much information they have in common. (2) The second metric is the popular Adjusted Rand index [22]. It calculates the total number of pairs that belong to the same cluster, or to different clusters, comparing expected clusters and clustering results. All metrics scale between 0 and 1 for a random or a perfect clustering result, respectively.

Experimental Platform. As the experimental platform, we rented a high-performance server (IBM x3650 m4) from National Super Computing Center of Changsha, located in Hunan province, China. The server comprises one CPU with 8 cores (Intel Xeon Processor E5-2603) and 16GB main memory. All algorithms are run on the high-performance server using the Windows server 2012 operating system. The E-Attractor and Attractor algorithm are implemented in Python. For the other four algorithms, we have downloaded the Python implementations from the official websites of the corresponding authors.

5.2 Sensitivity of parameter k

The first objective of experimental evaluation is to observe and validate the sensitivity of parameter k in the top-k Ego-Leader. Parameter k is defined to determine the scale of the nodes' Ego-Leaders and, further, to decide the direction of the influence of an exclusive neighbor on the distance: positive influence or negative influence. Generally, when parameter k has higher value, each node has more members in its Ego-Leader, an exclusive neighbor has a larger probability of exerting a positive influence, and the E-Attractor algorithm yields bigger communities. Conversely, with lower k, each node has fewer Ego-Leader members, an exclusive neighbor is more likely to exert a negative influence, and E-Attractor yields more communities. By modulating parameter k, E-Attractor allows analyzing network community structure from coarse to fine. Moreover, parameter k is robust and easy to tune. To evaluate the sensitivity of parameter k, we adopt the Jazz network as the experimental dataset and observe the change in community structure when modifying the value of k gradually.

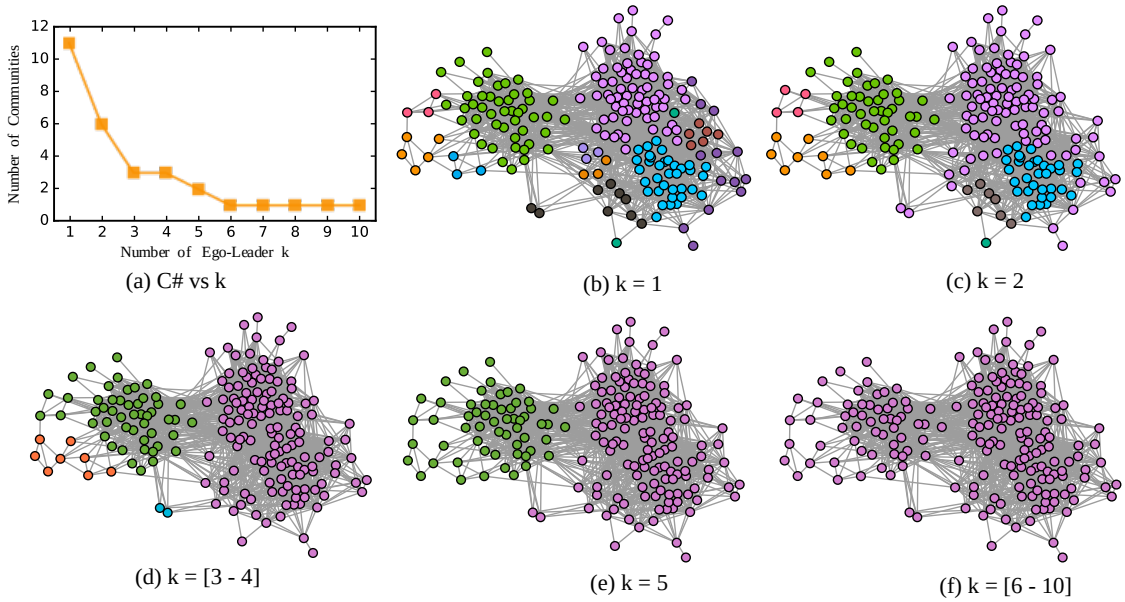


Fig. 5. Top-2 level Ego-Leader Sensitivity of parameter k on Jazz network.

The Jazz network is a collaboration network used by jazz musicians. Each node represents a jazz musician, and an edge connects two musicians who have played together in a band. The network was collected in 2003. **Fig. 5 (a)** plots the number of communities for k ranging from 1 to 10. The resulting community structures of the Jazz network with respect to distinct parameters are further illustrated in **Fig. 5 (b)** to **Fig. 5 (f)**. As shown in the plot, when the value of k is smaller, E-Attractor yields more communities. Conversely, E-Attractor yields larger communities with higher values of k. Moreover, we can see that E-Attractor yields a perfect partitioning with parameter k over a long stable range (2 - 5).

We also carry out numerous experiments on several synthetic networks to observe the resulting changes in communities discovered. Based on extensive experiments, we find that E-Attractor usually produces a good clustering result within the range k=[3-8]. Moreover, compared with the native Attractor algorithm requiring cohesion parameter λ, E-Attractor with parameter k is more robust. E-Attractor only needs to adjust parameter k 1~6 times to obtain a perfect partitioning of different networks, but Attractor needs to adjust parameter λ 8~60 times to obtain a perfect partitioning of different networks. We set parameter k=5 as the default value in the following experiments.

5.3 Synthetic network

(1) Network generation

To compare the performance of various community detection algorithms, we use the LFR benchmark to generate several synthetic networks featuring distinct characteristics. The LFR benchmark generation model is defined as LFR(C#, Cs, K_{max}, μ), where C# indicates the number of communities; Cs represents the number of nodes in one community; K_{max} means the maximum degree of a node; and μ is the mixing parameter indicating the fraction of a node's links outside its community, which is used to control the difficulty of community separation.

Table 4. Synthetic network.

Network	Node	Edge	Average degree	u	K_{max}	Cs	$C\#$
LFR1	140	917	13.1	0.25	18	50	4
LFR2	691	5071	14.7	0.25	18	100	8
LFR3	3180	33696	21.2	0.2	25	200	20
LFR4	3746	39011	20.8	0.2	25	100	50
LFR5	31194	141785	9.0	0.15	12	500	80
LFR6	30382	137955	9.0	0.2	12	500	120
LFR7	42170	271326	12.9	0.15	15	300	180
LFR8	59662	373362	12.5	0.1	15	300	250

Based on these four parameters, eight synthetic networks with ground-truth are generated, as listed in **Table 4**. For fair comparison, the eight synthetic networks have different network scale, community numbers (C#), average node degrees, and number of noise edges. The purpose of this generating scheme is to make synthetic networks that are more consistent with real-world networks. By modulating parameters C# and Cs, we cause each synthetic network to have a different network scale (nodes and edges) and community numbers (C#). Modulating parameter K_{max} gives each synthetic network a different average node degree. Modulating parameter μ gives each synthetic network a different number of noise edges in each community.

(2) Community detection performance

The second objective of experimental evaluation is to test the community detection performance of various algorithms on LFR synthetic networks, based on NMI, ARI, and computation time.

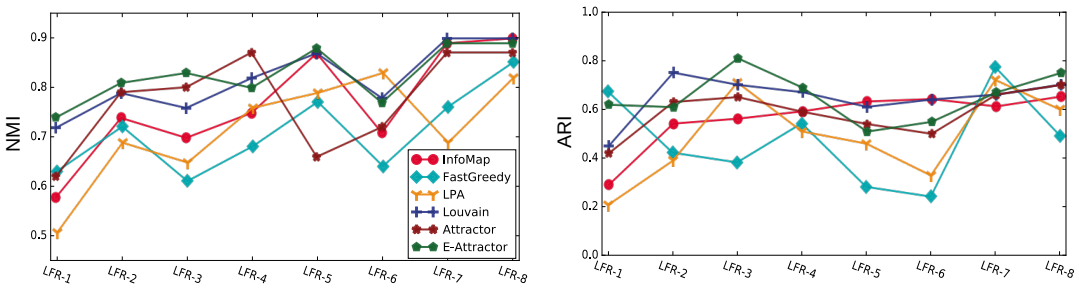


Fig. 6. Community detection performance of different algorithms on LFR networks.

Fig. 6 plots the community detection performance of various algorithms on LFR synthetic networks with respect to two distinct metrics. **Fig. 6 (a)** shows the respective NMI values, and **Fig. 6 (b)** shows the respective ARI values of the tested algorithms. From **Fig. 6**, we make the following observations. (1) For the NMI, the six community detection algorithms have good results, defined as an average value of NMI greater than 0.6. Comparing the six algorithms, we observe that the E-Attractor, Attractor, and Louvain algorithms have the best results and stability; the InfoMap algorithm is next; the FastGreedy and LPA algorithms are the worst. (2) For the ARI, the trend lines of the six algorithms are very uneven and imply the performance

of the algorithms is very unstable on different LFR networks. For instance, on the high noise networks (LRF-1~LFR-4, parameter u≥0.2), the ARI fluctuation is obvious for all six algorithms, where E-Attractor and Louvain are best, Attractor and InfoMap are next, and FastGreedy and LPA are worst. On the low noise networks (LRF-5~LFR-8, parameter u<0.2), Louvain and InfoMap are best, E-Attractor and Attractor are next, and LPA and FastGreedy are worst. (3) Merging NMI and ARI together, we see that the E-Attractor algorithm can perform better than the other five algorithms on the high noise networks (LFR-1~LFR-4). On the low-noise networks (LFR-5~LFR-8), the performance of the E-Attractor algorithm is poorer than that of the Louvain and InfoMap algorithms, but better than that of the Attractor, LPA, and FastGreedy algorithms.

Table 5. Computation time of various algorithms on LFR networks (ms).

Network	LFR-1	LFR-2	LFR-3	LFR-4	LFR-5	LFR-6	LFR-7	LFR-8
FastGreedy	803	4294	23095	24170	122141	101517	195473	244056
InfoMap	500	3738	19175	20002	86473	84012	142374	187721
Louvain	129	329	12143	1357	6029	5679	9521	13608
LPA	57	127	489	501	2316	2143	3897	5327
Attractor	575	3157	16608	17564	75561	73697	121124	165086
EAttractor	792	3938	20312	21197	90875	89414	149021	205881

Table 5 lists the computation time of the six algorithms on the LFR networks. As shown in **Table 5**, the computation time of the LPA algorithm is least, followed by the Louvain algorithm, then InfoMap, Attractor, and E-Attractor are next and have very similar times; FastGreedy is slowest. Contrasting the LPA and InfoMap algorithms, we observe that the computation time of InfoMap is close to 30~40 times that of LPA. Moreover, focusing on the Attractor, InfoMap and E-Attractor algorithms, the computation time of InfoMap is bigger than that of Attractor but smaller than that of E-Attractor.

5.4 Real-world network

(1) Network selection

To further evaluate the performance of the various community detection algorithms, we choose six typical real-world networks with ground-truth for more experiments, as listed in **Table 6**. All chosen real-world networks are publicly available from the UCI network data repository (https://networkdata. ics.uci.edu/index.php) and Stanford large network dataset collection (http://snap.stanford.edu/data/). The six real-world networks belong to different network types, where karate is a social network, polbooks is a book network, adjnoun is a word association network, football is a football network, polblogs is a blog network, and DBLP is a collaboration network. Moreover, the six real-world networks have varying network density, where karate, adjnoun, and DBLP are sparse networks, and football, polbooks and polblogs are dense networks.

Table 6. Real-world networks.

Network	Node	Edge	Average degree	Network type
karate	34	78	4.6	Social

polbooks	105	441	8.4	Book
adjnoun	112	425	7.6	Word
football	115	613	10.7	Football
polblogs	1490	19090	22.4	Weblogs
DBLP	317080	1049866	6.6	Collaboration

(2) Community detection performance

The third objective of experimental evaluation is to test the community detection performance of various algorithms on real-world networks, based on NMI, ARI, and computation time.

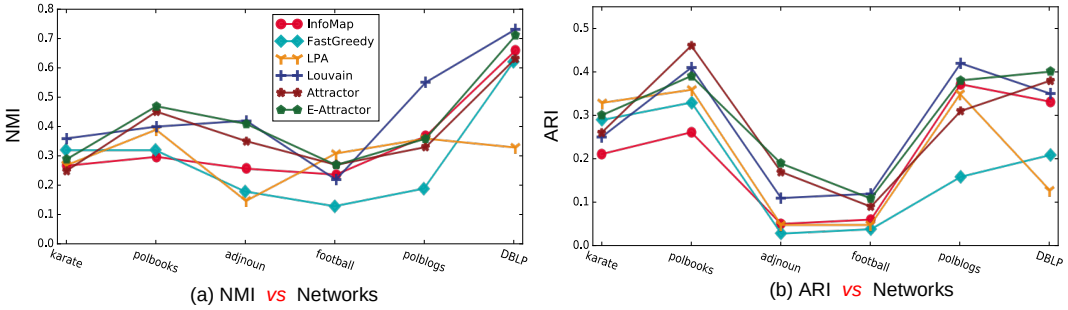


Fig. 7. Community detection performance of different algorithms on real-world networks.

Fig.7 shows the community detection performance of six algorithms on real-world networks, where **Fig.7 (a)** plots the NMI results, and **Fig.7 (b)** plots the ARI results. From **Fig.7**, we make the following observations. (1) For the NMI, the six algorithms display different benefits. Of the six algorithms, E-Attractor and Louvain generally provide the best NMI, followed by InfoMap, Attractor, and LPA, with FastGreedy performing most poorly. Moreover, the E-Attractor, Attractor and Infomap algorithms are more stable than the other three algorithms. (2) With respect to the ARI, the trend lines of the six algorithms fluctuate widely, and the average ARI value is less than 0.5 for all six algorithms. Overall, the ARI values of E-Attractor and Louvain are better than those of the other four algorithms. (3) When we consider NMI and ARI together, E-Attractor, Attractor and Louvain clearly perform better than InfoMap, LPA and FastGreedy on both the high density real-world networks (football, polblogs and polbooks) and the sparse real-world networks (karate, adjnoun and DBLP). Focusing on the three better algorithms, the average NMI and ARI of E-Attractor are slightly better than those of Louvain and Attractor.

Table 7. Computation time of various algorithms on real-world networks (ms).

Network	karate	polbooks	adjnoun	football	polblogs	DBLP
FastGreedy	339	1139	1289	1689	31977	6666447
InfoMap	228	928	941	1021	21756	3766891
Louvain	28	199	207	248	6987	250974
LPA	12	129	133	141	3427	110889
Attractor	241	991	1017	1145	22147	3910124
E-Attractor	292	1035	1264	1406	29520	4359569

Table 7 shows the computation times of the six algorithms on the real-world networks. As shown in **Table 7**, LPA has the smallest computation time, Louvain is second, Attractor, InfoMap and E-Attractor are virtually tied for third, and FastGreedy requires the most computation time. Focusing on E-Attractor, InfoMap and Attractor, we observe that the computation time of Attractor is smaller than that of E-Attractor and larger than that of InfoMap.

6. Conclusion

This paper presents the design of an enhanced distance dynamics model based on Ego-Leader and proposes a corresponding community detection algorithm, E-Attractor. The paper's contributions have two primary aspects. One, to remove the strong dependence of the distance dynamics model on cohesion parameter λ, the Ego-Leader is introduced to replace parameter λ in determining the influence of an exclusive neighbor on distance. Two, based on Ego-Leader, we design an enhanced distance dynamics model with better robustness. Using the new model, we propose a corresponding community detection algorithm, E-Attractor. Extensive experiments have been executed on both synthetic networks and real-world networks. The experimental results show the benefits of our algorithm.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (61573299,61174140, 61472127, 61272395); Social Science Foundation of Hunan Province(16ZDA07); China Postdoctoral Science Foundation (2013M540628,2014T70767); Natural Science Foundation of Hunan Province (14JJ3107); Excellent Youth Scholars Project of Hunan Province (15B087).

References

[1] Fortunato S and Hric D, "Community detection in networks: A user guide," *Physics Reports*, vol.659, no.11, pp.1-44, November, 2016. Article (CrossRef Link).

[2] Papadopoulos S, Kompatsiaris Y, Vakali A and Spyridonos P, "Community detection in social media," *Data Mining and Knowledge Discovery*, vol.24, no.3, pp.515-554, May, 2012. Article (CrossRef Link).

[3] Fortunato S, "Community detection in graphs," *Physics reports*, vol.486, no.3, pp.75-174, February, 2010. Article (CrossRef Link).

[4] Böhm C, Plant C, Shao J and Yang Q, "Clustering by synchronization," in *Proc. of 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp.583-592, July 25-28, 2010. Article (CrossRef Link).

[5] Shao J, Plant C, Yang Q and Bohm C, "Detection of arbitrarily oriented synchronized clusters in high-dimensional data," in *Proc. of 11th International Conference on Data Mining*, pp.607-616, December 11-14, 2011. Article (CrossRef Link).

[6] Xiong Y, Zhu Y, Philip S Y and Jian Pei, "Towards Cohesive Anomaly Mining," in *Proc. of 27th AAAI Conference on Artificial Intelligence*, pp.984-990, July 14-18, 2013. Article (CrossRef Link).

[7] Shao J, Ahmadi Z and Kramer S, "Prototype-based learning on concept-drifting data streams," in *Proc. of 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp.412-421, August 24-27, 2014. Article (CrossRef Link).

[8] Novikov A and Benderskaya E, "Oscillatory Network Based on Kuramoto Model for Image Segmentation," in *Proc. of International Conference on Parallel Computing Technologies*, pp. 210-221, August 21-30, 2015. Article (CrossRef Link).
[9] Hong L, Cai S M, Zhang J and Zhuo Z, "Synchronization-based approach for detecting functional activation of brain," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol.22, no.3, pp. 113-128, August, 2012. Article (CrossRef Link).
[10] Shao J, Han Z, Yang Q and Zhou T, "Community detection based on distance dynamics," in *Proc. of 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.1075-1084, August 10-13, 2015. Article (CrossRef Link).
[11] Wang J, Li M, Wang H and Pan Y, "Identification of essential proteins based on edge clustering coefficient," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol.9, no.4, pp.1070-1080, November, 2012. Article (CrossRef Link).
[12] Radicchi F, Castellano C, Cecconi F and Loreto V, "Defining and identifying communities in networks," in *Proc. of Proceedings of the National Academy of Sciences of the United States of America*, vol.101, no.9, pp.2658-2663, January, 2004. Article (CrossRef Link).
[13] Clauset A, Newman M E and Moore C, "Finding community structure in very large networks," *Physical Review E*, vol.70, no.6 Pt 2, pp.264-277, December, 2004. Article (CrossRef Link).
[14] Blondel V D, Guillaume J L, Lambiotte R and Lefebvre E, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics Theory & Experiment*, vol.2008, no.10, pp.155-168, October, 2008. Article (CrossRef Link).
[15] Jalan S, Singh A, Acharyya S and Kurths J, "Impact of a leader on cluster synchronization," *Physical Review E*, vol.91, no.2, pp.22-34, February, 2015. Article (CrossRef Link).
[16] Goyal A, Bonchi F and Lakshmanan LVS, "Discovering leaders from community actions," in *Proc. of 17th ACM conference on Information and knowledge management*, pp.499-508, October 26-30, 2008. Article (CrossRef Link).
[17] Mehra A, Dixon AL, Brass DJ and Robertson B, "The social network ties of group leaders: Implications for group performance and leader reputation," *Organization science*, vol.17, no.1, pp. 64-79, February, 2006. Article (CrossRef Link).
[18] Wang J, Ma Q and Zeng L, "Observer-based synchronization in fractional-order leader–follower complex networks," *Nonlinear Dynamics*, vol.73, no.2, pp.921-929, March, 2013. Article (CrossRef Link).
[19] Rosvall M and Bergstrom CT, "Maps of random walks on complex networks reveal community structure," in *Proc. of Proceedings of the National Academy of Sciences*, vol.105, no.4, pp.1118-1123, January, 2008. Article (CrossRef Link).
[20] Gower JC, "Measures of similarity, dissimilarity and distance," *Encyclopedia of statistical sciences*, vol.5, no.3, pp.397-405, July, 1985. Article (CrossRef Link).
[21] Mark EJ Newman, "Modularity and community structure in networks," in *Proc. of Proceedings of the National Academy of Sciences*, vol.103, no.23, pp.8577–8582, May, 2006. Article (CrossRef Link).
[22] William M Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol.66, no.336, pp.846–850, April, 1971. Article (CrossRef Link).
[23] Alexander Strehl and Joydeep Ghosh, "Cluster ensembles-a knowledge reuse framework for combining multiple partitions," *The Journal of Machine Learning Research*, vol.3, no.12, pp.583–617, December, 2003. Article (CrossRef Link).
[24] Wang F and Zhang C, "Label propagation through linear neighborhoods," *IEEE Transactions on Knowledge and Data Engineering*, vol.20, no.1, pp.55-67, November, 2008. Article (CrossRef Link).

Cai LiJun received the Ph.D. degree in College of Information Science and Engineering from Hu Nan University in 2007. He is currently a Professor at Hu Nan University. His research interests include bioinformatics, cloud computing, big data scheduling and management.

Zhang Jing received the Ph.D. degree in College of Electrical and Information Engineering from Hu Nan University in 1997. He is currently a Professor at Hu Nan University. His research interests include complex process control and optimization.

Chen Lei received the M.S. degree in College of Information Science and Engineering from Hu Nan University, Changsha, China. He is currently pursuing the Ph.D. degree in the College of Electrical and Information Engineering, Hu Nan University, Changsha, China. His research interests include date mining, cloud computing, big data scheduling and analysis.

He TingQin received the M.S. degree in College of Information Science and Engineering from Hu Nan University, Changsha, China. He is currently pursuing the Ph.D. degree in the College of Information Science and Engineering, Hu Nan University, Changsha, China. His research interests include date mining, pattern recognition.