IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 29, 2018, accepted October 11, 2018, date of publication October 22, 2018,
date of current version November 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2877235

A Modified Distance Dynamics Model for Improvement of Community Detection

TAO MENG[1], LIJUN CAI[1], TINGQIN HE[1], LEI CHEN[2], ZIYUN DENG[3], WEIPING DING[4,5], (Member, IEEE), AND ZEHONG CAO[5]

[1]College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
[2]College of Information and Electrical Engineering, Hunan University of Science and Technology, Xiangtan 411201, China
[3]Department of Economics and Trade, Changsha Commerce and Tourism College, Changsha 410082, China
[4]School of Computer Science and Technology, Nantong University, Nantong 226019, China
[5]Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia

Corresponding author: Lijun Cai (ljcai@hnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61472127 and Grant 61272395, in part by the Natural Science Foundation of Hunan Province under Grant 2017JJ5064, in part by the Social Science Foundation of Hunan Province under Grant 16ZDA07, and in part by the Open Project of State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body under Grant 31715010.

ABSTRACT Community detection is a key technique for identifying the intrinsic community structures of complex networks. The distance dynamics model has been proven effective in finding communities with arbitrary size and shape and identifying outliers. However, to simulate distance dynamics, the model requires manual parameter specification and is sensitive to the cohesion threshold parameter, which is difficult to determine. Furthermore, it has difficulty handling rough outliers and ignores hubs (nodes that bridge communities). In this paper, we propose a robust distance dynamics model, namely, Attractor++, which uses a dynamic membership degree. In Attractor++, the dynamic membership degree is used to determine the influence of exclusive neighbors on the distance instead of setting the cohesion threshold. In addition, considering its inefficiency and low accuracy in handling outliers and identifying hubs, we design an outlier optimization model that is based on triangle adjacency. By using optimization rules, a postprocessing method further judges whether a singleton node should be merged into the same community as its triangles or regarded as a hub or an outlier. Extensive experiments on both real-world and synthetic networks demonstrate that our algorithm more accurately identifies nodes that have special roles (hubs and outliers) and more effectively identifies community structures.

INDEX TERMS Community detection, complex network, distance dynamics model, membership function.

I. INTRODUCTION

Many complex systems in the real world can be viewed as complex networks [1], such as social networks, sensor networks, collaboration networks, biological networks and other types of complex networks. In recent years, the discovery of community structures in complex networks has gradually become a hot research field. Community detection plays an important role in complex network analysis because it provides comprehensive insight into the organizational structure, functional behavior, and evolutionary dynamics of the network [2], [3]. The main objective of community detection is to group similar nodes into the same community while partitioning dissimilar nodes into different communities, where a community can be regarded as a group of nodes with high-density links within the group and relatively low-density links with nodes in external groups [4]. Understanding the community structure of a network is an important problem and is very useful in our lives. The development of algorithms for detecting communities in networks has attracted the interest of physicists, sociologists, and especially computer scientists [5]–[7].

Up to now, many community detection algorithms have been developed to reveal hidden community structures, which mainly include the graph-partitioning method [8], [9], modularity-based method [10], [11], density-based method [12], [13], and dynamic method [14], [15]. Most existing community detection algorithms use a greedy optimization metric to qualify community structure from various points of view and each algorithm has its own advantages and limitations. Apart from the user-defined metric algorithm, how can

we identify the communities in a real-world network in an intuitive way?

Lately, one of the most successful community detection methods, namely, Attractor, which is a distance dynamics model, was proposed by Shao *et al*. [15]. Unlike the traditional algorithms [8], [12], [14], [16], Attractor provides an intuitive way to analyze the community structure of a network. This model views the entire graph as an adaptive global dynamical system and simulates the synchronization dynamics over time. The process of the traditional distance dynamics model involves the following stages: First, each edge is associated with an initial distance. Then, in a sequential process, each distance gradually shrinks or stretches via interaction with its local topological structure. Finally, all distances converge to 0 or 1. As a result, all communities and outliers are naturally obtained by removing the edges with distance that are equal to 1. The traditional model has several attractive benefits, such as intuitive community detection, small community detection, and anomaly detection. However, there are several limitations of the traditional distance dynamics model.

- *Extremely Sensitive Parameter Settings:* In the traditional distance dynamics model, the global cohesion parameter, which is denoted as λ, is used to determine the positive or negative interaction influence on the distances for exclusive neighbors. Typically, a lower value of λ yields larger communities whereas a higher value of λ produces more communities. However, different networks have different local structures and may require different parameter settings. Thus, it is difficult to find a proper value of the cohesion parameter λ for a specified network. In some cases, minor changes to parameter λ may cause great differences in the resulting community structure.
- *Unreasonable Influence From Exclusive Neighbors:* During the local dynamic interaction process, the structures of the communities are constantly changing as new distances converge. In the traditional distance dynamics model, once the underlying influence of exclusive neighbors on the distances has been determined by the cohesion parameter λ, the influence does not change during the entire dynamic interaction process. Even if an exclusive neighbor has a positive influence on the distance at time step 0, it would have a negative influence on the distance at time step t (the exclusive neighbor may have been moving far away from the corresponding node at time step t).
- *Poor Quality of Anomaly Detection:* In the process of synchronization dynamics, the traditional model easily produces many rough outliers, especially in a large-scale, high-density, or noisy network. Many outliers that are identified by the traditional distance dynamics model belong to a community in the ground truth of the network. Consider the typical email-enron network as an example: The network consists of 1133 nodes and 5451 edges. By using the traditional distance dynamics model with parameter $\lambda = 0.5$, we identify 359 classes, namely, 300 outliers and 59 communities. In this model, many other real-world networks have similar scenarios.
- *Unable to Identify the Differences Between Outliers and Hubs:* Actually, Some of the sparsely connected nodes in a network may not be outliers but hubs. In addition to detecting communities and outliers, identifying nodes with special roles, such as hubs, is a challenging task in determining the structure of a complex network, as hubs play important roles in many real complex networks [12]. For instance, hubs in epidemiology networks can be core nodes for spreading diseases; in collaboration networks, hubs can be core nodes for sharing ideas.

To describe the hubs more clearly, let us consider a simple example; see Figure 1. By using the traditional distance dynamics model, we identify 2 communities and 2 outliers. All nodes with the same color belong to the same community and the two red nodes are the outliers. Our method, namely, Attractor++, identifies two communities, namely, 1, 2, 3, 4, 5, 6 and 9, 10, 11, 12, 13, 14, and identifies node 7 as an outlier and node 8 as a hub.

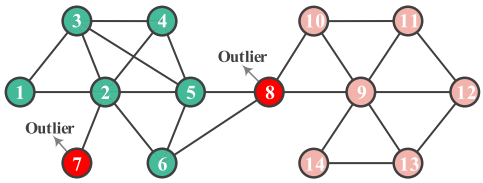


FIGURE 1. Running example.

A robust distance dynamics model should be able to overcome the above limitations. We propose a robust distance dynamics model for community detection. To overcome the parameter-sensitivity problem, the dynamic membership degree is introduced to determine the influence of an exclusive neighbor on the distance. Furthermore, the dynamic influences from exclusive neighbors can also be easily determined by our algorithm. The membership degree is a dynamic function that is based on the characteristics of the communities during the local dynamic interaction process in real time. To overcome the rough-outlier problem, an outlier optimization rule is proposed for further judging whether an outlier should be merged into a community based on the adjacent triangle. To overcome the unidentified-hub problem, another outlier optimization rule is developed for further judging whether an outlier should be as hub based on the connected triangle. We summarize the main contributions of this paper as follows.

- *A Robust Distance Dynamics Model:* Based on a dynamic membership degree, we propose a robust distance dynamics model that has improved robustness. The dynamic membership degree is used to handle the traditional cohesive parameter λ. The dynamic membership degree is a similarity index that is used to measures

the similarity between nodes and communities. The experimental results demonstrate the effectiveness and accuracy of finding communities via the robust distance dynamics model.

- *An Outlier Optimization Model:* To further judge whether each outlier should be merged into the same community as its triangles or classified as a hub, we design two outlier optimization rules that help identify vertices that have special roles (i.e., hubs and outliers) and integrate them into the distance dynamics model.
- *Robust Algorithm (Attractor++):* A robust community detection algorithm, namely, Attractor++, is proposed. It is based on a robust distance dynamics model and an outlier optimization model. Experimental results on artificial and real-world networks demonstrate that Attractor++ is more robust and efficient in overcoming the above limitations than the original algorithm.

The remainder of this paper is organized as follows: Related works are discussed in Section *II*. Section *III* presents our robust model and corresponding community detection algorithm. An extensive experimental evaluation is presented in Section *IV*. Finally, Section *V* presents the conclusions of this paper.

II. RELATED WORKS

Community detection has been studied for decades in many fields. Recently, scholars have proposed many algorithms for detecting community structures in complex networks, particularity in computer science [2]. We review related works, which are organized according to the community detection algorithms, dynamic method and distance dynamics model, and dynamic membership degree.

Community Detection Algorithms: Currently, the most widely used and practical community detection algorithms can be divided into four categories: graph-partitioning algorithms, modularity-based algorithms, density-based-method algorithms and dynamics algorithms. In graph-partitioning algorithms, community detection was first modeled as a graph partitioning problem. Hence, graph-partitioning algorithms [8], [9] are natural choices for community detection. However, these algorithms rely on a prespecified number of communities k, which renders them not highly applicable to real-world networks. Since the community structures are highly complex, it is often expensive or impossible to obtain the number of communities in many real-world networks. For modularity-based algorithms, many researchers devoted their efforts to improving the effectiveness of community detection. One typical method is to optimize the modularity measure [10], which is widely used to evaluate the community structure of a network from a global perspective. Unfortunately, although modularity-based algorithms are effective in many applications [10], [11], they are difficult to apply to large-scale networks due to their high time complexity (which is called the resolution limit problem). Due to the resolution limit of modularity-based algorithms, density-based algorithms have attracted wide research interest [12], [17]. One of the most successful density-based algorithms, namely, SCAN [12], not only detects meaningful communities but also identifies hubs and outliers. To overcome the problems of parameter sensitivity and exhaustive similarity evaluations in SCAN, two parameter-free methods, namely, SHRINK [18] and SkeletonClu [17], have been proposed and SCAN++ [13] and pSCAN [19] have been proposed to reduce the time complexity.

Dynamic Method and Distance Dynamic Model: Dynamic algorithms that support additional community semantics are another research area. Dynamic-process-based methods are important in the field of complex network community discovery. Typical dynamic methods include label propagation [16], random walk [20], and distance dynamics [15]. Owing to the simplicity of its procedure, the label propagation method can detect communities in almost linear time; however, it has poor stability due to the randomness in the label propagation process [2]. Random-walk-based methods are routinely used for community detection from the global perspective [20], [21]. However, the quality of the detected communities heavily depends on the choice of the seed node. Recently, inspired by synchronization clustering [22], Shao *et al*. [15] consider the problem of community detection from a new point of view: distance dynamics. Unfortunately, this method has several problems, which were analyzed in Section *I*. To overcome the sensitivity of parameter λ, E-Attractor [18] was recently proposed. It improves the stability of Attractor by employing Ego-Leader to replace cohesion parameter λ in the dynamic interaction process. By using Ego-Leader, the underlying influence of exclusive neighbors can be determined by identifying the top-k neighbors. However, it still has difficulty determining the globally optimal value of k and lacks an automated way to find a satisfactory value of k. Moreover, clustering based on the global parameter settings cannot always describe the intrinsic community structure accurately and easily produces many rough outliers. In addition, Fan *et al*. [23] proposed a semisupervised community detection method that integrates the prior information into the distance dynamics model to improve the accuracy of community detection. Although this approach is novel, it does not consider these problems. To the best of our knowledge, ours is the first work to solve these problems systematically.

Dynamic Membership Degree: The dynamic membership degree is essentially a dynamic membership function. Dynamic membership functions have been extensively studied [24], [25] and are widely used in fuzzy systems to describe the system dynamics [26]. Nepusz *et al*. [27] define a numerical membership degree and develop an algorithm for determining the optimal membership degree that is able to identify outlier vertices that do not belong to any of the communities, which are called bridge vertices, and quantify the centrality of a vertex with respect to its dominant community. Kundu *et al*. [28] proposed a community detection algorithm that identifies fuzzy-rough communities in

which a single node can belong to many groups with various membership degrees. The method performs well when the network contains overlapping communities. Recently, Luo *et al*. [29] used various dynamic membership functions to describe the dynamics of the process of community formation and achieved satisfactory result. Therefore, evidence increasingly supports that dynamic membership functions have substantial advantages in describing system dynamics.

Motivated by the advantageous properties of dynamic membership functions, we replace cohesion parameter λ by a dynamic membership degree to simulate the distance dynamics more accurately. In this paper, we integrate the dynamic membership degree into the distance dynamics model and propose a robust distance dynamics model that has no parameters. We combine the original network topology with the membership degree to modify the distance model, which can substantially shorten the time step to accelerate the convergence of the distance between nodes and improve the accuracy of our algorithm.

III. PROPOSED METHOD:ATTRACTOR++

A. PRELIMINARIES

Before introducing our method, we present the basic notions and related definitions that we use throughout the paper. In this paper, we focus on an undirected graph $G = (V, E, W)$, where V, E and W denote the node set, edge set, and edge weight set, respectively. The distance between two nodes depends on their shared neighbors. Thus, prior to computing distances, the structural neighbors of a node are defined. The structural neighbors of a node are its adjacent nodes and the node itself.

Definition 1 (Neighbors of Node u*):* In an undirected graph $G = (V, E, W)$, the neighbors of node u, which are denoted by $N(u)$, are defined as follows:

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

The distance between adjacent nodes is computed according to the common nodes in the structural neighborhoods. This measurement is called the Jaccard distance and is defined as follows:

Definition 2 (Jaccard Distance): In an unweighted undirected graph $G = (V, E)$, the Jaccard distance 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, $|*|$ denotes the number of nodes in set $*$ and $N(*)$ denotes the neighbors of node $*$. The Jaccard distance is a score that varies from 0 to 1 and indicates the scale of the matching degree of the common neighbors. When two adjacent nodes share few common neighbors, their Jaccard distance is large.

For a weighted undirected graph, because each edge has a different weight, the equation for the Jaccard distance is different:

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

B. DYNAMIC MEMBERSHIP DEGREE CONSTRUCTION

To systematically address the limitations of the traditional distance dynamics model and enable efficient community detection, we propose a new metric, namely, the dynamic membership degree, for measuring the similarity between an exclusive neighbor node and core nodes and border nodes. Specifically, if an exclusive neighbor node has a stronger membership degree to the core nodes than to the border nodes, the exclusive neighbor node will have a positive influence on the distance. Moreover, because the node set of core nodes and border nodes will change over time, the membership degree is dynamic. The key to the dynamic membership degree is that each community in a graph consists of a set of core nodes and the border nodes that are associated with these core nodes. Thus, the dynamic membership degree can replace traditional cohesion parameter λ to determine the influence of an exclusive neighbor on the distance. To compute the membership degree of an exclusive neighbor node, we define the core nodes of the community that is associated with a node as follows:

Definition 3 (Core Nodes): For any arbitrary node u, the core nodes $C(u)$ are defined as follows: First, the node u and its neighbors are considered core nodes if they have nonzero similarity degree with node u. Second, for a node that is not a neighbor of node u to become a core node, it must have a distance of 0 from node u or any other core node.

These core nodes are more likely to cluster with node u. The core membership degree of exclusive neighbor node x to the community that is associated with node u is computed as follows:

$$CM(x, u) = \frac{|T(x) \cap C(u)|}{|T(x)|}, \tag{4}$$

where $T(x)$ is the set of neighbors of the exclusive neighbor node x and not include the node x and $C(u)$ is the set of core nodes that are associated with node u.

Definition 4 (Border Nodes): For any arbitrary node u and exclusive neighbor node x such that $u \neq x$, the border nodes $B(u)$ are define as follows: First, node x and its neighbors are considered border nodes if they are not core nodes that are associated with node u. Second, for a node that is not a core node that is associated with node u to become a border node, it must have a distance of 0 from node x or any other border node.

The border nodes are those nodes that have a small probability of clustering with node u. The border membership degree of exclusive neighbor node x is computed as follows:

$$BM(x, u) = \frac{|T(x) \cap B(u)|}{|T(x)|}, \tag{5}$$

where $T(x)$ is the set of neighbors of exclusive neighbor node x and does not include the node x and $B(u)$ is the set of border nodes that are associated with node u.

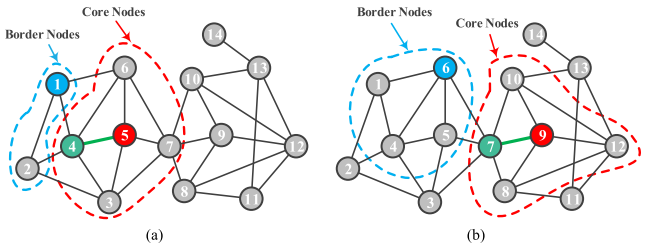


FIGURE 2. Illustration of the dynamic membership degree.

After computing both the core membership degree and the border membership degree, we can easily determine the positive or negative influence of exclusive neighbors on the distances. To illustrate the dynamic membership degree more clearly, Figure 2 shows two examples.

Consider the graph G in Figure 2(a). Node 4 and node 5 are indirectly connected and node 1 is an exclusive neighbor of node 5 on edge $e(4, 5)$. The core nodes (circled by a red dotted line) of node 5 are nodes 3, 4, 5, 6 and 7, according to Definition 3. The border nodes (circled by a blue dotted line) of node 5 are nodes 1 and 2, according to Definition 4. Since $CM(1, 5) = 2/3$ and $BM(1, 5) = 1/3$, node 1 is more similar to the core nodes than to the border nodes. Therefore, exclusive neighbor node 1 will have a positive influence and reduce the distance on edge $e(4, 5)$. For comparison, consider the graph G in Figure 2(b). Node 7 and node 9 are indirectly connected nodes and node 6 is an exclusive neighbor of node 9 on edge $e(7, 9)$. The core nodes (circled by a red dotted line) of node 9 are nodes 7, 8, 9, 10 and 12, according to Definition 3. The border nodes (circled by a blue dotted line) of node 9 are nodes 1, 4, 5 and 6, according to Definition 4. Since $CM(6, 9) = 1/4$ and $BM(6, 9) = 3/4$, node 6 is more similar to the border nodes than to the core nodes. Therefore, exclusive neighbor node 6 will have a negative influence and increase the distance on edge $e(7, 9)$.

Furthermore, as time evolves, the node sets $C(u)$ and $B(u)$ change as distances converge (to 0 or 1). Thus, $CM(x, u)$ and $BM(x, u)$ are dynamic membership degree functions. For instance, after many time steps, the distance on edge $e(4, 1)$, $e(4, 2)$, $e(3, 2)$ or $e(6, 1)$ may converge to 0 in Figure 2(a). As a result, node 1 or node 2 may join the core nodes of node 5 and may be removed from the border nodes of node 1.

C. ROBUST DISTANCE DYNAMICS MODEL

In the traditional distance dynamics model, three interaction patterns (DI, CI, and EI) are designed for simulating the distance dynamics. However, a naive cohesion parameter λ is used to determine the positive or negative influence of an exclusive neighbor on the distance in interaction pattern EI. A poor choice of parameter λ may lead to bad results. Hence, we use notions of core nodes and border nodes and the properties of the dynamic membership degree that are discussed above to improve the EI pattern. Because the traditional DI and CI patterns do not use cohesion parameter λ, these two patterns are unchanged in our robust model.

Robust Pattern 1: In the first interaction pattern (Figure 3(b)), the distance $d(u, v)$ is influenced by two directly linked nodes: u and v. In this scenario, one node attracts another to move toward itself, thereby leading to a decrease in the distance $d(u, v)$. Formally, the influence from the directly linked nodes, which is denoted as DI, is defined as follows:

$$DI = -\left(\frac{f\left(1 - d\left(u, v\right)\right)}{deg\left(u\right)} + \frac{f\left(1 - d\left(u, v\right)\right)}{deg\left(v\right)}\right). \quad (6)$$

In the pattern DI, $f(*)$ is a coupling function and $sin(*)$ is used in Attractor, $deg(*)$ indicates the degree of node $*$. The DI is score of varying from -1 to 0 that indicates the degree of influence on the distance from direct linked nodes. When two direct linked nodes are more similar, the higher influence between each other they will have, and vice versa.

Robust Pattern 2: Influence from common neighbors. In the second interaction pattern (Figure 3(c)), the distance $d(u, v)$ is influenced by the common neighbors. The common neighbors have links with both nodes u and v and are denoted as $CN = (N(u) - u) \cap (N(v) - v)$. In this scenario, each common neighbor attracts both node u and node v to move toward itself, thereby resulting in a decrease in the distance $d(u, v)$. Formally, the influence of the common nodes, which is denoted as CI, is defined as follows:

$$CI = -\sum_{x \in CN}\left(\frac{f\left(1 - d\left(x, u\right)\right) \cdot \left(1 - d\left(x, v\right)\right)}{deg\left(u\right)}\right) - \sum_{x \in CN}\left(\frac{f\left(1 - d\left(x, v\right)\right) \cdot \left(1 - d\left(x, u\right)\right)}{deg\left(v\right)}\right). \quad (7)$$

In the pattern CI, for any common neighbor x, the CI is score of varying from -1 to 0 that indicates the degree of

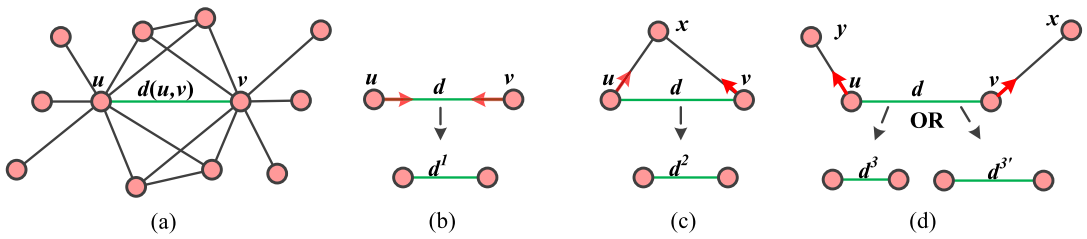


FIGURE 3. Three distinct interaction patterns.

influence on the distance from common neighbor x. When the common neighbor x share many members between node u and node v, the influence becomes large, and vice versa

Robust Pattern 3: Influence from exclusive neighbors. Unlike in the *DI* pattern and *CI* patterns, where directly linked nodes or common nodes can only exert a positive influence on the distance, in the *EI* pattern exclusive neighbors can exert a positive or negative influence on the distance (Figure 3(d)); otherwise, all distances would converge to 0. To avoid this problem, instead of using a cohesion parameter λ to determine the underlying influence, we focus on the dynamic membership degree. For edge $e(u, v)$, node x is an exclusive neighbor of node u and we calculate the values of $CM(x, u)$ and $BM(x, u)$. If $CM(x, u) \geq BM(x, u)$, the relationship between x and u is very close and results in the decrease of the distance $d(u, v)$. Similarly, if $CM(x, u) < BM(x, u)$, the relationship between x and u is not close and results in the increase of the distance $d(u, v)$. To determine the positive or negative influence of an exclusive neighbor on distance $d(u, v)$, the dynamic membership degree is used, which is defined as:

$$\sigma(x, u) = \begin{cases} (1 - d(x, v)), & (CM(x, u) - BM(x, u)) \geq 0, \\ (d(x, v) - 1), & otherwise. \end{cases} \tag{8}$$

In equation 8, $CM(x, u)$ and $BM(x, u)$ are localized similarity indices for assessing the similarity between the exclusive neighbor and the core nodes and border nodes, respectively. The function $\sigma(x, u)$ not only characterizes the degree of influence of the exclusive neighbor x on the distance but also indicates the direction (positive or negative) of this influence. Based on the function $\sigma(x, u)$, the robust pattern of influence by exclusive neighbors, which is called *REI*, is defined as follows:

$$REI = -\sum_{x \in EN(u)} \left(\frac{f(1 - d(x, u)) \cdot \sigma(x, u)}{deg(u)} \right) - \sum_{y \in EN(v)} \left(\frac{f(1 - d(y, v)) \cdot \sigma(y, v)}{deg(v)} \right), \tag{9}$$

where $EN(u)$ and $EN(v)$ are node sets of exclusive neighbors of nodes u and v, respectively and are expressed as $EN(u) = N(u) - (N(u) \cap N(v))$ and $EN(v) = N(v) - (N(u) \cap N(v))$.

As a result, we obtain the robust distance dynamics model by considering three interaction patterns together. The distance dynamics $d(u, v, t + 1)$ between nodes u and v over time is defined as follows:

$$d(u, v, t + 1) = d(u, v, t) + DI(u, v, t) + CI(u, v, t) + REI(u, v, t), \tag{10}$$

where $d(u, v, t + 1)$ is the new distance at time step $t + 1$ and $DI(u, v, t)$, $CI(u, v, t)$ and $REI(u, v, t)$ indicate the influences of directly connect end nodes, common neighbors, and exclusive neighbors, respectively, on the distance $d(u, v, t)$ at time step t.

Finally, as time evolves, all distances will converge, and all communities and outliers can be easily identified by removing the edges with the distances equivalent to 1.

D. OUTLIER OPTIMIZATION MODEL

In a network, an outlier has few links in its neighbor set [13]. Therefore, we try to exploit the structure connectivity of neighbors to further optimize the accuracy of outliers, which are identified via the distance dynamics model. The concept of structure connectivity has been widely used in community detection [12]. Contrary to traditional methods [12], [13], we take the triangles instead of the edges as the basic indicator of a strong relation in the graph. The main reason we choose the triangle structure connectivity is that real-world graphs such as social networks contains more triangles than random graphs and have many triangles in a community [30]. Moreover, triangles are known as fundamental building blocks of a network. In a network, a triangle implies a strong tie among three nodes or the existence of a common node between other two nodes. In this section, we introduce two outlier optimization rules: triangle adjacency and triangle connectivity. Based on these optimization rules, we propose an outlier postprocessing method. If the triangles of an outlier satisfy triangle adjacency and all adjacent triangles belong to the same community, this outlier should cluster with its triangles. For that, we define triangle adjacency.

Definition 6 (Triangle Adjacency): Two triangles, namely, Δ_o and Δ_c, in $G = (V, E)$ are adjacent if and only if Δ_o and Δ_c share a common edge, which is denoted by $\Delta_o \cap \Delta_c = e(x, y) \in E(G)$.

For the graph G in Figure 4(a), $\Delta_{5,14,15}$ and $\Delta_{5,8,15}$ are adjacent as they share a common edge: $e(5, 15)$. Based on triangle adjacency, we propose the first optimization rule.

Optimization Rule 1: Given the set C of clusters in a graph G, a vertex u that is not in any cluster in C is not an outlier vertex if its triangles are only adjacent to other triangles that are in same community, which is denoted as C_i. In this case, the outlier u belongs to community $C_i (u \in C_i)$.

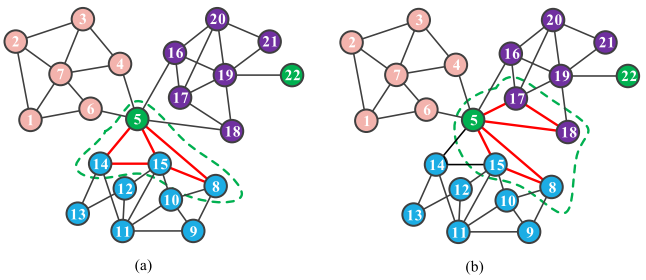


FIGURE 4. Illustration of the optimization model.

To describe the rule more clearly, we consider a simple example. In Figure 4(a), a simple social network is illustrated. By using the robust distance dynamics model, we find 3 classes and 2 outliers, where all nodes with same color belong to the same community and the two green nodes are the outliers. The triangles of outlier 5, namely, $\Delta_{5,14,15}$

and $\Delta_{5,8,15}$, are adjacent to triangles $\Delta_{11,14,15}$ and $\Delta_{8,10,15}$, respectively. Furthermore, triangles $\Delta_{11,14,15}$ and $\Delta_{8,10,15}$ both belong to the blue community. Hence, optimization rule 1 is satisfied. Therefore, node 5 is not an outlier and should be merged into the blue community. Unlike node 5, node 22 doesn't satisfy optimization rule 1. Therefore, node 22 is an outlier and should not merged into the purple community.

In many real-world networks, there are typically hub nodes [12] that bridge many clusters but don't belong to any cluster. Outliers are the nodes that are neither clusters nor hubs. Each outlier is only weakly associated with a cluster. If the triangles of an outlier are connected and belong to different communities, this node is not an outlier; it is a hub. For that, we define triangle connectivity.

Definition 10 (Triangle Connectivity): Two triangles, namely, Δ_o and Δ_c, in $G = (V, E)$ are connected if and only if Δ_o and Δ_c share only one common node, which is denoted by $\Delta_o \cap \Delta_c = u \in E(V)$.

For the graph G in Figure 4(b), $\Delta_{5,8,15}$ and $\Delta_{5,17,18}$ are connected as they share a common node: node 5. Based on the triangle connectivity, we propose the second optimization rule.

Optimization Rule 2: Given the set C of clusters in a graph G, a vertex u that is not in any cluster in C is not an outlier vertex if its triangles are connected to other triangles that are in different communities, which are denoted as C_i and C_j. In this case, node u is not an outlier, but a hub.

To describe the rule more clearly, let us consider a simple example. In Figure 4(b), a simple social network is illustrated. By using the robust distance dynamics model, we identify 3 classes and 2 outliers, where all nodes with the same color belong to the same community and the two green nodes are the outliers. Triangles $\Delta_{5,17,18}$ and $\Delta_{5,8,15}$ of outlier 5 are connected to triangles $\Delta_{17,19,20}$ and $\Delta_{10,11,15}$, respectively. However, triangles $\Delta_{17,19,20}$ and $\Delta_{10,11,15}$ belong to the purple community and blue community, respectively. Hence, optimization rule 2 is satisfied. Therefore, node 5 is not an outlier; it is a hub. Unlike node 5, node 22 does not satisfy optimization rule 2 and should be identified as an outlier. Since outliers have little or no influence on the clustering of a network, they should be isolated as noises.

Based on the two optimization rules, we propose an outlier postprocessing algorithm for further optimizing the outliers that were identified by the distance dynamics model, to enhance the accuracy of outlier identification. Before we describe the postprocessing algorithm, we make three important remarks: First, when an outlier is a leaf node, we do not change it. Second, when neighbors of an outlier are also outliers, they will be excluded from the neighbor set and not considered in the two optimization rules. Finally, in this paper, we consider a partition that has less than two nodes to be an outlier.

The outlier postprocessing algorithm is presented as Algorithm 1. First, the postprocessing method checks if the singleton node satisfies the triangle adjacency condition in Definition 6 (line 8). If the singleton node satisfies triangle adjacency, the method merges the singleton node into the community via optimization rule 1 (line 9). After that, the postprocessing method classifies the singleton nodes that do not belong to any community as either hubs or outliers. If a singleton node satisfies the definition of triangle connectivity, it is regarded as a hub according to optimization rule 2 (line 11); otherwise, it is regarded as an outlier (line 13).

Algorithm 1 : Outlier Postprocessing

1: **Input:** Rough communities C_R, and outliers O_R;
2: **Output:** Final communities C, hubs H, and outliers O;
3: **Procedure:** Outlier_Optimization(C_R, O_R);
4: // Initialization.
5: Set $C = C_R$, $H = \emptyset$, $O = \emptyset$;
6: // Handling Outliers.
7: **for** each outlier $o \in O_R$ **do**
8: **if** o satisfies optimization rule 1 **then**
9: merge o in to the C with its adjacent triangles;
10: **else if** o satisfies optimization rule 2 **then**
11: label o as a hub and add it to node set H;
12: **else**
13: label o as an outlier and add it to node set O;
14: **end if**
15: **end for**
16: **Return**: C, H, O;

E. THE ATTRACTOR++ ALGORITHM

In this section, we discuss the main components of our algorithm. The proposed algorithm, namely, Attractor++, consists of three stages: the distance initialization stage, the dynamic clustering stage and the cluster refinement stage. The output of the dynamic clustering method is the input of the cluster refinement method. In the dynamic clustering stage, Attractor++ roughly clusters the specified graph. At this stage, the communities and outliers have been roughly identified. After identifying the candidate clusters and outliers, Attractor++ refines the clusters by isolating hubs and outliers in the cluster refinement stage.

The pseudocode of our proposed method, namely, Attractor++, is given in Algorithm 2. First, Attractor++ runs the distance initialization stage (lines 5-8). At the initial time (t = 0), without any interaction, all the edges are associated with an initial distance via the Jaccard-distance function, which is expressed in Eq. 2 and Eq. 3 (line 7). Then, the dynamic clustering stage begins (line 10-31), which relies on the three proposed interaction patterns (*DI*, *CI*, and *REI*) and the distances among nodes change gradually as time evolves (lines 11-24). Because the dynamic membership degree is used to determine the underlying influences, nodes that share the same community tend to synchronize and the distances between these nodes decrease. By contrast, nodes that are in different communities will separate and the distances between these nodes will increase. As a result, all distances will converge to either 0 or 1 and the communities and

Algorithm 2 : Attractor++

1: **Input:** An undirected Graph $G = (V, E, W)$;
2: **Output:** Final communities C, hubs H, and outliers O;
3: **Procedure:** Attractor++(G);
4: // Stage 1: Initialization Distance
5: Set $C_R = \emptyset$, $O_R = \emptyset$, $C = \emptyset$, $H = \emptyset$, $O = \emptyset$;
6: **for** each edge $e = (u, v) \in E$ **do**
7: compute the distance $d(u, v, t_0)$ via Eq.(2) or Eq.(3);
8: **end for**
9: // Stage 2: Dynamic Clustering
10: **while** any edge has not converged to 0 or 1 **do**
11: **for** each edge $e = (u, v) \in E$ **do**
12: **if** $0 < d(u, v, t_i) < 1$ **then**
13: compute $DI(u, v, t_i)$ via Eq.(6);
14: compute $CI(u, v, t_i)$ via Eq.(7);
15: compute $REI(u, v, t_i)$ via Eq.(9);
16: update distance $d(u, v, t_{i+1})$ via Eq.(10);
17: **end if**
18: **if** $d(u, v, t_{i+1}) \leq 0$ **then**
19: $d(u, v, t_{i+1}) = 0$;
20: **end if**
21: **if** $d(u, v, t_{i+1}) \geq 1$ **then**
22: $d(u, v, t_{i+1}) = 1$;
23: **end if**
24: **end for**
25: **end while**
26: **for** each edge $e = (u, v) \in E$ **do**
27: **if** $d(u, v, t_{i+1}) = 1$ **then**
28: remove edge e from graph G;
29: **end if**
30: **end for**
31: the communities C_R and outliers O_R are roughly identified.
32: // Stage 3: Cluster Refinement
33: use the algorithm 1 Outlier_Optimization(C_R, O_R) to obtain C, H, and O.
34: **Return**: C, H, O;

outliers will be roughly identified by removing all edges with distances of 1 (line 26-30). Finally, the cluster refinement process is executed (line 33). After classifying all singleton nodes as communities, hubs, and outliers, Attractor++ terminates the community detection procedure.

F. COMPLEXITY ANALYSIS

In this section, we analyze the computational complexity of algorithm Attractor++. Given a graph with $|V|$ nodes and $|E|$ edges, Attractor++ finds all communities, hubs and outliers without any parameter settings. For the time complexity analysis, Attractor is divided into three parts: initialization, dynamic clustering and cluster refinement. First, each edge is associated with an initial distance; thus, the computation time is $O(|E|)$. After that, for dynamic clustering, Attractor++ must compute the corresponding core membership degrees and border membership degrees for exclusive neighbors at each time step. Thus, time complexity of this process is $O(T * K * |E|)$, where T is the number of time steps and K is the average number of exclusive neighbors of two linked nodes. Finally, the outliers must be further optimized. The time complexity is $O(D * |O|)$, where D is the average degree of the graph and $|O|$ is the number of outliers that are identified in the dynamic clustering phase. In total, the time complexity of Attractor++ is $O(|E| + T * K * |E| + D * |O|)$.

IV. EXPERIMENTS

In this section, we preform extensive experiments to evaluate the effectiveness and efficiency of the proposed algorithm using a variety of synthetic and real-world networks.

A. EXPERIMENTAL SETUP

Baseline: To evaluate the performance of Attractor++, we compare it to four representative community detection algorithms. All comparison algorithms are listed in Table 1, of which the Louvain algorithm has been recognized as a state-of-the-art community detection algorithm, the LPA algorithm shows linear time complexity for community detection, and the E-Attractor algorithm is a state-of-the-art algorithm that was extended from Attractor and provides a parameter-insensitive distance dynamics model that is based on Ego-Leader. In addition, the Attractor algorithm, as the native algorithm that is based on the traditional distance dynamics model, is indispensable. For all community detection algorithms, unless otherwise stated, the recommended default parameter values are used to obtain the best experimental results.

TABLE 1. Comparison algorithms.

Algorithm	Type	Implementation
Louvain [11]	modularity based algorithm	Python
LPA [16]	dynamic algorithm	Python
Attractor [15]	dynamic algorithm	Python
E-Attractor [31]	dynamic algorithm	Python
Attractor++	dynamic algorithm	Python

Experimental Platform: To simulate the performances of all algorithms on both real and synthetic graphs, we rented a high-performance server (IBM x3650 m4) from National Super Computing Center of Changsha, which is located in Hunan province, China. The server is comprised of one CPU with 8 cores (Intel Xeon Processor E5-2603) and 16 GB main memory. All algorithms are run on the high-performance server using the Windows server 2012 operating system. The Attractor, E-Attractor and Attractor++ algorithms are implemented in Python. For the other two algorithms, we have downloaded the Python implementations from the official websites of the corresponding authors.

Evaluate Metrics: To extensively compare the community detection algorithms in terms of effectiveness, we adopt the following three quality measures:

NMI: The first metric is the normalized mutual information (NMI [32]), which is based on information theory and compares the similarity between the memberships of two communities. It has been most widely used to measure the quality of a community when the ground truth is known. The NMI provides a real number that ranges from 0 to 1 via normalization. If the detected communities are completely independent of the real communities, then NMI $= 0$; if the detected communities are identical to the real communities, then NMI $= 1$.

F-measure: We also use F-measure [33] to quantify the performances of the identified communities. F-measure is a commonly used criterion for community detection algorithms when the community ground truth is known. F-measure provides a real number that is between zero and one and combines recall and precision. A poorly performing community detection algorithm should be associated with a low F-measure. The higher the F-measure value, the better the algorithm performs.

ARI: The Adjusted Rand Index (ARI [34]) is selected as the third metric for all algorithms. ARI measures the similarity between two clustering results (the agreement on whether to put two nodes in the same cluster or in different clusters). ARI has a value that is between 0 and 1, where 1 indicates that the two clustering results are completely same. If the detected communities are poor, then ARI $= 0$.

B. SYNTHETIC NETWORKS

1) NETWORK GENERATION

To evaluate the performance and the sensitivity to community-structure of the selected algorithms, we investigated the results on synthetic networks that were generated by the Lancichinetti Fortunato Radicchi (LFR) benchmark.

The network generation model, namely, LFR(N, C, k, $k_{\max}$, μ, ...), has five important parameters: N is the number of nodes in the network, C is the number of communities, k is the average degree of the nodes, $k_{\max}$ is the maximum degree of the nodes, and μ is the mixing parameter, which indicates the proportion of a node's neighbors that reside in other communities. Typically, the larger the mixing parameter of a network is, the more difficult it is to identify the intrinsic communities. Furthermore, the average degree and community size follow power-law distributions. By varying the parameters of the LFR benchmark, we can analyze the performances of the algorithms in detail. In these experiments, we generate eight synthetic networks with ground-truth information. The values of the parameters for the generated networks are listed in Table 2.

To make the synthetic networks more consistent with the real-world networks, we generate eight networks with various numbers of communities. By setting parameters k, $k_{\max}$ and μ, we ensure that all synthetic networks have different average degrees, maximum degrees of nodes and noise edges in each community.

TABLE 2. Synthetic networks and parameters for the LFR benchmark.

Networks	N	C	k	$k_{\max}$	μ	Edges
LFR1	200	4	15	18	0.25	3208
LFR2	800	8	15	18	0.25	12453
LFR3	2000	20	20	25	0.2	35268
LFR4	4000	50	20	25	0.2	79011
LFR5	10000	80	10	12	0.15	107056
LFR6	20000	120	10	12	0.15	216717
LFR7	40000	180	12	15	0.10	453054
LFR8	60000	250	12	15	0.10	674128

2) COMMUNITY DETECTION PERFORMANCE

We evaluate the community detection performances of various algorithms on LFR synthetic networks. Then, the average values of NMI, F-measure, ARI and running time are calculated. The experimental results are shown in Figure 5.

Figure 5(a) displays the MNI results of various algorithms on LFR synthetic networks, from which we make the following observations: (1) The five community detection algorithms yield satisfactory results and the average value of NMI exceeds 0.6. (2) Comparing the five algorithms, we find that Attractor++ and E-Attractor offer better efficiency and stability, followed by Attractor and Louvain; the LPA algorithm performs worst. (3) Focusing on the Attractor, E-Attractor and Attractor++ algorithms, we find that Attractor++ and E-Attractor are more stable than Attractor on most LFR networks and E-Attractor has very similar performance to Attractor++.

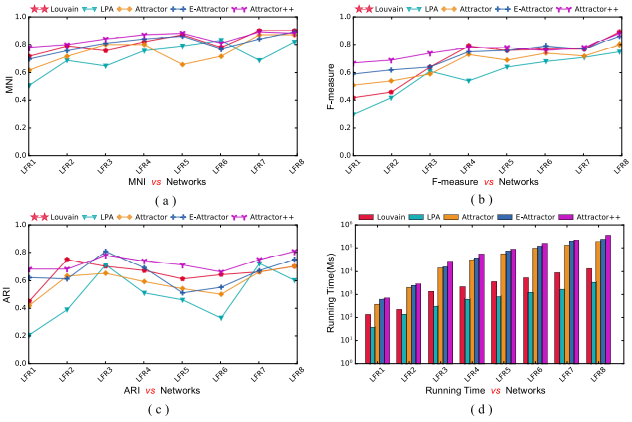


FIGURE 5. Community detection performances of various algorithms on LFR networks.

Figure 5(b) displays the F-measure of various algorithms on LFR synthetic networks, from which we make the following observations: (1) On the high-noise networks (LFR1 LFR4, mixing parameter $\mu \geq 0.2$), the F-measure fluctuation is substantial for all five algorithms, of which Attractor++ and E-Attractor perform best, followed by Attractor and Louvain, and LPA performs worst. (2) On the low-noise networks (LFR5 LFR8, mixing parameter μ<0.2), the performances of five algorithms are similar.

Figure 5(c) displays the ARI values of the five algorithms on LFR synthetic networks, from which we make the following observations: (1) For the ARI, the differences among the five algorithms are substantial and the performances of the algorithms are unstable among the LFR networks. (2) Comparing the five algorithms, we find that Attractor++ offers higher efficiency and stability.

Figure 5(d) displays the running times of the five algorithms on the LFR synthetic networks, from which we make the following observations: (1) Comparing the LPA and Louvain algorithms, we observe that the running time of LPA is 3∼10 times shorter than that of Louvain on average and a few orders of magnitude shorter than that of Attractor. (2) The running times of the Attractor, E-Attractor and Attractor++ algorithms are very similar. Attractor is slightly faster than E-Attractor and E-Attractor is slightly faster than Attractor++.

In summary, for synthetic networks, Attractor++ performs well in identifying ground-truth communities and is more robust to noise than the other algorithms.

3) OUTLIER OPTIMIZATION PERFORMANCE

In the following, we compare the clustering accuracies of Attractor++ and Attractor on various synthetic networks.

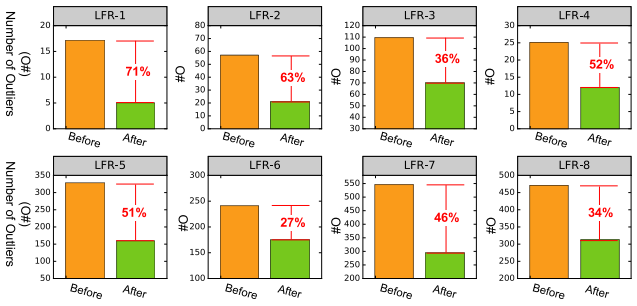


FIGURE 6. Outlier optimization performances on LFR networks.

Figure 6 presents the outlier optimization results on the eight LFR networks. In Figure 6, the "before" column lists the number of outliers (#*O*) that are identified by Attractor without the outlier postprocessing, the "after" column lists the number of outliers (#*O*) that are detected by Attractor++ with the outlier postprocessing, and the red number indicates the percentage reduction of #*O*. As shown in Figure 6, Attractor (the distance dynamics model) easily finds many outliers: the numbers of identified outliers exceed 450 on the LFR-7 and LFR-8 networks. By using the outlier postprocessing, the number of identified outliers can be substantially reduced and the accuracy of outlier identification enhanced. For example, on the four high-density and noisy networks (LFR-1 LFR-4, parameter $\mu \geq 0.2$), the outlier optimization percentages (reducing #*O*) are very large: 71%, 63%, 36% and 52% respectively. On the four low-density and low-noise networks (LFR-5 LFR-8), the outlier optimization percentages are slightly lower: 51%, 27%, 46%, and 34%, respectively. Considering all eight LFR networks, we find that the distance dynamics model faces the drawback of easily producing many rough outliers and the outlier optimization step is highly necessary for the Attractor++ algorithm. Moreover, according to Figure 6, our proposed outlier postprocessing has a very good performance, which further demonstrates the effectiveness of outlier postprocessing.

C. REAL-WORLD NETWORKS

1) NETWORK DESCRIPTION

To evaluate the performance and efficiency, it is necessary to conduct experiments on real-world networks. We compare the performances of algorithms in terms of the accuracy and speed on networks with accurate ground-truth communities.

Six commonly used real-world networks are considered in the experiments; the characteristics of the networks are listed Table 3. These networks are selected because they are very well-known and contain the real structures of communities, which can be used to evaluate the results of each algorithm with a desirable accuracy. Moreover, the six selected real-world networks vary in terms of graph density: polbooks, football and polblogs are dense networks, whereas karate, adjnoun and DBLP are sparse networks. All selected real-world networks are publicly available from the UCI network data repository (https://networkdata.ics.uci.edu/index.php) and the Stanford large network dataset collection (http://snap.stanford.edu/data/).

TABLE 3. The characteristic of commonly used real-world networks.

Networks	Node	Edge	Average degree	Communities
karate	34	78	4.6	2
polbooks	105	441	8.4	3
adjnoun	112	425	7.6	2
football	115	613	10.7	12
polblogs	1490	19090	22.4	2
DBLP	317080	1049866	6.6	13477

2) COMMUNITY DETECTION PERFORMANCE

We evaluate the community detection performances of various algorithms on real-world networks. Then, the average values of NMI, F-measure, ARI and running time are calculated. The experimental results are shown in Figure 7.

Figure 7(a) displays the NMI results of the algorithms on real-world networks, from which we make the following observations: (1) In terms of NMI, the five algorithms yield different results. Of particular interest, on football and polblogs, which are high-density networks, Attractor++ obtains the best NMI, followed by E-Attractor and Attractor, and Louvain and LPA perform worst. (2) Focusing on the Attractor, E-Attractor and Attractor++ algorithms, we find that Attractor++ is more stable than Attractor and E-Attractor on most real-world networks and E-Attractor is more stable than Attractor.

Figure 7(b) displays the F-measure of the algorithms on real-world networks, from which we make the following

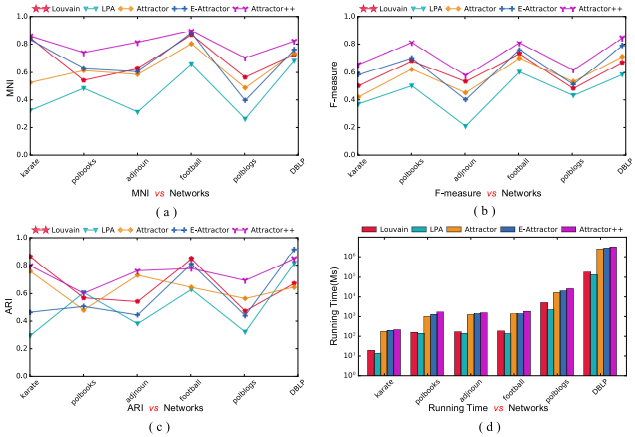


FIGURE 7. Community detection performances of various algorithms on real-world networks.

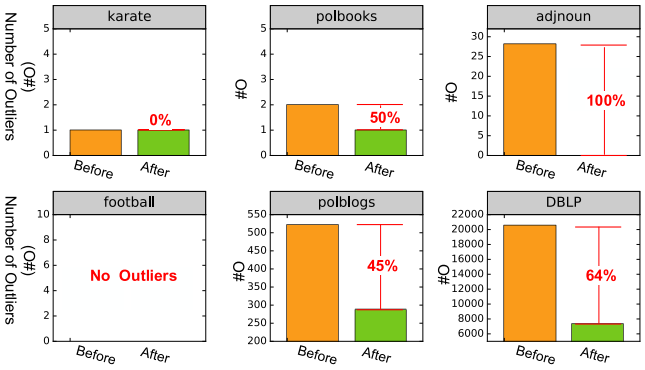


FIGURE 8. Outlier optimization performances on real-world networks.

observations: (1) In terms of F-measure, the differences among the five algorithms are significant, where Attractor++ outperforms the other four algorithms in terms of average F-measure. (2) Focusing on the Attractor, E-Attractor and Attractor++ algorithms, the performance of E-Attractor is more stable than that of Attractor and that of Attractor++ is more stable than that of E-Attractor.

Figure 7(c) displays the ARI values of the algorithms on real-world networks, from which we make the following observations: (1) In terms of average ARI, LPA performs significantly worse than the other four algorithms. (2) Comparing the five algorithms, we find that Attractor++ has the highest efficiency and stability on most real-world networks.

Figure 7(d) displays the running times of the algorithms on real-world networks, from which we make the following observations: (1) The average running time of the LAP algorithm is slightly shorter than that of the Louvain algorithm. (2) The running times of the LPA and Louvain algorithms are a few orders of magnitude shorter than those of the Attractor, E-Attractor and Attractor algorithms. (3) The running times of the Attractor, E-Attractor and Attractor++ algorithms are very similar. Specifically, Attractor is slightly faster than E-Attractor and E-Attractor is slightly faster than Attractor++.

In summary, Attractor++, E-Attractor, Attractor and Louvain outperform LPA on both the sparse real-world networks (karate, adjnoun and DBLP) and the high-density real-world networks (polbooks, football and polblogs). In addition, on some real-world networks, Attractor++ outperforms the Attractor and E-Attractor algorithms.

3) OUTLIER OPTIMIZATION PERFORMANCE

In this subsection, we compare Attractor++ with Attractor in terms of clustering accuracy on various real-world networks. Figure 8 presents the outlier optimization results on the six real-world networks.

In Figure 8, the ''before'' column lists the numbers of outliers (#O) that are detected by Attractor without the outlier postprocessing, the ''after'' column lists the numbers of outliers (#O) that are identified by Attractor++ with the outlier postprocessing, and the red number indicates the percentage by which #O is reduced. According to Figure 8, the distance dynamics model typically detects many outliers, e.g., on the polblogs and DBLP networks. By using our proposed outlier postprocessing method, the number of outliers that are identified by Attractor++ can be substantially reduced and the accuracy of outlier identification enhanced. For example, on the polbooks network, the outlier optimization percentage reaches 50%; on the adjnoun network, all outliers are optimized; and on the DBLP network, the percentage exceeds 60%. Considering all real-world networks, the distance dynamics model faces the drawback of easily producing many rough outliers and the outlier optimization step is highly necessary for the Attractor++ algorithm. Moreover, Figure 8 demonstrates the effectiveness of the proposed outlier postprocessing method.

D. TIME STEPS

In the distance dynamics model, the dynamics of each distance is simulated according to the three interaction patterns. Before all distances in the network converge (either 1 or 0), the entire interaction process needs to go through multiple time steps. In this experiment, we compare the number of time steps of our proposed algorithm, namely, Attractor++, with that of the Attractor algorithm on two real-world networks (polblogs and DBLP).

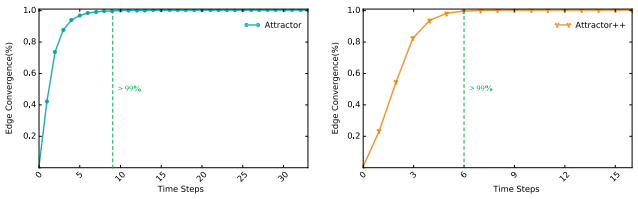


FIGURE 9. Comparison of the number of time steps for convergence on the polblogs network.

Figure 9 shows the convergence ratio of the edges at each time step on the polblogs network. The green dashed line in Figure 9 indicates the time steps when at least 99% of

the edges have converged. Attractor++ is much faster than the original algorithm, namely, Attractor. For the Attractor algorithm, it takes at least 9 time steps to achieve convergence of 99% of the distances and all distances converge after 34 time steps. However, for the Attractor++ algorithm, all distances converge after 17 time steps and it only take 6 time steps to achieve 99% convergence of 99% of the distances. Although the total number of time steps of Attractor++ is much less than that of Attractor, the convergence speed of Attractor++ is slower than that of Attractor at the early time steps. Unfortunately, the convergence speed of Attractor slows as the number of time steps increases, whereas the convergence speed of Attractor++ increases. For example, after one time step, nearly 42% of the distances have converged when the Attractor algorithm is used, compared to only 23% distances for the Attractor++ algorithm. However, after six time steps, nearly 99% of the distances have converged with the Attractor++ algorithm, compared to nearly 98% distances with the Attractor algorithm. The main reason is that Attractor adopts a global parameter setting to determine the underlying influence of exclusive neighbors on the distance, but the structures of the communities are constantly changing with the convergence of new distances. In contrast, Attractor++ adopt the dynamic membership degree to determine the underlying influence of exclusive neighbors on the distance. A similar result can easily be obtained from Figure 10; it is not discussed due to space limitations.

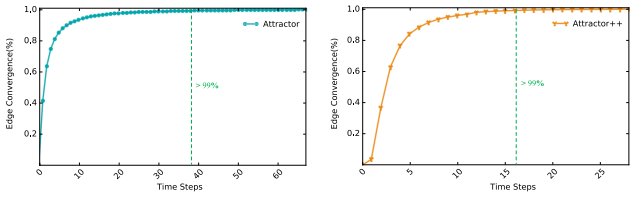


FIGURE 10. Comparison of the number of time steps on the DBLP network.

E. CASE STUDIES

To evaluate the effect effectiveness of our robust distance dynamics model, we select two well-known real-world networks, namely, dolphins (without class labels) and polbooks (with ground truth), for case studies. These real-world networks are publicly available from the UCI data repository at https://networkdata.ics.uci.edu/index.php.

The first network is the dolphins social network, which consists of 62 vertices and 159 undirected edges. There are two communities in the dataset but no class label information. Each node in the network represents a dolphin that lives in New Zealand. If two dolphins are in contact frequently, there is an edge between their two nodes. Figure 11 shows the detection results that were obtained by the Attractor algorithm, which identified 2 communities and 11 outliers. In Figure 11, all nodes that are the same color belong to the same community and the 11 green nodes are outliers. Of the 11 outliers that were identified by the Attractor algorithm,

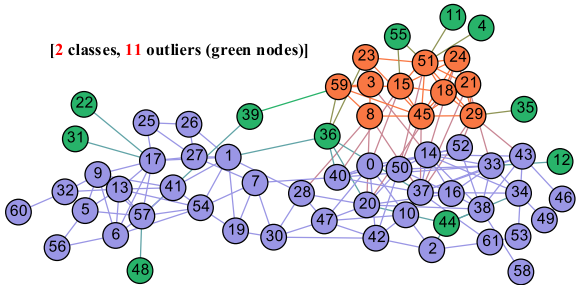


FIGURE 11. Case study on the dolphins network using the Attractor algorithm.

7 are leaf nodes and we cannot intuitively decide whether these nodes are real outliers. For the other 4 outliers, we can use the outlier postprocessing algorithm to further optimize the results.

Figure 12 shows the detection results that are obtained by the Attractor++ algorithm, which identifies 2 communities and 7 outliers. According to Figure 12, the 3 outliers (nodes 36, 44 and 55) in the red dashed circle are optimized and the other 8 outliers are filtered out in the optimization process. Specifically, nodes 4, 11, 12, 22, 31, 35 and 48 are leaf nodes and do not satisfy the optimization rules and nodes 36 and 44 should be merged into the light-blue community because all the triangles of these two nodes are only adjacent to the light-blue community. Similar to nodes 36 and 44, node 55 should be merged into the light-red community. Because node 39 is not in any of the triangles, it is an outlier.

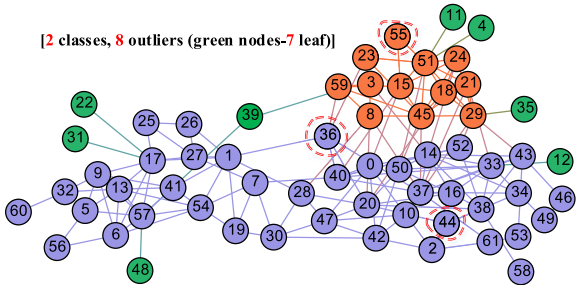


FIGURE 12. Case study on the dolphins network using the Attractor++ algorithm.

The second network is the Books About US Politics network, which is referred to as the polbooks network and consists of 105 nodes and 441 edges. There are three communities in the network and ground-truth information is available. Each node in the network represents a book about US politics. An edge between two books indicates that they are often purchased together by customers. Figure 13(a) shows the ground truth of the polbooks network, which covers 3 clusters. Figure 13(b) shows the detection results that were obtained by the Attractor algorithm, which identified 4 communities and 6 outliers (red dashed circle). Figure 13(c) shows the detection results that were obtained by the Attractor++ algorithm, which identified 3 communities and 1 hub (red dashed circle). Comparing Figure 13(b) to

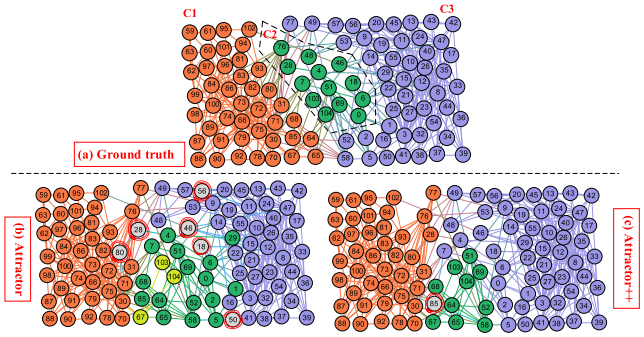


FIGURE 13. Case study on the polbooks network.

Figure 13(a) and Figure 13(c) to Figure 13(a), Attractor++ performs better in identifying ground-truth communities.

Based on the above two case studies, we make the following remarks: (1) Our algorithm, namely, Attractor++, can effectively identify vertices that have special roles (hubs and outliers). (2) Our robust distance dynamics model, which is based on the dynamic membership degree, is effective on various networks.

V. CONCLUSIONS

In this paper, we have presented the novel concept of dynamic membership degree. It enables us to avoid strong dependence on the cohesion parameter λ. Thus, we can conveniently identify high-quality communities. Based on this concept, a robust distance dynamics model has been developed, along with a robust community detection algorithm: Attractor++. Moreover, to improve the accuracy of outlier node identification, we further propose two optimization rules for judging whether an outlier should be merged into same community as its triangles or be classified as a hub. We conduct extensive experiments on both synthetic and real-world networks, and the results demonstrate the effectiveness and efficiency of the proposed algorithm.

However, complex networks in the real world change dynamically over time and their community structures are dynamically updated. In the face of dynamic networks with complex changes, designing dynamic community discovery algorithms that are based on distance dynamics models requires further study. In addition, multiobjective optimization, game theory, statistics and other theories can be used in dynamic community discovery scenarios to design better-performing dynamic community discovery algorithms.

Acknowledgment

Tao Meng and authors thank the experimental equipments provided by National Super Computing Center of Changsha, located in Hunan province of China.

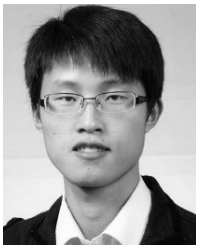
REFERENCES

[1] J. Duch and A. Arenas, "Community detection in complex networks using extremal optimization," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 72, no. 2, p. 027104, 2005.
[2] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, nos. 3–5, pp. 75–174, 2010.
[3] S. Fortunato and M. Barthélemy, "Resolution limit in community detection," *Proc. Nat. Acad. Sci. USA*, vol. 104, no. 1, pp. 36–41, 2007.
[4] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 80, no. 5, p. 056117, 2009.
[5] W. Cui, Y. Xiao, H. Wang, and W. Wang, "Local search of communities in large graphs," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2014, pp. 991–1002.
[6] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, no. 6295, pp. 163–166, 2016.
[7] L. Chen, J. Zhang, L. Cai, and Z. Deng, "Fast community detection based on distance dynamics," *Tsinghua Sci. Technol.*, vol. 22, no. 6, pp. 564–585, Dec. 2017.
[8] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[9] L. Wang, Y. Xiao, B. Shao, and H. Wang, "How to partition a billion-node graph," in *Proc. IEEE 30th Int. Conf. Data Eng. (ICDE)*, Mar./Apr. 2014, pp. 568–579.
[10] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 23, pp. 8577–8582, 2006.
[11] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, p. P10008, 2008.
[12] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger, "SCAN: A structural clustering algorithm for networks," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2007, pp. 824–833.
[13] H. Shiokawa, Y. Fujiwara, and M. Onizuka, "SCAN++: Efficient algorithm for finding clusters, hubs and outliers on large-scale graphs," *Proc. VLDB Endowment*, vol. 8, no. 11, pp. 1178–1189, 2015.
[14] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proc. Nat. Acad. Sci. USA*, vol. 105, no. 4, pp. 1118–1123, 2008.
[15] J. Shao, Z. Han, Q. Yang, and T. Zhou, "Community detection based on distance dynamics," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 1075–1084.
[16] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 76, no. 3, p. 036106, 2007.
[17] J. Huang, H. Sun, Q. Song, H. Deng, and J. Han, "Revealing density-based clustering structure from the core-connected tree of a network," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 8, pp. 1876–1889, Aug. 2013.
[18] 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 *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage.*, 2010, pp. 219–228.
[19] L. Chang, W. Li, L. Qin, W. Zhang, and S. Yang, "*pSCAN*: Fast and exact structural graph clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 387–401, Feb. 2017.
[20] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Proc. Int. Symp. Comput. Inf. Sci.* Berlin, Germany: Springer, 2005, pp. 284–293.
[21] Y. Wu, R. Jin, J. Li, and X. Zhang, "Robust local community detection: On free rider effect and its elimination," *Proc. VLDB Endowment*, vol. 8, no. 7, pp. 798–809, 2015.
[22] C. Böhm, C. Plant, J. Shao, and Q. Yang, "Clustering by synchronization," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 583–592.
[23] L. Fan, S. Xu, D. Liu, and Y. Ru, "Semi-supervised community detection based on distance dynamics," *IEEE Access*, vol. 6, pp. 37261–37271, 2018.
[24] J. Virant and N. Zimic, "Attention to time in fuzzy logic," *Fuzzy Sets Syst.*, vol. 82, no. 1, pp. 39–49, 1996.
[25] S. Wu and M. J. Er, "Dynamic fuzzy neural networks-a novel approach to function approximation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, no. 2, pp. 358–364, Apr. 2000.
[26] M. Cerrada, J. Aguilar, E. Colina, and A. Titli, "Dynamical membership functions: An approach for adaptive fuzzy modelling," *Fuzzy Sets Syst.*, vol. 152, no. 3, pp. 513–533, 2005.
[27] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó, "Fuzzy communities and the concept of bridgeness in complex networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 77, no. 1, p. 016107, 2008.

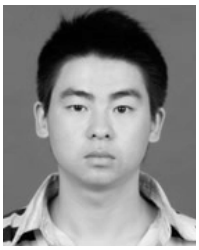
[28] S. Kundu and S. K. Pal, "Fuzzy-rough community in social networks," *Pattern Recognit. Lett.*, vol. 67, pp. 145–152, Dec. 2015.
[29] W. Luo, D. Zhang, H. Jiang, L. Ni, and Y. Hu, "Local community detection with the dynamic membership function," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 5, pp. 3136–3150, Oct. 2018.
[30] A. Prat-Pérez, D. Dominguez-Sal, J. M. Brunat, and J.-L. Larriba-Pey, "Shaping communities out of triangles," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 1677–1681.
[31] C. Lijun, Z. Jing, C. Lei, and H. T. Qin, "Enhanced distance dynamics model for community detection via ego-leader," *KSII Trans. Internet Inf. Syst.*, vol. 12, no. 5, pp. 2142–2161, 2018.
[32] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *J. Stat. Mech.*, vol. 2005, no. 9, p. 09008, 2005.
[33] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Publications Amer. Stat. Assoc.*, vol. 66, no. 336, pp. 846–850, 1971.
[34] R. R. Larson, "Introduction to information retrieval," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 61, no. 4, pp. 852–853, 2010.

TAO MENG is currently pursuing the Ph.D. degree with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. His research interests include date mining and network analysis.

LIJUN CAI received the Ph.D. degree from the College of Computer Science and Electronic Engineering, Hunan University, in 2007. He is currently a Professor with Hunan University. His research interests include bioinformatics, cloud computing, big data scheduling, and management.



TINGQIN HE received the M.S. degree from the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, where he is currently pursuing the Ph.D. degree. His research interests include date mining, cloud computing, and big data analysis.



LEI CHEN is currently pursuing the Ph.D. degree with Hunan University, China. His research interests are mainly in modeling and scheduling of distributed computing systems, approximation and randomized algorithms, game theory, and grid and cloud computing.

ZIYUN DENG received the Ph.D. degree from the College of Electrical and Information Engineering, Hunan University, in 2016. He is currently a Professor with the Changsha Commerce and Tourism College. His research interests include high-performance computing and logistics information technology.

WEIPING DING (M'16) received the Ph.D. degree in applied computing from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2013. He was a Visiting Researcher with the University of Lethbridge, Lethbridge, AB, Canada, in 2011. From 2014 to 2015, he was a Post-Doctoral Researcher with the Brain Research Center, National Chiao Tung University, Hsinchu, Taiwan. In 2016, he was a Visiting Scholar with the National University of Singapore, Singapore. From 2017 to 2018, he was a Visiting Scholar with the University of Technology Sydney, Ultimo, NSW, Australia. He has published over 50 papers in flagship journals and conference proceedings as the first author. To date, he holds 10 approved invention patents among a total of 18 issued patents. His current research interests include data mining, machine learning, and granular computing.

ZEHONG CAO received the B.E. degree in electronic and information engineering from Northeastern University in 2012, the M.S. degree in electronic engineering from The Chinese University of Hong Kong in 2013, and the dual Ph.D. degrees in information technology and electrical and control engineering from UTS and National Chiao Tung University (NCTU), respectively, in 2017. He is currently a Post-Doctoral Research Fellow with the Centre for Artificial Intelligence, University of Technology Sydney, Australia. His research interests cover data science, human-machine interfaces, computational intelligence, pattern recognition, machine learning, and clinical applications. He received the NCTU and Songshanhu Scholarship in 2013, the UTS President Scholarship in 2015, the UTS CAI Best Paper Award in 2017, the UTS FEIT Publication Award in 2017, and the CAMP in 2018. He serves as an Associate Editor for the IEEE Access and an Editorial Board Member for SCI journals, including *Advances in Robotics and Automation* and the *International Journal of Sensor Networks and Data Communications*. He was invited to give oral presentations at the IJCNN in 2015, the BME Annual Conference of Taiwan in 2015, and the IEEE-FUZZY in 2017.