International Journal of Modern Physics C
Vol. 31, No. 11 (2020) 2050155 (18 pages)

DOI: 10.1142/S0129183120501557

World Scientific
www.worldscientific.com

An improved algorithm for detecting community defined by node-to-node dynamic distance

Jiaxin Wan[*,†], Dingding Han[*,§,‖], Zhengzhuang Yang[*] and Ming Tang[‡,¶,‖]

[*]*School of Information Science and Technology*
Fudan University, Shanghai 200433, P. R. China

[†]*School of Communication and Electronic Engineering*
East China Normal University, Shanghai 200241, P. R. China

[‡]*School of Physics and Electronic Science*
East China Normal University
Shanghai 200241, P. R. China
[§]*ddhan@fudan.edu.cn*
[¶]*tangminghan007@gmail.com*

Received 19 February 2020
Accepted 17 July 2020
Published 11 September 2020

The study of community structure is of great significance when analyzing the structural and functional characteristics of networks. *Attractor* is a fast community detection method with the advantage of high accuracy for complex networks. However, in the connected nodes interaction model proposed by the Attractor algorithm, there is a problem with slow convergence during the distance updating process. To solve this problem, we propose an improved Attractor algorithm based on the change trend of the distances between connected nodes. We have generally found that distances between connected nodes exhibit a consistent trend. The dynamic distance trend is determined by setting a window of evaluation. The convergence of the Attractor algorithm is accelerated by the consistent change trend. Experiments on datasets for real-world networks and synthetic networks have shown that our proposed algorithm not only maintains high-quality communities, but also reduces the calculation time significantly and greatly improves the speed of the algorithm.

Keywords: Community detection; improved attractor algorithm; node-to-node dynamic distance.

PACS Nos.: 89.20.Ff, 89.75.Fb.

1. Introduction

In recent years, research on complex networks has received extensive attention. Theory and research methods for complex networks are increasingly employed in real-world complex systems. Examples can be found in biology, computer science,

‖Corresponding authors.

sociology, and ecology.[1–3] Such real systems can be represented by graphs. Graphs, which are also called networks, are a collection of nodes linked by edges. For example, nodes and edges are referred to as people and virtual acquaintanceships in the Facebook network, and pages and hyperlinks in the World Wide Web.

To explore how a network is organized further, community detection has become a hot topic.[4–8] Researchers have found that community structure in a network is closely related to the function within the system.[9] For example, in protein–protein interaction networks, communities are groups of proteins having the same function[10]; in the World Wide Web, they are clusters of HTML pages with related themes[11]; in social networks, they correspond to a group of people with the same interests or opinions.[12] Community structures may offer insight into the units of structural function of the network. For example, Martin *et al.*[13] studied the problem of community detection in uncertain network data. Protein–protein interaction networks[14] are networks with only probabilistic estimates for the presence of each interaction. The communities found in this network help to infer interactions that have low probability. Peixoto *et al.*[15] studied the US air traffic network and found that Las Vegas and Atlanta were the major hubs. The difference is that the former was usually the traveler's destination, while the latter was usually the transfer hub for other destinations. Thus, combining the topological structure of networks and the dynamics of networks can provide a better understanding of the mechanisms of complex systems.

There are a series of works related to community detection, such as methods based on different indicators that include betweenness,[16] normalized cut[17] and modularity.[18–21] Methods that are based on running dynamical processes on the network include random walk,[22,23] synchronization[24,25] and label propagation.[26–28] Other methods include statistical inference,[29,30] and spectral clustering.[31,32] Among the large number of community detection methods, the modularity[19] proposed by Newman and Girvan is the most popular quality function. High modularity usually represents the high quality of a partition of the network. However, the scale of communities determined by an algorithm based on modularity is generally large, and there is a problem with resolution limit.[33] The dynamics-based method can properly solve the problem of resolution limitation. Inspired by the dynamics in the network, Shao *et al.*[34] proposed an algorithm called *Attractor* to detect communities based on the dynamics of the distances between nodes, and finally discovered the potential community structure of a network by simulating the dynamic evolution of the distances between nodes. The Attractor algorithm directly obtains community detection results based on the network topology, and finds communities of different sizes with a time complexity of $|E|$, which is suitable for large-scale networks. The Attractor algorithm is also superior to classic algorithms such as Louvain and Infomap in terms of evaluation indicators such as modularity and normalized mutual information (NMI). However, if the network is large, the Attractor algorithm requires more iterations, and there is still potential for the algorithm to run faster.

We found that most distances between connected nodes showed trends during the experiment. That is, the distances between connected nodes continued to increase to 1, or to decrease to 0, and the case in which distances increased somewhat and then decreased happened rarely. Inspired by this phenomenon, we propose an improved Attractor algorithm based on the trend of the change in the distance between connected nodes, called the T-Attractor, which can reduce the number of iterations and improve the speed of the Attractor algorithm. To identify the trends, we set a window on the distance dynamics model proposed by Attractor. When the change in distance between node u and v continues to increase or decrease within the window, we consider that the edge has reached a steady state, and update its distance to 1 or 0, respectively. This can speed up the time required for all distances to converge to 0 or 1. Experiments on synthetic and real-world networks proved that our algorithm not only rivaled the accuracy of the Attractor algorithm, but also outperformed the Attractor algorithm in terms of speed, with T-Attractor being up to 1–3 times faster than Attractor in synthetic networks.

In Sec. 2, we first illustrate the original Attractor algorithm. In Sec. 3, our new improved algorithm, the T-Attractor, is proposed for accelerating the Attractor algorithm and parameter of the new algorithm is discussed. In Sec. 4, the T-Attractor is tested on synthetic benchmarks and empirical networks. Comparisons between the original Attractor algorithm and our T-Attractor are also made.

2. Attractor Algorithm

2.1. *Basic definition*

$G = (V, E, W)$ is an undirected and unweighted graph, where V is the set of nodes, E is the set of edges, and W is the set of edge weights. $e = (u, v) \in E$, which represents a connection between the nodes u and v, and $w(u, v)$ represents the weight of the connection between nodes u and v, in an undirected and unweighted graph, $w(u, v) = 1$.

Definition 1. Given an undirected graph $G = (V, E, W)$, the set containing node u and the neighbors of node u is $\Gamma(u)$ as follows:

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

Definition 2. Given an undirected graph $G = (V, E, W)$, the Jaccard distance[35] of two nodes, u and v, is defined as follows:

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

For weighted graphs, the Jaccard distance of two nodes u and v is defined as

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

2.2. *Distance dynamics interaction model*

This section briefly introduces the dynamic distance interaction model. This model envisions the network as an adaptive dynamic system, and the distance between connected nodes is our research object. During the evolution of the system, nodes will interact with their neighbors, and the interaction changes the distance between nodes. If the nodes and their neighbors belong to the same community, the interaction between them will decrease the distance. Conversely, if the nodes and its neighbors belong to different communities, the interaction between them will increase the distance between them. Finally, the distance between nodes belonging to the same community becomes smaller, the distance between nodes belonging to different communities increases, and the potential community structure gradually emerges.

In the initial state of the system, the Attractor algorithm uses Jaccard similarity to calculate the initial distance between each pair of connected nodes. As the system evolves, there are three interaction patterns that affect the distance between nodes u and v.

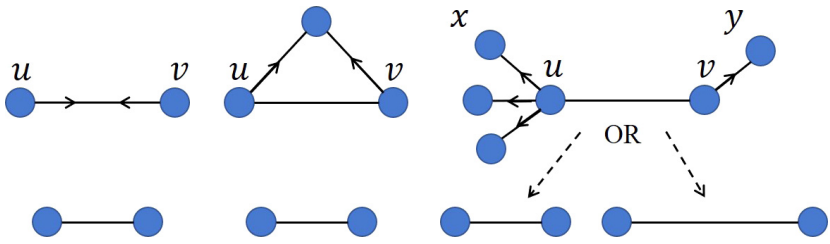
The direct connection between u and v can attract these nodes to each other and cause the distance to decrease. As shown in Fig. 1(a), we formally define the DI function to represent the influence of directly connected nodes on the change in the distance $d(u, v)$:

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

where $f(\cdot)$ represents a coupling function and $\sin(\cdot)$ is used in this paper, $\deg(u)$ represents the degree of the node u.

The common neighbors $\mathrm{CN} = (\Gamma(u) - u) \cap (\Gamma(v) - v)$ of nodes u and v may also cause the distance to decrease. As shown in Fig. 1(b), we formally define the CI function to represent the influence of CN on the change in the distance $d(u, v)$:

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



(a) Pattern one (b) Pattern two (c) Pattern three

Fig. 1. (Color online) Illustration of three distinct interaction patterns.

The third influence is by the exclusive neighbors (EN) of node u or node v, $\mathrm{EN}(u)$, $\mathrm{EN}(v)$, respectively. As shown in Fig. 1(c), we formally define the change in $d(u,v)$ by the influence of exclusive neighbors, as follows:

$$\mathrm{EI} = -\sum_{x\in EN(u)} \frac{f(1-d(x,u))\cdot\rho(x,u)}{\deg(u)} - \sum_{y\in \mathrm{EN}(v)} \frac{f(1-d(y,v))\cdot\rho(y,v)}{\deg(v)}, \tag{6}$$

where $\mathrm{EN}(u) = \Gamma(u) - \Gamma(u)\cap\Gamma(v)$, $\mathrm{EN}(v) = \Gamma(v) - \Gamma(u)\cap\Gamma(v)$ and $\rho(x,u)$ represents that the influence of exclusive neighbors, nodes u or v, on $d(u,v)$ is positive or negative.

$$\rho(x,u) = \begin{cases} 1-d(x,v), & (1-d(x,v)) \geq \lambda, \\ 1-d(x,v)-\lambda, & \text{otherwise.} \end{cases} \tag{7}$$

If the distance between the exclusive neighbor x and node v is small, it means that x and node v are similar. When the similarity is greater than the threshold λ, the influence from the interaction between x and u causes $d(u,v)$ to decrease. When the similarity is less than the threshold λ, the influence of the interaction between x and u causes $d(u,v)$ to increase. $d(x,v)$ is given by Eq. (3):

$$d(x,v) = 1 - \frac{\sum_{z\in\Gamma(x)\cap\Gamma(v)}((1-d(x,z)) + (1-d(v,z)))}{\sum_{p\in\Gamma(x)}(1-d(p,x)) + \sum_{q\in\Gamma(v)}(1-d(v,q))}. \tag{8}$$

Through the above three interaction modes, we can obtain the total change in distance between u and v over time, and the renewed distance at time step $t+1$ over time:

$$\Delta d(u,v;t) = \mathrm{EI}(u,v;t) + \mathrm{CI}(u,v;t) + \mathrm{DI}(u,v;t), \tag{9}$$

$$d(u,v;t+1) = d(u,v;t) + \Delta d(u,v;t). \tag{10}$$

As the system evolves, the distance between nodes gradually changes. When the distance between nodes is greater than 1, we set the distance to 1, and consider the distance between the pair of nodes to have reached a steady state. When the distance between nodes is less than 0, we set the distance between the pair of nodes equal to 0, and also consider the distance between the pair of nodes to have reached a steady state. In the end, all distances are equal to 0 or 1. When we remove those edges that are equal to 1, we obtain the community structure of the network.

3. Improved Attractor Algorithm Based on the Trends of Distances Between Connected Nodes

3.1. *Algorithm*

Through an analysis of the Attractor algorithm, it was found that the algorithm may need to calculate the change in distance during the iteration process. This requires considerable time to traverse neighbors when calculating a change in distance. If the number of iterations required for all distances to converge can be reduced, the runtime of Attractor will be reduced.

In the experiments, we found that the distances between most connected nodes decreased or increased incrementally during the evolution process. If the trend in distance change could be determined, the final stable value of distance could also be determined. Therefore, in this paper, we consider the change trend of the distance between connected nodes as a basis for the stable value of distance. If the distance between nodes u and v, $\Delta d(u, v; t)$, continues to be greater than 0 or less than 0, we consider that this edge has reached a steady state, and thus the convergence speed of the algorithm increases. For example, in social networks, if the distance between two people gets smaller, that is, if they become closer, the relationship between them becomes more intimate and they eventually form a small group. If the distance gets longer, the interaction between them decreases, and they will eventually belong to different communities.

3.2. *Model*

This section details the principle of the T-Attractor algorithm.

In our algorithm, we set a time window η to evaluate the trend of the change in distance. Within this window, we calculate the change in distance $\Delta d(u, v; t)$ at each step. The distance has an increasing trend when it increases continuously in the window and a decreasing trend when it decreases continuously in the window. Specifically, we use $s(u, v; t)$ to represent the number of iteration steps for each pair of nodes to continuously increase or decrease in the window. If the sign of the distance change between the previous step $t - 1$ and the current step t is the same, then $s(u, v; t) = s(u, v; t - 1) + 1$. Otherwise, $s(u, v; t) = 0$. In other words, if each step $\Delta d(u, v; t)$ is greater than 0 in the window, we will get $s(u, v; t) = \eta$, and the distance converges to the maximal distance 1. Conversely, if each step $\Delta d(u, v; t)$ is less than 0 in the window, we will also get $s(u, v; t) = \eta$, and the distance converges to the minimal distance 0. Consider a special case, if for some steps in the window, $\Delta d(u, v; t) > 0$, and for other steps $\Delta d(u, v; t) < 0$, then we will update $s(u, v; t) = 0$ when the sign of $\Delta d(u, v; t)$ changes. After several iterations, all distances quickly converge, and the community structure of the network can be identified by removing all edges with a distance of 1. The basic flow of the algorithm is described as follows:

(i) In the initial state, $t = 0$, the distances are calculated by Jaccard similarity.
(ii) Then the distance of each edge is calculated according to three interaction patterns. If $\Delta d(u, v; t) \cdot \Delta d(u, v; t - 1) < 0$, then update $s(u, v; t)$ to equal 0, otherwise update $s(u, v; t) = s(u, v; t - 1) + 1$.
(iii) When $s(u, v; t) = \eta$ and $\Delta d(u, v; t) < 0$, update the distance to minimal value 0. When $s(u, v; t) = \eta$ and $\Delta d(u, v; t) < 0$, update the distance to maximal value 1. Otherwise, update the distance $d(u, v; t + 1) = d(u, v; t) + \Delta d(u, v; t)$.
(iv) Repeat Steps (ii) and (iii) until all distances are convergence.
(v) Remove edges with a distance of 1 to obtain the community structure in the network.

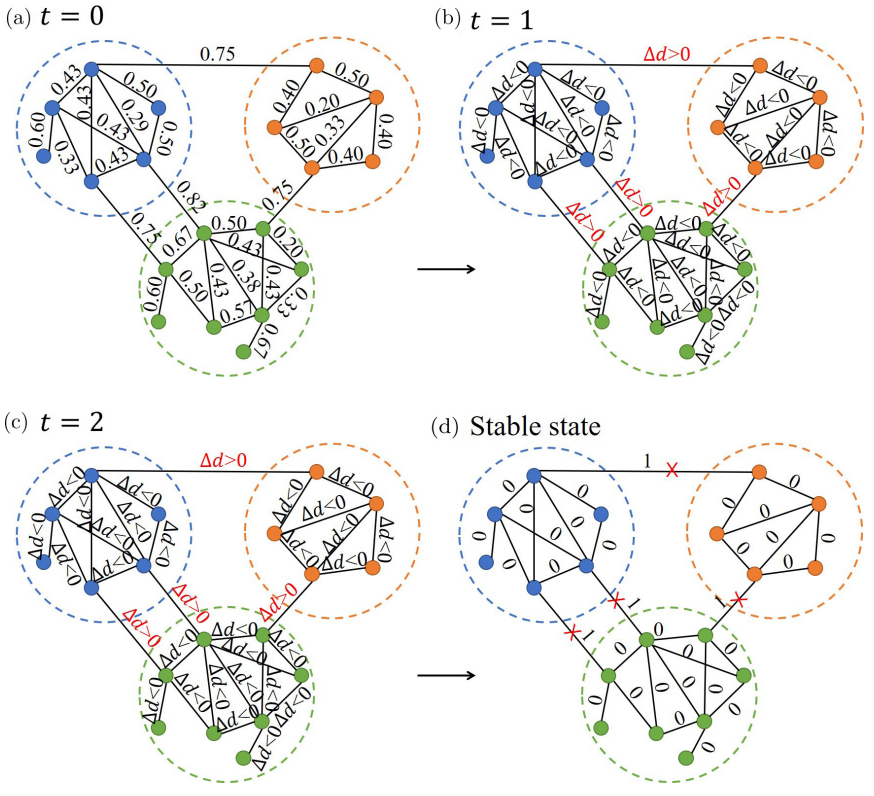


Fig. 2. (Color online) Illustration of the distance dynamics model based on the distance evolution trends of the connected nodes. (a) Example network. The numbers on the edges represent the initial distances between the nodes. (b) The differences in the distances after a single time step; $\Delta d > 0$ means the distance is increasing; $\Delta d < 0$ means the distance is decreasing. (c) The differences in the distances after two time steps. (d) The steady state of the network.

The T-Attractor algorithm flow is shown in Fig. 2. In this illustration, we set the window to $\eta = 2$. Figure 2(a) shows the initial distance of the network, Fig. 2(b) shows the distance change for each edge after one distance update step $\Delta d(0)$, and Fig. 2(c) shows the change in distance of each edge after two distance update steps $\Delta d(1)$. After two time steps, the distances of all edges are decreased or increased incrementally. Therefore, we update the converge value of each edge to 0 or 1 according to whether the change trend in distance of edges are decreasing or increase within the window, and the final community detection result shown in Fig. 2(d) can be obtained. In the end, there are three communities. The T-Attractor algorithm is further given as Algorithm 1.

As shown in Fig. 3, taking the Karate network as an example. All distances converge to 0 or 1 after 12 iterations when Attractor is executed on the Karate network. However, T-Attractor only needs 4 iterations to reach convergence. In this example, the time window is 2.

Algorithm 1. T-Attractor

Input: Graph $G = (V, E, W)$, cohesive parameter λ, time window η
Output: $C = \{c_1, c_2, c_3 ... c_k\}$ and k is the number of community.
Initialize the distance $d(u, v; 0)$ of each edge e from E using Eq. (3)
FLAG = TRUE
while FLAG **do**
  FLAG = FALSE
  for each edge $e = \{u, v\} \in E$ **do**
    if $d(u, v; t) <= 0$ **or** $d(u, v; t) >= 1$ **then**
      continue
    end if
    $\Delta d(u, v; t) = DI(u, v; t) + CI(u, v; t) + EI(u, v; t)$ using Eqs. (4)–(6)
    if $\Delta d(u, v; t) = 0$ **then**
      continue
    end if
    if $\Delta d(u, v; t) \cdot \Delta d(u, v; t-1) > 0$ **then**
      $s(u, v; t) = s(u, v; t-1) + 1$
    else
      $s(u, v; t) = 0$
    end if
    if $s(u, v; t) = \eta$ **and** the trend of e is increasing **then**
      $d(u, v; t+1) = 1$
    else if $s(u, v; t) = \eta$ **and** the trend of e is decreasing **then**
      $d(u, v; t+1) = 0$
    else
      $d(u, v; t+1) = d(u, v; t) + \Delta d(u, v; t)$
      if $d(u, v; t+1) > 1$ **then**
        $d(u, v; t+1) = 1$
      end if
      if $d(u, v; t+1) < 0$ **then**
        $d(u, v; t+1) = 0$
      end if
      FLAG=TRUE
    end if
  end for
end while
Find the resulting communities $\{c_1, c_2, c_3 ... c_k\}$ by removing all edges with $d(u, v; t+1) = 1$

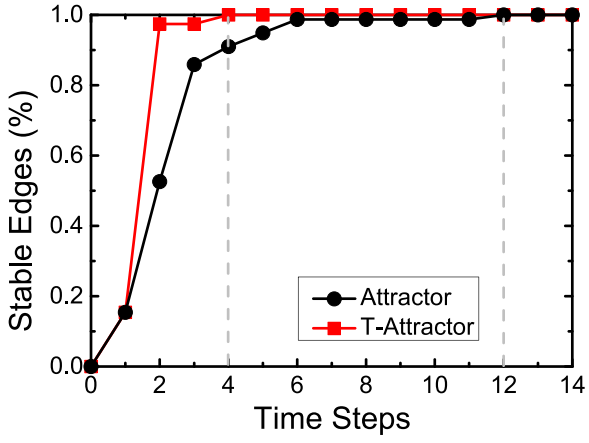


Fig. 3. (Color online) Iterations of two algorithms on the Karate network.

3.3. *Estimation of η*

In order to evaluate the trend of distance, we have added a time window η to the Attractor. The window η is used to fast determine the final value of the distance. For small value of η, the number of iteration steps required to determine the distance trend is small. When η increases, the distance convergence speed gets slowly and communities obtained by algorithm become different. In order to study the influence of η on the algorithm performance, we use the LFR[36] benchmark to generate four synthetic networks with known community structure.

Figure 4 illustrates the results of the analysis. The following input parameters are the same for the four benchmark networks used here: the network size is 1000, the maximum degree is 50, the maximum community size is 50, the minimum

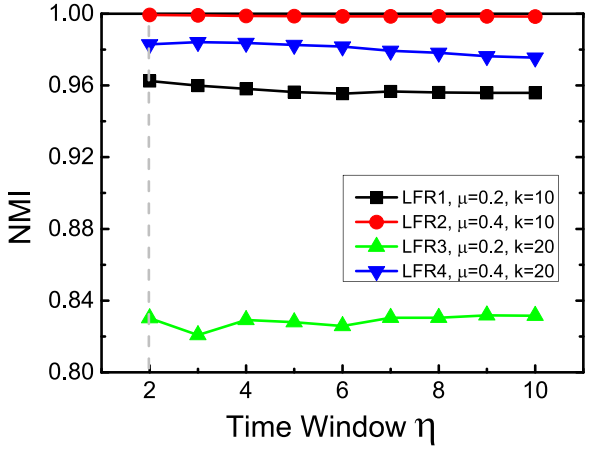


Fig. 4. (Color online) The performance of T-Attractor with different η on four LFR networks. When η increase, the performance will not improve much.

Table 1. Parameters of the 64 kinds of LFR networks.

Parameter	Value
Number of nodes	1000, 5000
Average degree	10, 15, 20, 25
Mixing parameter	0.2, 0.4, 0.6, 0.8
Maximum degree	50
{Minimum, Maximum} community size	{10, 50}, {20, 100}
Degree distribution exponent	−2
Community size distribution exponent	−1

community size is 10, the exponent of the degree distribution is −2, and the community size distribution is −1. In the plot, we show four curves, corresponding to two different mixing parameter (0.2 and 0.4) and, for a given μ, to two different average degree (10 and 20). And we use NMI[37] to evaluate the accuracy of communities obtained on each network. Then we change the window η from 2 to 10 to observe the performance of the T-Attractor. As η increases (see Fig. 4), the NMI will not improve much. Just choosing small η will obtain good performance.

To further study the effect of possible differences in network properties on the best time window η, we again use the LFR benchmark to randomly generate 64 kinds of LFR networks with different parameters. The combinations of the parameters of LFR networks are shown in Table 1. And we change η from 2 to 10 to observe the performance of T-Attractor on every network. For each window η, we count how many networks it can get the highest NMI on 64 kinds of LFR networks used here, and select the window that can get the highest NMI on most networks as the best window.

The plot of Fig. 5, stacked column charts, illustrates the result of the analysis. In the figure, the two different colors correspond to two different ranges for the community sizes, indicated by the legends "Small Community" and "Big Community": "Small Community" means that communities have between 10 and 50 nodes and "Big Community" means that communities have between 20 and 100 nodes. As we can see from Fig. 5, the improved algorithm can obtain the best results on nearly half of the 64 networks when the time window η is set to 2. Therefore, in the following experiments, we set $\eta = 2$.

3.4. *Complexity*

Compared with the Attractor algorithm, our T-Attractor algorithm has a lower time complexity, because it accelerates distance convergence according to the trends of distance change between iterations, requiring a fewer number of iterations. The complexity of the T-Attractor algorithm is further analyzed: the Jaccard similarity between any two connected nodes must be calculated during initialization, so the time complexity is $O(|E|)$, and the time computation for the influence of exclusive neighbors is $O(k \cdot |E|)$, where k is the average number of exclusive neighbors of the

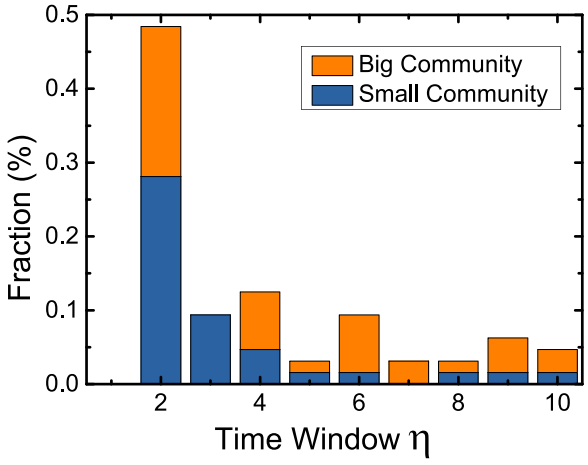


Fig. 5. (Color online) The fraction of the best results obtained by T-Attractor with different η among 64 kinds of LFR networks. Small Community: networks with communities that have between 10 and 50 nodes, Big Community: networks with communities that have between 20 and 100 nodes.

two connected nodes. Then, it needs T steps to converge, and the time complexity is $O(T \cdot |E|)$, so the total time complexity is $O(T \cdot |E| + k \cdot |E| + |E|) = O(c \cdot |E|)$, where c is a constant. It can be seen that the total time complexity is only related to the edges of the network. In most cases, $3 \leq T \leq 15$, while for Attractor, $3 \leq T \leq 50$.

4. Experiments Results

In this section, we evaluate the performance of the Attractor and T-Attractor algorithms on LFR synthetic benchmark networks and real-world networks. The experimental environment uses an Intel Xeon Platinum 8269CY 2.50 GHz CPU, and 32 GB RAM. The cohesion parameter λ for Attractor and T-Attractor is set to 0.6.

4.1. *Evaluation*

NMI[37]: NMI is used to evaluate the similarity between the community divided by the algorithm and the real community. The value of NMI ranges from 0 to 1, and is given by the following equation:

$$\mathrm{NMI}(X, Y) = \frac{2I(X, Y)}{H(X) + H(Y)}, \tag{11}$$

where X represents the community structure divided by the algorithm, and Y represents the real community structure. $I(X, Y)$ represents the mutual information of X and Y, and $H(X)$ and $H(Y)$ represent, respectively, the information entropy of X and Y. The higher the NMI value, the more similar X and Y.

Modularity[38]: Modularity is another popular measure of the performance of different algorithms. The calculation of modularity does not require the actual

community structure, but instead compares it with the corresponding zero model of the network to measure the quality of the algorithm's partition results.

4.2. *Synthetic network*

To evaluate the performance of Attractor and the proposed algorithm in terms of accuracy and running speed, we first use the LFR benchmark[36] as the test network and choose NMI and modularity as the evaluation criteria for accuracy. The LFR benchmark can generate networks with different distributions of community size and degree through parameter settings. The mixing parameter μ controls the complexity of the community separation. Communities are more easily detected when μ is low. When μ becomes large, communities become difficult to detect.

First, we compare the performance of two algorithms, the Attractor and our T-Attractor, on LFR benchmark networks with different mixing parameters μ. In addition, we compare with another algorithm, F-Attractor,[39] which also improves the speed of the Attractor. We set the network size as $N = 2000$ and the average degree $k = 20$, and adjusted the change in the mixing parameters μ from 0.1 to 0.8. As shown in Fig. 6, the T-Attractor is as good as Attractor in terms of NMI and modularity when μ is less than 0.7. However, when μ is greater than 0.7, Attractor outperforms T-Attractor in modularity. This difference is likely because the window ($\eta = 2$) could not correctly determine the change trend in distance when μ was close to 1. As for F-Attractor, its NMI is close to 0 when $\mu > 0.6$, and it is almost impossible to detect the community correctly.

In terms of the speed of the algorithm, T-Attractor is clearly faster than Attractor. When μ is less than 0.3, there is not a large difference in runtime between the two algorithms. However, when μ is greater than 0.3, T-Attractor is much faster

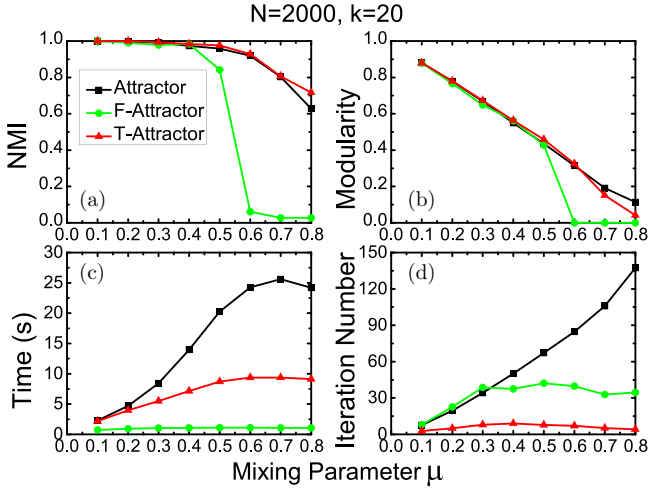


Fig. 6. (Color online) Performance of three algorithms on LFR benchmark networks by varying the mixing parameter.

than Attractor, reaching runtimes 23 times faster for higher values of μ. On the other hand, T-Attractor maintains a low number of iterations, especially for high values of μ, for which it is more difficult to detect good partitions. For F-Attractor, it is obvious that its speed is the fastest, but it cannot maintain a high accuracy when communities become difficult to detect. Figure 6 shows that T-Attractor performs better than Attractor in terms of runtime and maintains the quality of the communities identified. Because F-Attractor could not get good results when $\mu > 0.5$, we did not compare with it in the following experiments.

Next, we study the performance of two algorithms on the LFR benchmark networks with different values for the average degree k. We set the network size as $N = 2000$, and the mixing parameter as $\mu = 0.5$, and increased the average degree k from 5 to 25. As shown in Fig. 7, the difference in quality is small in terms of NMI and modularity. Moreover, T-Attractor is also significantly faster than Attractor, reaching runtimes roughly 13 times faster than Attractor. The difference is especially large for a large average degree k. Figure 7 also shows that T-Attractor performs better than Attractor in terms of runtime.

4.3. *Real-world network*

We now compare how the two algorithms perform on real-world networks. The datasets used in this paper can be downloaded through the UCI dataset website (https://networkdata.ics.uci.edu) and the Stanford University dataset website (http://snap.stanford.edu/data/index.html). The statistical information for the datasets is shown in Table 2.

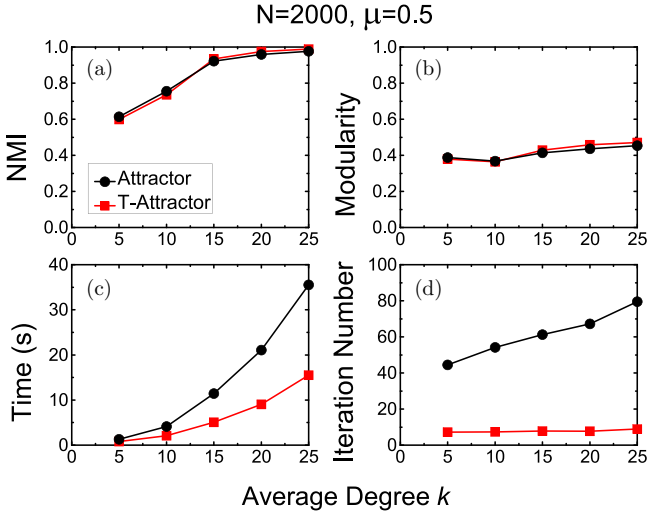


Fig. 7. (Color online) Performance of two algorithms on LFR benchmark networks by varying the average degree.

Table 2. Statistics of real-world datasets. N: number of nodes, E: number of edges, Com.: number of communities, AD: average degree, CC: clustering coefficient.

Dataset	N	E	Com.	AD	CC
Karate	34	78	2	4.588	0.571
Football	105	613	12	10.661	0.403
Polbooks	105	441	3	8.400	0.488
Amazon	334863	925872	—	5.530	0.397
Collaboration	9857	25973	—	5.260	0.312
Friendship	58228	214078	—	7.353	0.172
Road	1088092	1541898	—	2.834	0.047

Zachary's karate club[40]**:** The network represents the social relationships between 34 members of a university karate club in the 1970s. Nodes in the network represent club member, and edges represent the friend relationship between two members. This network could be divided into two communities because of the disagreement between the administrator and the instructor. This network contains 34 nodes and 78 edges.

American college football[16]**:** The network is a social network derived from the American College Football League. Nodes in the network represents football teams, and edges represent that games had been played between two teams. The 115 teams were divided into 12 conferences, which represents the real community structure of the network. The network contains 115 nodes and 613 edges.

Political books: The network was established from the politic books about US politics sold on Amazon. Nodes represent books sold on Amazon's online bookstore, and edges represent frequently co-purchased of books by the same readers. Books are divided into three categories: l, n, and c, which represent "libera", "neutra" and "conservative", respectively. These labels are divided by Mark Newman's manual analysis of book opinions and evaluations on Amazon. The network contains 105 nodes and 441 edges.

Amazon[41]**:** The network was collected from Amazon. Nodes represent products, and edges represent frequently co-purchased of products. The network contains 334 863 nodes and 925 872 edges.

Hepth collaboration[42]**:** The network is a collaboration network on the theory of high energy physics collected from the e-print arXiv. Nodes represent authors, and edges represent co-authored relationship between authors. The network contains 9875 nodes and 25 973 edges.

Brightkite friendship[43]**:** The network is a location-based social network, and it contains 58 228 nodes and 214 078 edges.

Pennsylvania road[44]**:** The network is a road network of Pennsylvania. Nodes represent intersections and endpoints, and edges represent the roads connecting these intersections or endpoints. The network contains 1088 092 nodes and 1541 898 edges.

Table 3. Performance of two algorithms on labeled real-world networks. Mod.: modularity, Iter.: number of iterations.

	Karate			Football			Polbooks		
	NMI	Mod.	Iter.	NMI	Mod.	Iter.	NMI	Mod.	Iter.
Attractor	0.927	0.371	12	0.924	0.600	8	0.585	0.510	19
T-Attractor	0.927	0.371	**4**	0.929	0.597	**4**	0.586	0.511	**5**

As can be seen in Table 3, the convergence speed of T-Attractor is faster than Attractor in real-world networks. For the Karate, Football, and Polbooks networks, which contain a known community structure, T-Attractor obtains communities of high quality, as does Attractor. At the same time, T-Attractor requires fewer iterations than does Attractor. For the Friendship, Collaboration, Amazon, and Road networks without a known community structure (see Table 4), the difference in modularity between T-Attractor and Attractor is small. However, the speed improvement realized by T-Attractor relative to Attractor is large. For the road network, due to its large sparseness, we have increased the window ($\eta = 3$) to obtain high accuracy. In summary, the experiments on all real-world networks demonstrate that T-Attractor performs better than Attractor in terms of runtime and maintains the quality of the identified communities.

The runtime of T-Attractor and Attractor on seven real-world networks is given in Table 5. It can be seen from the table that T-Attractor reduces the runtime by about half.

4.4. *Runtime*

In this section, we present the scaling of two algorithms with network size. We generated benchmark networks with edge sizes ranging from 10 000 to 1 million.

Table 4. Performance of two algorithms on real-world networks without class information. Com.: number of communities, Mod.: modularity, Iter.: number of iterations.

	Friendship			Collaboration			Amazon			Road (η=3)		
	Com.	Mod.	Iter.	Com.	Mod.	Iter.	Com.	Mod.	Iter.	Com.	Mod.	Iter.
Attractor	11651	0.429	470	1001	0.432	94	38179	0.754	67	56975	0.865	33
T-Attractor	12442	0.426	**11**	961	0.378	**7**	34228	0.779	**8**	21511	0.887	**12**

Table 5. Runtime of two algorithms on real-world networks (s).

	Karate	Football	Polbooks	Friendship	Collaboration	Amazon	Road
Attractor	0.046	0.080	0.092	288.775	2.994	104.136	49.657
T-Attractor	**0.043**	**0.076**	**0.070**	**111.984**	**1.599**	**50.397**	**28.031**

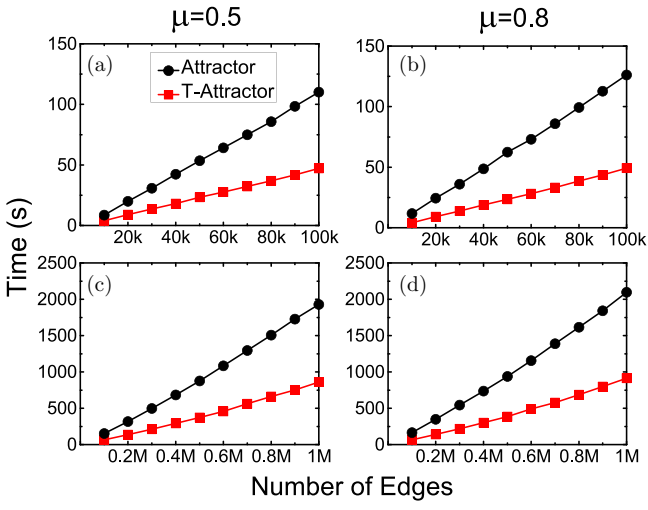


Fig. 8. (Color online) Runtime of two algorithms.

The mixing parameters were set to 0.5 and 0.8, and the average degree was set to 20 to compare the runtimes of the T-Attractor and Attractor algorithms. As shown in Fig. 8, T-Attractor was significantly faster than Attractor because T-Attractor can evaluate the final distance between nodes in advance according to the distance change trend. This speeds up convergence and shortens the runtime of the algorithm.

5. Conclusion

In this paper, we proposed an improved Attractor algorithm. Compared with the original Attractor algorithm, we set a window of evaluation in the distance dynamics model to identify the change trend of distance between iterations. The final distance value could be determined from the trend, and we reduced the computation time required for the distance to reach a stable value, thereby decreasing the number of iterations until convergence. This method for identifying trends is simple but useful, and it effectively maintains the quality of the communities and accelerates convergence speed.

We tested our improved method on synthetic benchmark networks and real-world networks. Our method offers an obvious improvement across various benchmark network datasets, especially when our algorithm is used on networks with a complex community structure. The results on real-world networks by our improved method also show excellent performance. Our method rivals the accuracy of the Attractor algorithm, and reduces its runtime.

Our improved T-Attractor algorithm can decrease the number of iterations required by the Attractor algorithm, but the required calculation time for the influence of neighbors on distance still exists. Further studies to solve this problem are required to improve the Attractor algorithm. In addition, we have added an time window to

Attractor. There may be suitable methods for setting the value η used in T-Attractor. If research is conducted on the free parameter for window η, T-Attractor may show better performance.

It can be concluded that our T-Attractor algorithm accurately detects communities using methods based on distance dynamics. It is fast because distance trends can be identified quickly. The idea of trends might be useful to the improvement of other algorithms based on a node-to-node dynamic index.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China under Contracts Nos. 11875133, 11975099, 11075057 and 11575041. We also acknowledge the National Key R&D Program of China under Grant No. 2018YFB2101302.

References

1. S. Fortunato, *Phys. Rep.* **486**, 75 (2010).
2. S. Fortunato and D. Hric, *Phys. Rep.* **659**, 1 (2016).
3. M. E. Newman, *SIAM Rev.* **45**, 167 (2003).
4. A. Clauset, M. E. Newman and C. Moore, *Phys. Rev. E* **70**, 066111 (2004).
5. A. Lancichinetti and S. Fortunato, *Phys. Rev. E* **80**, 056117 (2009).
6. J. Han, W. Li and W. Deng, *Sci. Rep.* **6**, 1 (2016).
7. L. G. Jeub, M. W. Mahoney, P. J. Mucha and M. A. Porter, *Network Sci.* **5**, 144 (2017).
8. J. Han, W. Li, L. Zhao, Z. Su, Y. Zou and W. Deng, *PloS one* **12**, e0188655 (2017).
9. D. Hric, R. K. Darst and S. Fortunato, *Phys. Rev. E* **90**, 062805 (2014).
10. V. Spirin and L. A. Mirny, *Proc. Natl. Acad. Sci.* **100**, 12123 (2003).
11. S. Kumar, R. West and J. Leskovec, Disinformation on the web: impact, characteristics, and detection of wikipedia hoaxes, in *International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva*, 2016, pp 591–602.
12. A. L. Traud, P. J. Mucha and M. A. Porter, *Phys. A: Statist. Mech. Appl.* **391**, 4165 (2012).
13. T. Martin, B. Ball and M. E. Newman, *Phys. Rev. E* **93**, 012306 (2016).
14. C. Von Mering, L. J. Jensen, B. Snel, S. D. Hooper, M. Krupp, M. Foglierini, N. Jouffre, M. A. Huynen and P. Bork, *Nucl. Acids Res. D* **33**, 433 (2005).
15. T. P. Peixoto and M. Rosvall, *Nature commun.* **8**, 582 (2017).
16. M. Girvan and M. E. Newman, *Proc. Natl. Acad. Sci.* **99**, 7821 (2002).
17. J. Shi and J. Malik, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, 888 (2000).
18. M. E. Newman, *Phys. Rev. E* **69**, 066133 (2004).
19. V. D. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre, *J. statist. Mech.: Theory Exp.* **2008**, P10008 (2008).
20. V. A. Traag, *Phys. Rev. E* **92**, 032801 (2015).
21. Z. Zhang, P. Pu, D. Han and M. Tang, *Phys. A: Statist. Mech. Appl.* **506**, 975 (2018).
22. M. Rosvall, A. V. Esquivel, A. Lancichinetti, J. D. West and R. Lambiotte, *Nature commun.* **5**, 4630 (2014).
23. Q. Zhou, S.-M. Cai and Y.-C. Zhang, *Int. J. Mod. Phys. C* **31**, 2050056 (2020).
24. A. Arenas, A. Díaz-Guilera and C. J. Pérez-Vicente, *Phys. Rev. Lett.* **96**, 114102 (2006).

25. Z. Zhuo, S.-M. Cai, M. Tang and Y.-C. Lai, *Chaos: An Interdiscip. J. Nonlinear Sci.* **28**, 043119 (2018).
26. U. N. Raghavan, R. Albert and S. Kumara, *Phys. Rev. E* **76**, 036106 (2007).
27. Z.-H. Deng, H.-H. Qiao, Q. Song and L. Gao, *Phys. A: Statist. Mech. Appl.* **519**, 217 (2019).
28. J. Han, W. Li, Z. Su, L. Zhao and W. Deng, *Eur. Phys. J. B* **89**, 272 (2016).
29. T. P. Peixoto, *Phys. Rev. X* **4**, 011047 (2014).
30. Q. Zhou, S. Cai and Y. Zhang, *IEEE Access* **7**, 171223 (2019).
31. F. Liu, D. Choi, L. Xie and K. Roeder, *Proc. Natl. Acad. Sci.* **115**, 927 (2018).
32. C. E. Priebe, Y. Park, J. T. Vogelstein, J. M. Conroy, V. Lyzinski, M. Tang, A. Athreya, J. Cape and E. Bridgeford, *Proc. Natl. Acad. Sci.* **116**, 5995 (2019).
33. S. Fortunato and M. Barthelemy, *Proc. Natl. Acad. Sci.* **104**, 36 (2007).
34. J. Shao, Z. Han, Q. Yang and T. Zhou, Community detection based on distance dynamics, in *Proc. 21th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining* (ACM, New York, 2015), pp. 1075–1084.
35. C. Hennig and B. Hausdorf, Design of dissimilarity measures: A new dissimilarity between species distribution areas, in *Data Science and Classification* (Springer, 2006), pp. 29–37.
36. A. Lancichinetti, S. Fortunato and F. Radicchi, *Phys. Rev. E* **78**, 046110 (2008).
37. A. Strehl and J. Ghosh, *J. Mach. Learn. Res.* **3**, 583 (2002).
38. M. E. Newman, *Proc. Natl. Acad. Sci.* **103**, 8577 (2006).
39. L. Chen, J. Zhang, L. Cai and Z. Deng, *Tsinghua Sci. Technol.* **22**, 564 (2017).
40. W. W. Zachary, *J. Anthropol. Res.* **33**, 452 (1977).
41. J. Yang and J. Leskovec, *Knowl. Inf. Syst.* **42**, 181 (2015).
42. J. Leskovec, J. Kleinberg and C. Faloutsos, *ACM Trans. Knowl. Discovery Data (TKDD)* **1**, 2 (2007).
43. E. Cho, S. A. Myers and J. Leskovec, Friendship and mobility: user movement in location-based social networks, in *Proc. 17th ACM SIGKDD Int. Conf. Knowledge discovery and data mining* (ACM, New York, 2011), pp. 1082–1090.
44. J. Leskovec, K. J. Lang, A. Dasgupta and M. W. Mahoney, *Int. Math.* **6**, 29 (2009).