Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 1485592, 14 pages
https://doi.org/10.1155/2021/1485592

Research Article

An Improved Louvain Algorithm for Community Detection

Jicun Zhang,[1,2] **Jiyou Fei**,[1] **Xueping Song**,[1] **and Jiawei Feng**[2]

[1]*School of Mechanical Engineering, Dalian Jiaotong University, Dalian, Liaoning 116028, China*
[2]*Neusoft Group (Dalian) Co., Ltd., Dalian, Liaoning 116085, China*

Correspondence should be addressed to Xueping Song; sxp0724@djtu.edu.cn

Received 12 September 2021; Revised 17 October 2021; Accepted 9 November 2021; Published 23 November 2021

Academic Editor: Xin Tian

Social network analysis has important research significance in sociology, business analysis, public security, and other fields. The traditional Louvain algorithm is a fast community detection algorithm with reliable results. The scale of complex networks is expanding larger all the time, and the efficiency of the Louvain algorithm will become lower. To improve the detection efficiency of large-scale networks, an improved Fast Louvain algorithm is proposed. The algorithm optimizes the iterative logic from the cyclic iteration to dynamic iteration, which speeds up the convergence speed and splits the local tree structure in the network. The split network is divided iteratively, then the tree structure is added to the partition results, and the results are optimized to reduce the computation. It has higher community aggregation, and the effect of community detection is improved. Through the experimental test of several groups of data, the Fast Louvain algorithm is superior to the traditional Louvain algorithm in partition effect and operation efficiency.

1. Introduction

Community detection is a significant research issue with respect to complex networks, and the goal of community detection is to discover the communities in networks [1]. A community in a network is a group of cohesive nodes having dense connections within the group and sparse connections with other groups [2]. Generally, nodes of the same community have the same or similar features due to their strong cohesiveness. Therefore, community detection plays a very important role in the analysis of complex networks. For instance, it helps in identifying and visualizing the internal structure of the network, detecting potentially useful information, and mining the relationships between individuals.

With the advancement of information technology, the scale of complex networks is expanding larger all the time. Social networks are a typical example. Social network analysis (SNA) abstracts complex social networks into graph structures with complex relationships. Many studies can be carried out based on complex social networks [3–7]. Some researchers [8,9] discovered the epidemic spreading mechanism by using the community structure analysis of complex networks. Kim et al. [10] dealt with clustering techniques applied to Twitter network and Twitter trend analysis. By collecting the real-time user-generated content related to voting, it can be used to foresee the election results [11]. Gou et al. [12] analyzed the evolution trend of international migration using a community detection algorithm.

In the past decade, different community detection methods have been discovered, including global community detection algorithms [13–15], local community detection algorithms [16], and overlapping community detection algorithms [17]. Local community detection is the process of finding the community where a specified node is located. Overlapping community detection refers to the process where a node may belong to more than one community and finds all of those communities. Global community detection is the process of finding all communities of a given graph. The main research goal of this paper is global community detection.

Shao et al. [18] proposed a community detection algorithm called Attractor, which automatically spots communities in a network by examining the changes of "distances" among nodes. Attractor can discover communities of arbitrary size. In large network environments, however, the

distance dynamics model requires very long computation times. For global network detection, the most effective is the Louvain algorithm [15], but for large-scale datasets, Louvain algorithm performance is also relatively low.

Although community detection in networks has been studied for many years, a high-speed and high-quality community detection algorithm is crucial in large-scale network environments. Our goal in this paper is to quickly detect the community structure of a large network using the Louvain algorithm. To do so, we improve the speed of community detection via optimization strategies. The main contributions of this paper are as follows:

(1) An improved algorithm based on Louvain is proposed. The algorithm optimizes the iterative logic from the cyclic iteration to dynamic iteration, which speeds up the convergence speed.

(2) Split the local tree structure in the network. The split network is divided iteratively, then the tree structure is added to the partition results, the results are optimized to reduce the computation; it has higher community aggregation, and the effect of community division is improved.

(3) Through the experimental test of three groups of small-scale datasets and six groups of large-scale datasets, the Fast Louvain algorithm is better than the state of the art in community detection effect and efficiency.

The rest of this paper is organized as follows. Section 2 briefly reviews the development of community detection algorithm. Section 3 provides a problem statement and a detailed description of the improved Louvain algorithm. In Section 4, we conduct a couple of experiments to evaluate the performance of the approach proposed in this paper. Finally, Section 5 provides some concluding remarks and outlines future research directions.

2. Related Work

The early global community detection algorithm is the GN algorithm based on the edge betweenness, which was proposed by Girvan and others of the University of Michigan. This is a splitting method which means that each community is gradually separated from the whole [19,20]. Newman put forward an indicator to evaluate the quality of community partition: modularity [21]; based on the modularity indicator, the Newman algorithm was an aggregation algorithm which means that the community was initially merged from the discrete nodes [22–24]. Clauset and others optimized the Newman algorithm to reduce the time complexity. Although the community partition based on the optimized modularity indicator can get better results, how to obtain the optimal modularity is still unsolved [25]. To further improve the accuracy of the algorithm, more partition algorithms have been proposed [26,27]. At the same time, for large-scale networks, the computational efficiency of the existing methods at that time is still not ideal. Blondel and others proposed the Louvain algorithm. The computational efficiency of the Louvain algorithm is better than other community detection algorithms on the premise of ensuring the partition results, and the Louvain algorithm can be parallelized [28].

2.1. GN Algorithm. GN algorithm used edge betweenness which refers to the number of the shortest paths passing through each edge in the network. The basic idea of the GN algorithm is to constantly delete the edges with the maximum edge betweenness relative to all nodes in the network, then recalculate the edge betweenness of all edges, and delete edges until all of them in the network are deleted. Hierarchical clustering can be obtained in this way. GN algorithm is built around the idea of using centrality indices to find community boundaries. It can get a good community detection effect.

GN algorithm has two problems. First, the algorithm lacks evaluation indexes. Although hierarchical clustering can be obtained by the GN algorithm, it cannot evaluate which hierarchy is the optimal community partition. Second, because of the complexity of calculating the edge betweenness, the time complexity of the algorithm is higher, which is close to $o(m^2n)$.m and n are, respectively, the number of nodes and edges in the network. Therefore, when faced with a substantial amount of data, the computational efficiency of the GN algorithm is very low.

2.2. Fast Newman Algorithm. The basic idea of the Fast Newman algorithm is to first define each node as a community, calculate the change value of modularity after merging two communities in each iteration, and then merge the two communities, which can get the maximum gain of the modularity until no other mergences of any two communities can increase the modularity. Then, the optimal partition result of the Fast Newman algorithm is worked out. Compared with the GN algorithm, the time complexity of the Fast Newman algorithm is lower and approximately to $o(m(m+n))$, so it is faster in computational efficiency.

However, when it comes to large-scale community partitioning, the data to be processed are often of a huge scale such as social platform data and regional call data. The relationships can be in the millions or tens of millions. In this case, even the time complexity of $o(m(m+n))$ is still too time-consuming when processing large-scale data.

2.3. Louvain Algorithm. Louvain algorithm is an efficient hierarchical clustering algorithm based on graph theory. Its principle is to make the modularity of community partition result reach the maximum value through continuous iteration of mobile nodes and then obtain the optimal community partition.

The main steps of the Louvain algorithm are as follows:

Step 1: Initialize the community and set each node as a separate community, namely, community 1: (node1), community 2: (node2), and so on.

Step 2: Find out all the communities connected to node 1, and calculate the change of modularity after moving node 2 to each neighbor community. Move node 1 to the community, which can increase the modularity to the maximum.

Step 3: Iterate over all the nodes and execute step 2 until there are no nodes to move and get a layer of community partition.

Step 4: Merge each community in step 3 into a new node. The relationship between new nodes is the relationship between the original communities. Return to step 1 until all nodes are finally merged into one community. The multilevel community partition is obtained, and the partition with the highest modularity is selected as the final partition result.

Compared with the Fast Newman algorithm, this algorithm has the advantage of fewer computation times per iteration. Meanwhile, when calculating the movement of a single node, only the modularity gain of the node in the two communities where the movement occurs needs to be calculated. So, the computational efficiency is obviously better than that of the Fast Newman algorithm and the time complexity of this algorithm is approximately to $o(m+n)$.

In recent years, the Louvain algorithm and improved algorithms based on it have been widely applied in large-scale complex network partition [29–36]. Perrin et al. [36] proposed a recursive method based on the Louvain algorithm for community detection and the PageRank algorithm for authoritativeness weighting in networks. The method is effective in identifying a large number of significant modules, but the number of nodes in a subcommunity can only be less than 100. Mohammadi et al. [29] proposed an adaptive CUDA Louvain method (ACLM) algorithm that benefits from the graphic processing unit (GPU). This method can be parallelized to improve the detection speed. However, due to the high price of GPU resources, it is difficult to be widely used.

These traditional Louvain-based algorithms have a common drawback: these algorithms do not reduce the number of iterations of the Louvain algorithm. Under the same hardware computing conditions, the computational efficiency of the Louvain algorithm is not improved. For community networks with uneven nodes, excessive aggregation is easy to occur.

3. Materials and Methods

Community detection algorithm is an unsupervised clustering algorithm. How to measure the quality of its detection results is an important problem.

For data sources with known results, evaluation can be conducted using a comparison between partition results and real results, calculation of accuracy, and so on. However, this kind of evaluation can only be a reference because each group of real data has its particularity while the unsupervised clustering algorithm partitions the network through more general characteristics. Therefore, good results obtained by using special data are not necessarily universal, and vice versa.

An ideal partition of community is supposed to have the following characteristics: (1) more edges within the community; (2) fewer edges between communities.

Many partition evaluation indexes are generated based on this thought, such as the fitness index for local partition and modularity for global partition. This paper is concerned with large-scale global community detection, so modularity will be used for evaluation.

The modularity, proposed by Newman of the University of Michigan [37], has been widely used as an indicator of the quality of the results; the calculation formula is as follows:

$$Q = \frac{1}{2m}\sum_{i,j}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta\left(C_i, C_j\right), \tag{1}$$

where k_i refers to the degree of node i; C_i is the community where node i is; m is the weighted sum of all edges in the community network; A_{ij} is the weight of the edges connected between node i and node j; $\delta(C_i, C_j)$ represents whether node i and node j are in the same community. If so, $\delta(C_i, C_j) = 1$; otherwise, $\delta(C_i, C_j) = 0$. Q is less than 1. When Q is greater than 0.3, the network is considered to have an obvious community structure; the closer Q is to 1, the more obvious community structure is.

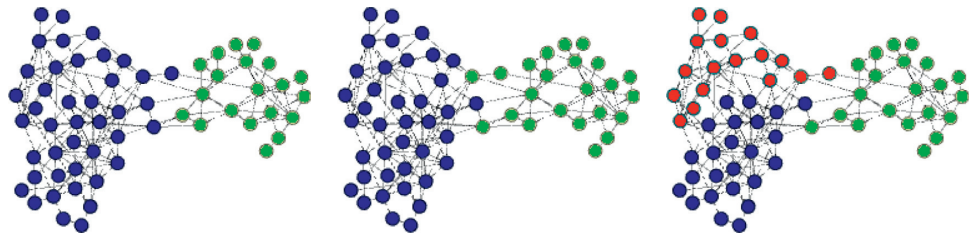
As can be seen from the formula of the modularity, the calculation is only aimed at the inner part of each community. When a certain node is moved, the change of modularity only occurs between the two communities where the nodes move. Therefore, in the process of the actual iteration, only the modularity increment and the difference need to be calculated, without the overall modularity before and after moving. The calculation formula for the modularity increment of a single node i for a certain community is as follows:

$$\begin{aligned}\Delta Q = &\left[\frac{\sum \text{in} + K_{i,\text{in}}}{2m} - \left(\frac{\sum \text{out} + k_i}{2m}\right)^2\right] \\ &- \left[\frac{\sum \text{in}}{2m} - \left(\frac{\sum \text{out}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right],\end{aligned} \tag{2}$$

where $\sum \text{in}$ is the sum of the weights of the internal edges of the community; $\sum \text{out}$ is the sum of the weights of all the edges connected to the community; k_i is the degree of the node i; $K_{i,\text{in}}$ is the sum of the weights of edges connecting node i with the community.

Figure 1 is the partition results of Louvain and Fast Newman algorithms on the bottlenose dolphin network dataset [38]. Louvain is not only superior to Fast Newman in speed but also in partition effect. In mining global community structure of complex networks, the time complexity is approximately linear. The Louvain algorithm also does better in the partition effect, and it is still the best choice at present.

As the era of big data approaches, the amount of data to be processed by the algorithm is increasing, and it becomes much more time-consuming. In fact, there is still room for optimization in operational efficiency. On the other hand, some problems still exist in the partition effect of the



(a) (b) (c)

Figure 1: The partition results of algorithms on the bottlenose dolphin network dataset. (a) Real result. (b) Result by Louvain. (c) Result by Fast Newman.

Louvain algorithm. When the density is not uniform, incomplete local aggregation or excessive aggregation may occur.

To solve these problems, an improved Fast Louvain algorithm is proposed in this paper, which has been tested and verified by multiple sets of public datasets. It is proved that the Fast Louvain algorithm has faster operation efficiency and better partition effect than the Louvain algorithm.

3.1. Fast Louvain Algorithm. Based on the Louvain algorithm, the Fast Louvain algorithm makes progress: (1) optimization of dynamic iteration; (2) split of the local tree structure.

The first one is mainly aimed at the efficiency optimization of algorithm iteration. And the second improvement can optimize the partition effect under the condition of uneven data density and improve the efficiency of algorithm iteration at the same time.

3.1.1. Dynamic Iterative Optimization. In the Louvain algorithm, all nodes and their neighboring communities need to be traversed several times in the clustering process of each layer. The modularity gain is calculated to determine whether to move the nodes until no nodes can be moved. Many repeated calculations are contained in this iteration.

When a single node moves, the change of modularity exists only in the two communities where the movement occurs. Therefore, when it comes to community A and community B existing in the network, if neither community A nor community B changed in the last iteration, a local stable structure will be formed between community A and community B before they change iteration.

Since the modularity gain of node movement between community A and community B has been calculated in the last iteration and no nodes can be moved, there will definitely be no node movement between community A and community B before the node changes in this iteration. It is called a local stable structure.

Since the calculation of modularity gain is relatively complicated, skipping directly when the algorithm traverses the local stable structure can save a lot of computation time. Especially at the later stage of the algorithm iteration, most of the nodes no longer could be moved. Compared with the Louvain algorithm, the Fast Louvain algorithm with dynamic iterative optimization can converge faster with the same result.

On the other hand, the stop condition of the Louvain algorithm for clustering at each layer is that no nodes can be moved. In fact, this condition can be relaxed. As mentioned above, most of the nodes belong to their communities in the postiteration stage of the algorithm, and only a few nodes are moving. Generally speaking, the modularity change of these nodes after moving is minor. In other words, the modularity gain of such nodes is similar in the community before and after the movement.

As shown in Figure 2, the modularity gain of the node among three neighboring communities is exactly the same. Even there is a slight change in the neighboring community, the community attribution of the node may change. However, the modularity gain generated by this change is very small. In the postiteration stage of the algorithm, the moves produced by each round of traversal are mostly such moves. Therefore, if a preset parameter, such as 0.1%, is specified, and the iteration is stopped when the number of moved nodes in a certain traversal is less than 0.1% of the total, the result should be similar to that in the original method. In this way, the traversal times of each layer of the cluster will be decreased, and the efficiency of the algorithm will be increased.

3.1.2. Splitting of Local Tree Structures. In complex social networks, there are always some leaf nodes that are only connected to one other node. Many previous improvements to the Louvain algorithm were based on leaf nodes. The leaf node can be predicted in the partition result of the algorithm because it will only be partitioned into the same community with the nodes connected to it. Therefore, removing the leaf nodes in the network before the algorithm iteration [39] can reduce the number of nodes participating in the iteration, thus improving the efficiency of the iteration.

In fact, after the leaf nodes in the network are removed, new leaf nodes may appear. If the new leaf nodes continue to be removed and this operation is repeated until no new leaf nodes are generated, three types of tree structures as shown in Figure 3 can be separated from the network.

As shown in Figure 4, the Fast Louvain algorithm first splits all local tree structures existing in the network before the iteration, divides the remaining network, and finally adds three tree structures to the partition result. In this way, the number of nodes participating in the iteration could be reduced, and the partition results could be optimized at the same time. Experiments show that this optimization method

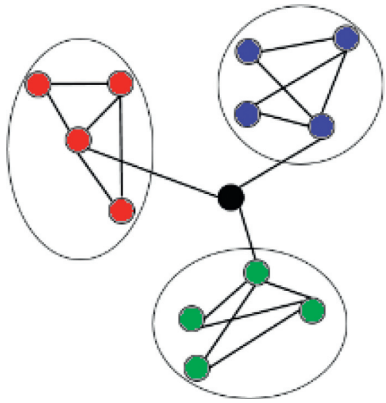


Figure 2: An example of a community belonging to a fuzzy node.

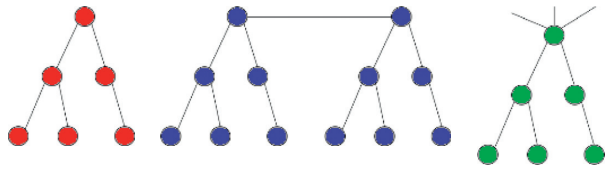


Figure 3: Three local tree structures: tree structure ①; tree structure ②; tree structure ③.

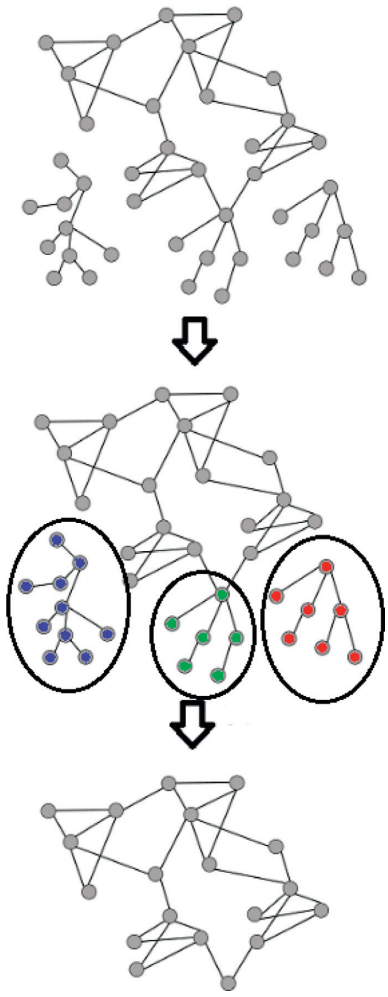


Figure 4: An example of tree splitting.

of splitting tree nodes can improve the algorithm and partition effect as expected.

Tree structures ① and ② are similar, and the difference lies only in whether the remaining fixed points are isolated nodes or two connected nodes in the process of cutting leaves many times. For these tree structures, the optimal processing scheme is to establish a separate network for each tree to divide and then put the divided community into the final result.

In fact, for these two types of tree structures, as an independent community or continuing to divide the community, its final modularity basically does not change.

Because when the algorithm does not converge in the first layer of cluster, even if the tree structure is divided in the first layer of the cluster, the divided communities will merge into a new leaf node in the next layer of the cluster and finally aggregate together again.

If the algorithm converges at the first layer, the difference in modularity caused by partitioning the tree structure is also very small.

$$Q_i = \frac{1}{2m} \sum_{j \in C_i} \left[A_{ij} - \frac{k_i k_j}{2m} \right]. \tag{3}$$

The modularity contribution of node i is shown above, where m is the sum of the total weights of the edges in the network; C_i is the community to which node i belongs; A_{ij} is the sum of weights of the edges between node i and node j; and k_i is the sum of weights of all edges connected to node i. Suppose that node i belongs to tree structure ① or ② and there are n edges inside the tree. Since all nodes in these two tree structures only have inner connected edges, $\sum_{j \in C_i} k_j = 2n$. On the other hand, node i is only connected to the nodes inside the tree, so $\sum_{j \in C_i} A_{ij} = k_i$. It follows that

$$\begin{aligned} Q_i &= \frac{1}{2m} \left(\left(\sum_{j \in C_i} A_{ij} \right) - \frac{k_i n}{m} \right) \\ &= \frac{1}{2m} \left(k_i - \frac{k_i n}{m} \right) \\ &= \frac{1}{2m} \left(1 - \frac{n}{m} \right) k_i. \end{aligned} \tag{4}$$

The overall modularity of the tree structure is

$$\begin{aligned} Q_{C_i} &= \sum_{i \in C_i} Q_i \\ &= \sum_{i \in C_i} \frac{1}{2m} \left(1 - \frac{n}{m} \right) k_i \\ &= \frac{1}{2m} \left(1 - \frac{n}{m} \right) \sum_{i \in C_i} k_i \\ &= \frac{n}{m} \left(1 - \frac{n}{m} \right). \end{aligned} \tag{5}$$

Suppose that the tree structure is divided into two parts, n_0 is the weight of edges to be divided, and the sum of weights of the edges of one part is n_1 so that of the other part of the two parts is $n_2 = n - n_0 - n_1$.

After the partition, the modularity of the two parts is, respectively,

$$\begin{aligned} Q_1 &= \frac{n_1}{m} \left(1 - \frac{(2n_1 + n_0)}{2m} \right), \\ Q_2 &= \frac{n_2}{m} \left(1 - \frac{(2n_2 + n_0)}{2m} \right). \end{aligned} \tag{6}$$

Suppose the change value of modularity is ΔQ ; then

$$\begin{aligned} \Delta Q &= Q_1 + Q_2 - Q_c \\ &= \frac{n_1}{m} \left(1 - \frac{(2n_1 + n_0)}{2m} \right) + \frac{n_2}{m} \left(1 - \frac{(2n_2 + n_0)}{2m} \right) - \frac{n}{m} \left(1 - \frac{n}{m} \right) \\ &= -\frac{\left(2n_1^2 + 2n_2^2 + n_1 n_0 + n_2 n_0 - 2n^2 \right)}{2m^2} - \frac{n_0}{m} \\ &= -\frac{\left(2n_1^2 + 2n_2^2 + n_1 n_0 + n_2 n_0 - 2\left(n_1 + n_2 + n_0 \right)^2 \right)}{2m^2} - \frac{n_0}{m} \\ &= -\frac{\left(-3n_1 n_0 - 3n_2 n_0 - 4n_1 n_2 \right)}{2m^2} - \frac{n_0}{m} \\ &= \frac{3n_1 n_0 + 3n_2 n_0 + 4n_1 n_2 - 2mn_0}{2m^2}. \end{aligned} \tag{7}$$

Since ΔQ is much greater than n_1, n_2, and n_0, generally speaking, ΔQ is negative. Only when n_0 is small enough and n_1 and n_2 are large enough, ΔQ can be positive. If ΔQ is obviously large, then n_1 and n_2 must be very large. At that time, the proportion of the tree in the network will be large, and the network structure is quite sparse with limited value to partition. Therefore, the loss of modularity caused by not dividing the tree structure is minor; thus, tree structures ① and ② can be directly added to the final result without partition.

Compared with tree structures ① and ②, tree structure ③ has its own particularity. Tree structures ① and ② exist in isolation, while the fixed point of tree structure ③ is connected to the inside of the network. Therefore, the analysis of tree structure ③ is divided into two parts: the processing of tree structure ③ itself and the influence of splitting it.

The first is how to deal with tree structure ③. Like the situation of tree structures ① and ②, it does not matter too much for the modularity of overall partition whether the interior of tree structure ③ is divided. On the other hand, suppose that all nodes of tree structure ③ are added to the community, which is connected with vertices directly after the partition.

Since the target community may be compact, when there are too many nodes in tree structure ③, the modularity will decrease after adding all the nodes into the neighboring community. At that time, a smaller threshold can be set. When the number of nodes of tree structure ③ exceeds this threshold, it will be added to the result as an independent community; otherwise, all the nodes will be added to the community connected with vertices.

On the other hand, the separation of tree structure ③ will affect the overall partition, which is generally positive. It is easy to think that the reason why the convergence speed will slow down due to tree structure ③ is that the network part connected to tree structure ③ is relatively sparse. For example, suppose that a certain network converges in the k layer of cluster. If tree structure ③ is added to the relatively sparse part in the network, then the part connected to tree structure ③ may need $k + 1$ layers of the cluster to realize the convergence, while other parts have been clustered too much and the result of algorithm could not be positive. In addition, the clustering effect will not be worse after the partition of tree structure ③. The partition only leads to faster aggregation of relatively sparse parts connected with tree structure ③ while the aggregation speed of other compact parts is obviously higher than that of the sparse parts.

Based on the above inference, the aggregation degree after the partition of tree structure ③ should be higher, and the number of communities should be less. In fact, tests based on multiple open-source datasets have proved this thought.

4. Results and Discussion

In this chapter, three groups of small open-source datasets and six groups of large-scale open-source datasets will be selected for testing. These open-source datasets are representative. They are classic datasets used for community detection experiments.

The details of the small datasets are shown in Table 1. Zachary's karate club network [40] is a social network of friendships between 34 members of a karate club at a US university in the 1970s. It has been used extensively as a benchmark for community detection. The Dolphins Social Network [27] reported by Lusseau contains 62 dolphins. A link is introduced to connect two dolphins or nodes if the two are observed to be together more often than expected by chance over a period of seven years from 1994 to 2001. The dolphins are mainly divided into male dolphins and female dolphins; therefore, two communities exist in the network. Political books network is a network of books about US politics published around the time of the 2004 presidential election and sold by the online bookseller Amazon.com. Edges between books represent frequent copurchasing of books by the same buyers. The books can be divided into three communities: liberal, conservative, and neutral.

For large-scale datasets, the "Stanford Large Network Dataset Collection" (SNAP:https://snap.stanford.edu) is selected because it contains large graphs used as benchmarks to test the scalability of the heuristics. The datasets include social networks, web graphs, Wikipedia networks, Internet networks, and citation networks. We select six datasets for testing. The details of the data are shown in Table 2.

The small-scale datasets have real community results, and we can compare the community detection results with the real result. The closer the algorithm is to the real result, the better the effect of the algorithm is. The large-scale datasets have no real results, and we judge the quality of the algorithm by comparing the modularity. The greater the modularity, the better the quality of the algorithm. We also compare the calculation time of the algorithm. The less the time, the better the performance of the algorithm.

The hardware configuration of the testing environment is shown in Table 3. All algorithms are implemented in Python 3.7. The value of the parameter to stop iteration in the Fast Louvain algorithm is 0.001, and the test is operated in a single-threaded way.

Due to the high complexity of the algorithm, we do not use the Fast Newman method for testing under large-scale datasets because it will consume too much time. For large-scale datasets without real communities, the partition result of the Fast Louvain algorithm is compared with that of the Louvain algorithm according to previous inference:

(1) In most cases, the modularity of the results of the Fast Louvain algorithm is much higher than that of the Louvain algorithm. But sometimes it is lower with little differences.

(2) Fast Louvain algorithm should take less time than the Louvain algorithm.

(3) In the partition results of the Fast Louvain algorithm, the compact parts should be close to the partition results of the Louvain algorithm, while the sparse parts should be more aggregated.

The first and second inferences can be proved by corresponding indicators. For the third inference, 10 communities with the highest average node modularity in the partition results of two algorithms are selected. If the average modularity of the two algorithms is close, then the compact parts of the two partition results are close. In the meanwhile, if the number of communities in the partitioning results of the Fast Louvain algorithm is relatively less, then it is proved that the sparse parts of the result are more aggregated.

4.1. Small-Scale Datasets

4.1.1. Friendship Network of Zachary's Karate Club. The Friendship Network of Zachary's Karate Club consists of 34 nodes and 78 edges. The actual data are divided into two communities. The comparison result of the three algorithms is in Table 4. The speed of Fast Louvain is faster than Louvain and Fast Newman. The partition results are shown in Figure 5. The Louvain and Fast Louvain algorithms get the same result as the real community, and the Fast Newman algorithm has a 26% error rate. The experimental results show that, for Friendship Network, the effects of the Louvain and Fast Louvain algorithms are similar and better than the Fast Newman algorithm.

4.1.2. Bottlenose Dolphin Network. Bottlenose dolphin network consists of 62 nodes and 159 edges. The actual data are divided into two communities. The partition results of the Louvain and Fast Louvain algorithm are the same with 3 node partition errors, and the Fast Newman algorithm gets 18 nodes partition errors. For the algorithm speed, the Fast Newman algorithm is slower than the Louvain and Fast

Table 1: Small-scale dataset.

Dataset	Number of nodes (V)	Number of edges (E)	$2E/V$
Karate club	34	78	4.59
Bottlenose dolphin network	62	159	5.13
Books network about US politics	105	441	8.40

Table 2: Large-scale dataset.

Dataset	Number of nodes (V)	Number of edges (E)	$2E/V$
Pokec	1632803	30622564	37.51
Facebook	22470	171002	15.22
Google	875713	5105039	11.66
Wikipedia	2394385	5021410	4.19
Stanford	281903	2312497	16.41
Berkeley-Stanford	685230	7600595	22.18

Table 3: Parameters of testing machine.

CPU	Inter(R) Core(TM) i7-2600 CPU @3.40 GHz
Memory	16 GB

Table 4: Result contrast of Friendship Network of Zachary's Karate Club.

	Louvain	Fast Louvain	Fast Newman
Errors	0	0	9
Error rate	0	0	26%
Time (ms)	6	3	34

Louvain algorithms. The test result of the algorithm is in Table 5 and Figure 6. The experimental results show that, for the bottlenose dolphin network, the effects of Louvain and Fast Louvain are similar and better than those of the Fast Newman algorithm.

4.1.3. Books Network about US Politics. Books network about US politics contains 105 nodes and 441 edges. The actual data are divided into 3 communities. The test result is shown in Table 6 and Figure 7. The partition results of the Louvain algorithm and Fast Louvain algorithm are the same with 16 node partition errors. The Fast Newman algorithm gets 18 node partition errors. For the algorithm speed, the Fast Newman algorithm is significantly slower than the Louvain and Fast Louvain algorithms. The experimental results show that, for books network about US politics, the effects of Louvain and Fast Louvain are similar and better than the Fast Newman algorithm.

4.2. Large-Scale Datasets

4.2.1. Pokec Social Network. Pokec is the most popular online social network in Slovakia. The popularity of networks has not changed even after the coming of Facebook. Pokec has been provided for more than 10 years and connects more than 1.6 million people.

The dataset [41] contains anonymous data of the whole network, including gender, age, hobbies, interests, and education. In this paper, the relationship of friends in the data is selected to build the network, which contains 1,632,803 nodes and 30,622,564 edges. The test result is in Table 7.

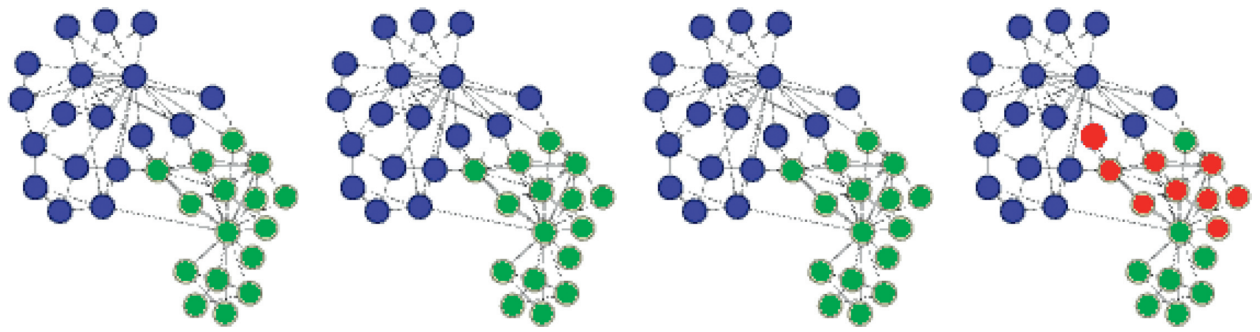
It can be seen that the modularity of the Fast Louvain algorithm has only decreased by 0.04%, the number of communities is less than that of the Louvain algorithm, and the time is reduced by 45%. On the other hand, 10 communities with the highest average modularity of nodes in the partition results of two algorithms are selected. In the Fast Louvain algorithm, 5 communities are exactly the same as the Louvain algorithm, and the remaining 5 communities are more than 90% similar to that of the Louvain algorithm. This is consistent with the previous inference.

4.2.2. Facebook Large Page-Page Network. The Facebook Large Page-Page Network dataset [42] is a page graph of verified Facebook sites. This graph was collected through Facebook Graph API in November 2017 and restricted to 4 categories of pages defined by Facebook. These categories are politicians, governmental organizations, television shows, and companies. Nodes represent official Facebook pages, while links are mutual fondness between sites. The network consists of 22,470 nodes and 171,002 edges. The test result is in Table 8.

The modularity of the Fast Louvain algorithm has increased by 0.96%, the number of communities is also less than that of the Louvain algorithm, and the time is reduced by 7%. Among the ten communities with the highest average modularity of nodes in the partition results of the two algorithms, 2 communities are the same, 7 communities have similar members (the difference is less than 10%), and 1 community is completely different. After further investigation, it shows that the totally different communities have their corresponding communities with the difference of less than 10%. This result is still in line with the previous inference.

4.2.3. Google Web Graph. The Google Web Graph dataset [43] was released by Google in 2002 as a part of Google Programming Contest. Nodes represent web pages, and directed edges represent hyperlinks between them. This network contains 875,713 nodes and 5,105,039 edges. The test result is shown in Table 9.

Similarly, the modularity of the Fast Louvain algorithm has increased by 0.14%, the number of communities is less than that of the Louvain algorithm, and the time is reduced by 84%. Ten communities with the highest average modularity of nodes in the partition results of two algorithms are the same.

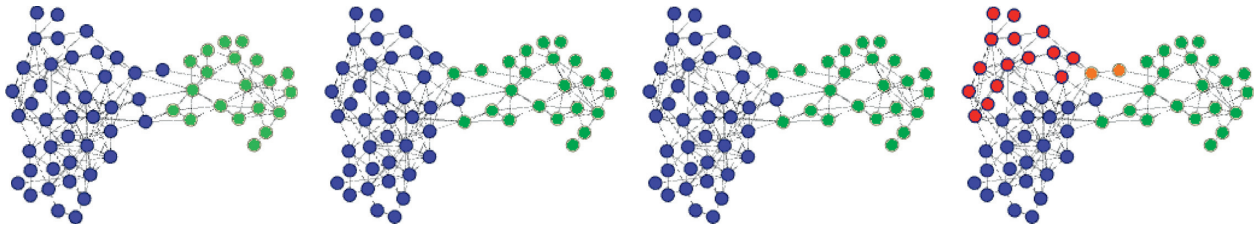


(a) (b) (c) (d)

Figure 5: The test result of the algorithms. (a) Real result. (b) Result by Louvain. (c) Result by Fast Louvain. (d) Result by Fast Newman.

Table 5: Result contrast of bottlenose dolphin network.

	Louvain	Fast Louvain	Fast Newman
Errors	3	3	18
Error rate	5%	5%	29%
Time (ms)	9	8	187

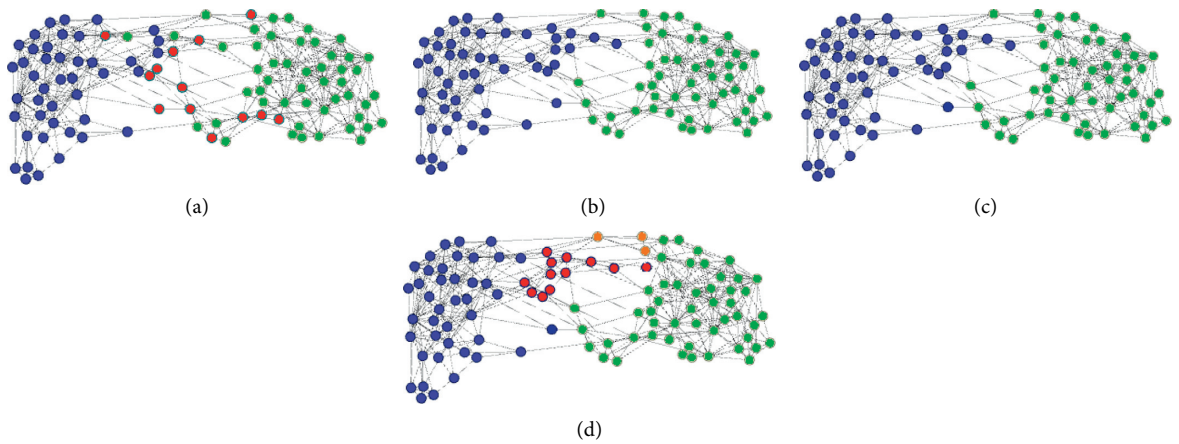


(a) (b) (c) (d)

Figure 6: The test result of the bottlenose dolphin network. (a) Real result. (b) Result by Louvain. (c) Result by Fast Louvain. (d) Result by Fast Newman.

Table 6: Result contrast of books network about US politics.

	Louvain	Fast Louvain	Fast Newman
Errors	16	16	18
Error rate	15%	15%	17%
Time (ms)	15	13	1203



(a) (b) (c) (d)

Figure 7: The test result of the algorithms. (a) Real result. (b) Result by Louvain. (c) Result by Fast Louvain. (d) Result by Fast Newman.

Table 7: Result contrast of Pokec social network.

	Modularity	Number of communities	Time (ms)
Louvain	0.682284	17070	9725441
Fast Louvain	0.681980	10978	5310240

Table 8: Result contrast of Facebook large page-page network.

	Modularity	Number of communities	Time (ms)
Louvain	0.757628	1610	7553
Fast Louvain	0.764911	1133	7017

Table 9: Result contrast of Google Web Graph.

	Modularity	Number of communities	Time (ms)
Louvain	0.972471	7911	2114998
Fast Louvain	0.973826	5816	332378

4.2.4. Wikipedia Talk Network. Wikipedia is a free encyclopedia written collaboratively by volunteers around the world. Each registered user has a talk page that he/she and other users can edit in order to communicate and discuss updates of various articles on Wikipedia. It uses the latest complete dump of edit history of Wikipedia page (from January 3, 2008). The dataset [44,45] contains all the users and discussion from the inception of Wikipedia till January 2008. Nodes in the network represent Wikipedia users, and a directed edge from node i to node j represents that user i at least once edited a talk page of user j. This network contains 2,394,385 nodes and 5,021,410 edges. The test result is shown in Table 10.

Similarly, the modularity of the Louvain algorithm has increased by 0.12%, the number of communities is less than the Louvain algorithm, and the time is reduced by 80%. Among the ten communities with the highest average modularity of nodes in the partition results of two algorithms, 6 communities are the same and 1 community has similar members (the difference is less than 10%). The remaining 3 communities of Fast Louvain are composed of some communities obtained by the Louvain algorithm; that is, in the Fast Louvain algorithm, these communities are aggregated, which conforms to previous inference.

4.2.5. Stanford Web Graph. Nodes of the Stanford Web Graph dataset [43] represent pages, and edges represent hyperlinks between webpages. It consists of two datasets.

Dataset 1 is all pages of Stanford University (stanford.edu) and hyperlinks between pages. It contains 281,903 nodes and 2,312,497 edges. Dataset 2 is all pages of Stanford University (stanford.edu) and Berkeley University (berkeley.edu) and hyperlinks between pages. It contains 685,230 nodes and 7,600,595 edges. The test result is shown in Table 11.

The modularity of the Fast Louvain algorithm has increased by 0.22%, the number of communities is less than that of the Louvain algorithm, and the time is reduced by 80%. The test result is shown in Table 12. Among the ten communities with the highest average modularity of nodes in the partition results of two algorithms, 9 communities are exactly the same, and 1 community has similar members (the difference is less than 10%).

Table 10: Result contrast of Wikipedia talk network.

	Modularity	Number of communities	Time (ms)
Louvain	0.541836	24818	9338877
Fast Louvain	0.542506	17307	1892466

Table 11: Result contrast of Stanford Web Graph (dataset 1).

	Modularity	Number of communities	Time (ms)
Louvain	0.899198	4252	6387420
Fast Louvain	0.901210	2083	1294824

Table 12: Result contrast of Stanford Web Graph (dataset 2).

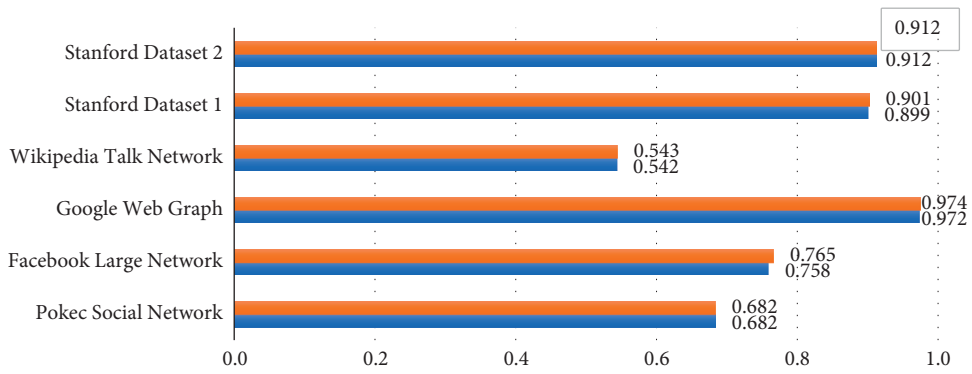
	Modularity	Number of communities	Time (ms)
Louvain	0.911988	8190	11843216
Fast Louvain	0.912073	5244	5527583

The modularity of the Fast Louvain algorithm has increased by 0.009%, the number of communities is less than that of the Louvain algorithm, and the time is reduced by 53%. Among the ten communities with the highest average modularity of nodes in the partition results of two algorithms, 7 communities are exactly the same and 2 communities have similar members (the difference is less than 10%). The remaining 1 community of Fast Louvain is composed of a part of the community obtained by the Louvain algorithm.

As for these two datasets, the performance of the improved method still conforms to the previous inference.

4.2.6. Comprehensive Analysis. The effects of Louvain and Fast Louvain are similar and better than the Fast Newman algorithm in small-scale datasets. This is because the number and depths of tree structures in these small-scale datasets are small and shallow; splitting tree structures will not affect the partition results. For the algorithm speed, the Fast Newman algorithm is significantly slower than Louvain and Fast Louvain. The time taken for Louvain and Fast Louvain algorithm is at milliseconds level, so the time taken for the two algorithms is similar.

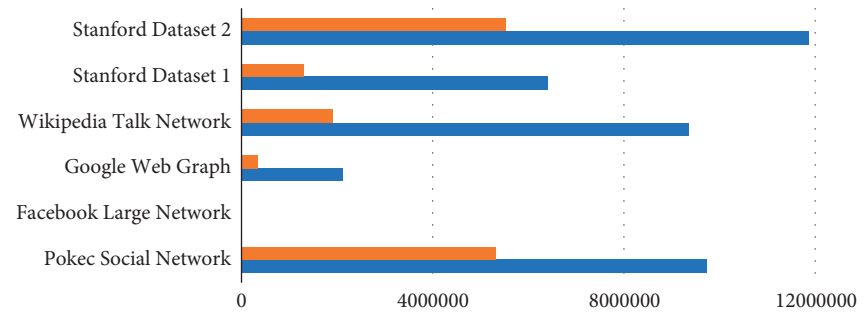
On the other hand, the modularity and performance of two algorithms in large-scale datasets are shown in Figures 8 and 9. The Fast Louvain algorithm has an obvious advantage. As for the modularity, except that the Fast Louvain algorithm is a little lower than the Louvain algorithm (by 0.04%) in Pokec Network, the modularity of Fast Louvain is higher than that of the Louvain algorithm in other datasets. In the meantime, the Fast Louvain algorithm takes much less time than the Louvain algorithm. The time-consuming comparison is shown in Table 13. The smallest time reduction is Facebook Large Network dataset, which is reduced by 7.1%. The biggest time reduction is Google Web Graph dataset, which is reduced by over 84%.



	Pokec Social Network	Facebook Large Network	Google Web Graph	Wikipedia Talk Network	Stanford Dataset 1	Stanford Dataset 2
fast Louvain	0.682	0.765	0.974	0.543	0.901	0.912
Louvain	0.682	0.758	0.972	0.542	0.899	0.912

Modularity

Figure 8: Modularity comparison between Fast Louvain and Louvain in large-scale datasets.



	Pokec Social Network	Facebook Large Network	Google Web Graph	Wikipedia Talk Network	Stanford Dataset 1	Stanford Dataset 2
fast Louvain	5310240	7017	332378	1892466	1294824	5527583
Louvain	9725441	7553	2114998	9338877	6387420	11843216

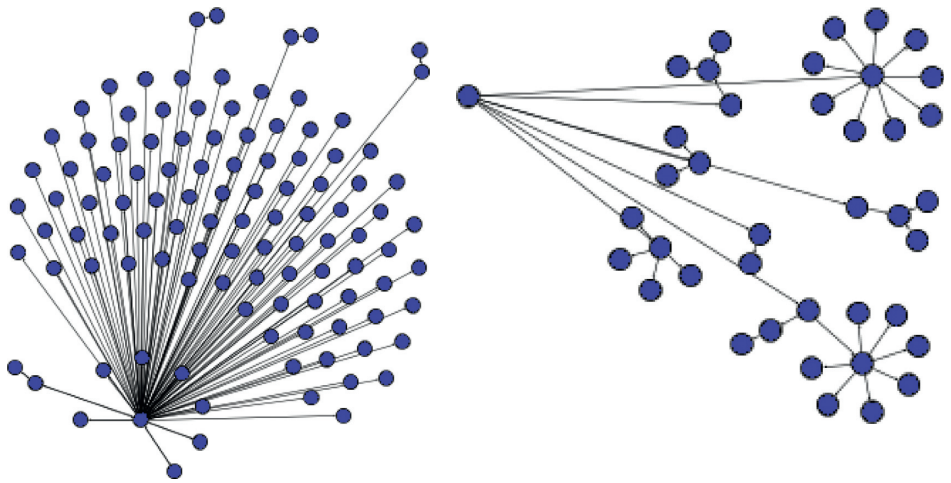
Time consuming [ms]

Figure 9: Time-consuming comparison between Fast Louvain and Louvain in large-scale datasets.

Table 13: Time-consuming comparison of Louvain and Fast Louvain for large-scale datasets.

Datasets	Louvain (ms)	Fast Louvain (ms)	Time reduction (%)
Pokec social network	9725441	5310240	45.40
Facebook Large Network	7553	7017	7.10
Google Web Graph	2114998	332378	84.28
Wikipedia talk network	9338877	1892466	79.74
Stanford dataset 1	6387420	1294824	79.73
Stanford dataset 2	11843216	5527583	53.33

In summary, the Fast Louvain algorithm performs better than the Louvain algorithm in large-scale datasets. This is because large-scale datasets have a large amount of data and often have many large local tree structures as shown in Figure 10, so the Fast Louvain algorithm has obvious advantages. In conclusion, the performance of the Fast Louvain algorithm is obviously better than the Louvain algorithm.



(a) (b)

Figure 10: A local tree structure. (a) Pokec dataset. (b) Google dataset.

5. Conclusions

The community structure detection issue in large networks has been widely investigated during past years. Although the Louvain algorithm is an effective community detection method, the detection efficiency decreases with the increase in the amount of data. In order to maintain high modularity and consume less time, an improved algorithm based on the Louvain algorithm is proposed in this paper.

(1) Before the iteration, the Fast Louvain algorithm firstly splits all the local tree structures existing in the network and then divides the remaining nodes of the network into iterative partitions.

(2) In the iteration process, the nodes participating in the iteration are dynamically adjusted according to the results of each iteration.

(3) At the end of each iteration, when a locally stable structure is formed and the modularity gain caused by the change of community ownership of a node is very small, then the iteration is ended by preset stopping parameters.

The above three methods greatly reduce the number of operations required for convergence. On the other hand, the split operation of the tree structure can make the partition result more uniform and avoid excessive aggregation. Therefore, Fast Louvain can get better partition results and consume less time compared with the traditional Louvain algorithm.

We select three groups of small-scale open-source datasets with real communities and six groups of large-scale open-source datasets without real communities for testing. The experimental results showed that, for the small datasets with real grouping results, the efficiency and accuracy of the Fast Louvain algorithm are basically consistent with that of the Louvain algorithm, but it is better than the Fast Newman algorithm; for the large-scale datasets, the average modularity of the Fast Louvain algorithm is slightly higher than that of the traditional Louvain algorithm, and the biggest time reduction is over 84% than the Louvain algorithm.

Therefore, Fast Louvain can get better partition results and consume less time than the traditional Louvain algorithm. This means that our proposed Fast Louvain algorithm has a better effect and performance in the field of global community detection than the state-of-the-art algorithms. In the future, we will do more research to further improve the modularity of large-scale datasets.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 51605069).

References

[1] L. Chaudhary and B. Singh, "Community detection using an enhanced louvain method in complex networks," in *Proceedings of the International Conference on Distributed Computing and Internet Technology*, pp. 243–250, Springer, Bhubaneswar, India, January 2019.

[2] N. Barbieri, F. Bonchi, and G. Manco, "Cascade-based community detection," in *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, pp. 33–42, Rome, Italy, February 2013.

[3] M. Coscia, F. Giannotti, and D. Pedreschi, "A classification for community discovery methods in complex networks," *Statistical Analysis and Data Mining*, vol. 4, no. 5, pp. 512–546, 2011.

[4] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.
[5] M. E. J. Newman, "Communities, modules and large-scale structure in networks," *Nature Physics*, vol. 8, no. 1, pp. 25–31, 2012.
[6] J. Zeng and H. Yu, "A study of graph partitioning schemes for parallel graph community detection," *Parallel Computing*, vol. 58, pp. 131–139, 2016.
[7] E. Rubio, O. Castillo, F. Valdez, P. Melin, C. I. Gonzalez, and G. Martinez, "An extension of the fuzzy possibilistic clustering algorithm using type-2 fuzzy logic techniques," *Advances in Fuzzy Systems*, vol. 2017, Article ID 7094046, 23 pages, 2017.
[8] Z. Liu and B. Hu, "Epidemic spreading in community networks," *Europhysics Letters*, vol. 72, no. 2, pp. 315–321, 2005.
[9] G. Ren and X. Wang, "Epidemic spreading in time-varying community networks," *Chaos*, vol. 24, no. 2, Article ID 068701, 2014.
[10] Y.-H. Kim, S. Seo, Y.-H. Ha, S. Lim, and Y. Yoon, "Two applications of clustering techniques to twitter: community detection and issue extraction," *Discrete Dynamics in Nature and Society*, vol. 2013, Article ID 903765, 8 pages, 2013.
[11] H. Khan, S. Nasir, K. Nasim, D. Shabbir, and A. Mahmood, "Twitter trends: a ranking algorithm analysis on real time data," *Expert Systems with Applications*, vol. 164, Article ID 113990, 2021.
[12] W. Gou, S. Huang, Q. Chen, J. Chen, and X. Li, "Structure and dynamic of global population migration network," *Complexity*, vol. 2020, Article ID 4359023, 17 pages, 2020.
[13] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.
[14] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 69, no. 6 Pt 2, Article ID 066133, 2004.
[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, pp. 1–12, 2008.
[16] C. Ali and A. Al Akhrass, "Supervised local community detection algorithm," *International Journal of Data Science*, vol. 5, no. 3, pp. 247–261, 2021.
[17] F. L. Huang and N. F. Xiao, "Discovering overlapping communities based on line graph and PSO," *Acta Automatica Sinica*, vol. 37, no. 9, pp. 1140–1144, 2011.
[18] J. Shao, Z. Han, Q. Yang, and T. Zhou, "Community detection based on distance dynamics," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1075–1084, Sydney, Australia, August 2015.
[19] R. G. Gayathri, J. J. Nair, and M. R. Kaimal, "Extending full transitive closure to rank removable edges in GN algorithm," *Procedia Computer Science*, vol. 93, pp. 995–1002, 2016.
[20] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the National Academy of Sciences*, vol. 101, no. 9, pp. 2658–2663, 2004.
[21] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 69, no. 2 Pt 2, Article ID 026113, 2004.
[22] F. Wu and B. A. Huberman, "Finding communities in linear time: a physics approach," *The European Physical Journal B (EPJ B) - Condensed Matter*, vol. 38, no. 2, pp. 331–338, 2004.
[23] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 74, no. 3 Pt 2, Article ID 036104, 2006.
[24] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 70, no. 6, Article ID 066111, 2004.
[25] U. Brandes, D. Delling, M. Gaertler et al., "Maximizing modularity is hard," 2006, https://arxiv.org/abs/physics/0608255.
[26] D. Li, L. Ma, Q. Yu, Y. Zhao, and Y. Mao, "An community detection algorithm based on the multi-attribute similarity," *International Conference on Communications IEEE*, vol. 1, pp. 118–121, 2013.
[27] M. A. Jie-Liang, Z.-zhen Pan, Lu Han, and Y. Song, "Community partition algorithm based on local consistency," *Science Technology and Engineering*, vol. 14, no. 27, pp. 135–139, 2014.
[28] S. Ghosh, M. Halappanavar, A. Tumeo et al., "Distributed louvain algorithm for graph community detection," in *Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS) IEEE Computer Society*, pp. 885–895, Vancouver, BC, Canada, May 2018.
[29] M. Mohammadi, M. Fazlali, and M. Hosseinzadeh, "Accelerating Louvain community detection algorithm on graphic processing unit," *The Journal of Supercomputing*, vol. 77, no. 6, pp. 6056–6077, 2021.
[30] X. Zhang, L. Wu, Z. Yao, and S. Yu, "A multi-layer network topology visualization layout based on louvain community detection," in *Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 760–763, IEEE, Guangzhou, China, June 2018.
[31] M. Seifikar, S. Farzi, and M. Barati, "C-blondel: an efficient louvain-based dynamic community detection algorithm," *IEEE Transactions on Computational Social Systems*, vol. 7, pp. 1–11, 2020.
[32] Z. Zhang, P. Pu, D. Han, and M. Tang, "Self-adaptive Louvain algorithm: fast and stable community detection algorithm based on the principle of small probability event," *Physica A: Statistical Mechanics and Its Applications*, vol. 506, pp. 975–986, 2018.
[33] A. K. Bhowmick, K. Meneni, M. Danisch, J.-L. Guillaume, and B. Mitra, "LouvainNE: hierarchical louvain method for high quality and scalable network embedding," in *Proceedings of the WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining ACM*, pp. 43–51, Houston TX USA, February 2020.
[34] M. Li, J. Qin, T. Jiang, and W. Pedrycz, "Dynamic relationship network analysis based on louvain algorithm for large-scale group decision making," *International Journal of Computational Intelligence Systems*, vol. 14, no. 1, pp. 1242–1255, 2021.
[35] J. Zeng and Yu. Hongfeng, "A scalable distributed louvain algorithm for large-scale graph community detection," in *Proceedings of the 2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 268–278, IEEE, Belfast, UK, September 2018.
[36] D. Perrin and G. Zuccon, "Recursive module extraction using Louvain and PageRank," *F1000Research*, vol. 7, p. 1286, 2018.
[37] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.
[38] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of

Doubtful Sound features a large proportion of long-lasting associations," *Behavioral Ecology and Sociobiology*, vol. 54, no. 4, pp. 396–405, 2003.

[39] Z. F. Wu, P.-fei Wang, Z.-guang Qin, and S.-quan Jiang, "Improved algorithm of Louvain communities dipartition," *Journal of University of Electronic Science and Technology of China*, vol. 42, no. 1, pp. 105–108, 2013.

[40] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, no. 4, pp. 452–473, 1977.

[41] R. Y. Dougnon, P. Fournier-Viger, J. C.-W. Lin, and R. Nkambou, "Inferring social network user profiles using a partial social graph," *Journal of Intelligent Information Systems*, vol. 47, no. 2, pp. 313–344, 2016.

[42] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *Journal of Complex Networks*, vol. 9, no. 2, Article ID cnab014, 2021.

[43] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, pp. 29–123, 2009.

[44] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," *Human Factors in Computing Systems*, pp. 1361–1370, 2010.

[45] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proceedings of the 19th international conference on World wide web*, pp. 641–650, Raleigh North Carolina USA, April 2010.