A Heuristic Algorithm for Bipartite Community Detection in Social Networks

Chengying Mao
School of Software and Communication Engineering,
Jiangxi University of Finance and Economics, 330013 Nanchang, China
Email: maochy@yeah.net

***Abstract*—Analysis on topological characteristics of the network, such as the vertex degree distribution, centrality and community structure, provides valuable insight into the structure and function of the interacting data entities. Community detection is one of the key problems in the field of social network analysis. In the paper, we mainly focus on the two-part division problem for network, i.e., community (or graph) partitioning. Based on the in-depth analysis on the community features of some known partitioning results, a three-stage heuristic algorithm named 3SHP is proposed. At first, two pseudo-centers are identified according to the clue of the diameter path in a network. Then, two heuristic strategies, i.e. the shortest path cutting (SPC) and two-point diffusing (TPD), are introduced to divide the whole network into three parts: two rudimental communities and a set of undecided nodes. Subsequently, an experience rule is used to classify such undecided nodes to produce two final communities. The experiment results show that the 3SHP algorithm is effective, able to yield the best partitioning results in most instances.**

***Index Terms*—heuristic method, graph algorithm, algorithm design and analysis, social networks**

I. Introduction

Data-intensive applications have been widely used in the era of information explosion. The rational way to model and analyze the massive data sets with complex interactions is network representation. The resulting graph-based structures are usually quite complex. Consequently, it is necessary to analyze the network structure and reveal the potential laws in the social relationships, as these laws are not usually readily discernible. That is the so-called social network analysis [1,2], which has emerged as a key technique in modern sociology.

In a social network, the nodes are not uniformly distributed but clustering together into some small groups. Therefore, the network not only consists of a mass of nodes but also has some subgroup structure. Research on partitioning a network into different subgroups called the community structures has become one of the major concerns of social network analysis. Accordingly, the process of identifying the clustering structures is called network community detection [2,3] The identification of community structure is beneficial to network reduction, scientific management, precise classification and so forth. As a consequence, community detection (or partitioning) has become an important research direction in the fields of system science and social science [4].

Community detection is essentially a NP-hard problem [5,6]. Therefore, nearly all existing algorithms are approximate, and failing to settle the problem fundamentally at all. Some algorithms such as K-L [7] can't achieve good partitioning result despite the short computing time. On the other hand, the algorithms such as GN [9] and L-Shell [11] need lots of time to produce high-quality division. In the paper, we mainly focus on the two-part division problem for social networks, and present a three-stage heuristic partitioning algorithm named 3SHP to solve it. In addition, we also performed some experimental investigations on three real-world networks to validate the effectiveness and efficiency of the proposed algorithm.

The remainder of this paper is organized as follows. The problem of detecting bipartite community is described in section 2. In section 3, the overall algorithm framework is addressed. Detailed implementation steps of the proposed algorithm are discussed in section 4, and its computational complexity is analyzed in section 5. In section 6, three real-world networks are used to validate the efficiency of our algorithm. Some existing work closely related to our method is addressed in section 7. Finally, we conclude the work in section 8.

II. Problem Description

The interaction data set is expressed using a graph abstraction $G(V, E)$, where V is the set of vertices representing unique interacting entities, and E is the set of edges representing the interactions. The number of vertices and edges are denoted by n and m respectively [2]. The graph may be directed or undirected, but we only consider the latter in this paper.

Given a network (referring to a connected and undirected graph here) $G=(V, E)$, $C=(C_1, C_2, \ldots, C_s)$ denote a partition of V such that $C_i \neq \varnothing$, $C_i \cap C_j = \varnothing$ and $\bigcup_{i=1}^{s} C_i = V(G)$. We call $G(C_i)$ a *community* of G, which is identified with the induced sub-graph $G[C_i]=(C_i, E(C_i))$, where $E(C_i)=\{<u,v> | <u,v> \in E; u,v \in C_i\}$. Then $E(C)=\bigcup_{i=1}^{s} E(C_i)$ is the set of intra-community edges and $\tilde{E}(C) = E - E(C)$ is the set of inter-community edges. For example, a figurative drawing of a network with community structure is shown in Figure 1.

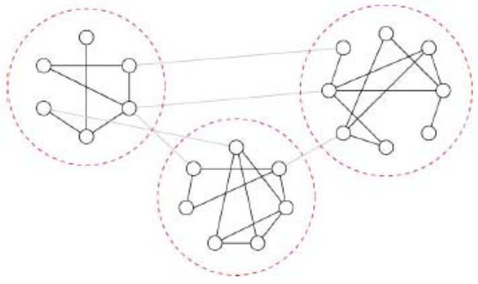


Figure 1. A small network with community structure of the type considered in this paper. In this case there are three communities, denoted by the circles, which have dense internal links but between which there is only a lower density of external links [12].

If the community number $s = 2$, graph G will be divided into two sub-graphs (a.k.a. communities): G_1 and G_2. Each node in G_1 or G_2 must belong to the node set of G, i.e. $V(G)$. Meanwhile, $V(G_1) \bigcup V(G_2) = V(G)$, which can also be expressed as $C_1 \bigcup C_2 = V(G)$. If there is an edge e_{ij} between the node v_i and v_j, which are all in one sub-graph G_1 or G_2, this edge should be classified into that sub-graph. On the contrary, if the two nodes v_i and v_j of edge e_{ij} belong to two different sub-graphs, for example, $v_i \in C_1$ and $v_j \in C_2$, the edge can't be classified into any community. Obviously, $\tilde{E} = \{e_{ij} =< v_i, v_j >| v_i \in C_1 \wedge v_j \in C_2\}$. Thus, $E(G_1) \bigcup E(G_2) \bigcup \tilde{E} = E(G)$. In this paper, we mainly consider the bipartite partitioning problem. The partitioning refers to two-part division in the rest of paper unless otherwise specified.

For a given network, there maybe exist several partitioning results. How to judge which one is the best partitioning is also an interesting problem. It's not hard to find that, the goal of partitioning a network into groups of nodes is to ensure that the connections within groups are dense and the connections between the groups are sparse. Among all proposed methods of evaluating the partitioning results, Newman and Girvan's method [13,14] is the most popular. They defined a measure of *modularity*, denoted by Q, which uses the denseness and sparsity of the groups' intra and interconnections to quantify community structure strength.

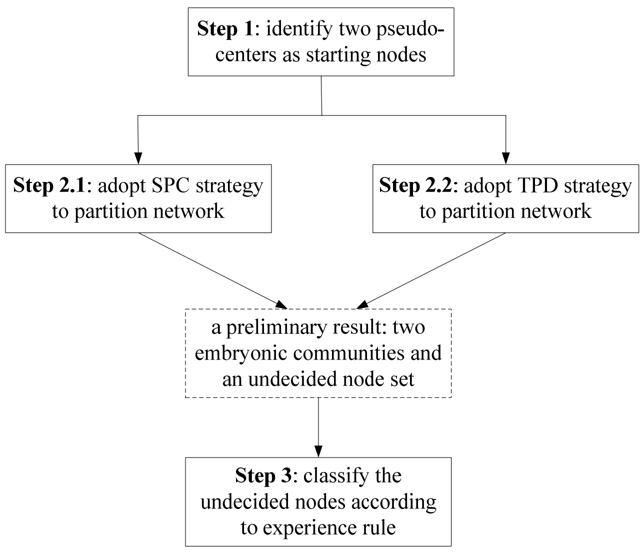
Given the symmetric matrix $\boldsymbol{e}$ whose element e_{ij} ($1 \leq i, j \leq |C|$) is the fraction of all edges in the network that link vertices in community i to vertices in community j, the row sums $a_i = \sum_j e_{ij}$ represent the fraction of edges that connection to vertices in community i. The modularity measurement can be defined via the following formula:

$$Q = \sum_i (e_{ii} - a_i^2) = \sum_i \left[\frac{m_i}{m} - \left(\frac{d_i}{2m} \right)^2 \right] \quad (1)$$

Where m is the number of all edges in network, m_i is the number of edges in community i, and d_i represents the sum of degree of each node in community i.

III. Algorithm Framework

As implied by the algorithm's name, our 3SHP algorithm can be divided into three stages: starting-point identification, preliminary partitioning and the final adjustment. The whole process of 3SHP algorithm can be illustrated in Figure 2.


Step 1: identify two pseudo-centers as starting nodes
Step 2.1: adopt SPC strategy to partition network
Step 2.2: adopt TPD strategy to partition network
a preliminary result: two embryonic communities and an undecided node set
Step 3: classify the undecided nodes according to experience rule
Figure 2. Overall framework of the three-stage heuristic partitioning algorithm.

At first, the algorithm selects two "core" points as the starting centers according to the feature information of network, such as degree distribution and diameter path (i.e., the longest one among all shortest paths). Secondly, two optional heuristic strategies are introduced to perform preliminary partitioning on network. SPC strategy removes the nodes along the shortest paths of these two pseudo-centers to get two basic isolated parts. In TPD strategy, algorithm can generate a series of node sets for each pseudo-center node through the *k*-step diffusing operation. Then, several possible divisions can be achieved by combining such diffusing node sets. For both strategies, the preliminary partition includes three parts: two embryonic communities and one undecided node set. Finally, an experience rule is used to classify the undecided nodes. Of course, the TPD strategy also needs to pick out the best result from the above several combinations.

IV. Key Steps of Algorithm 3SHP

A. Pseudo-center Identification

In most partitioning results of networks, it is not hard to find that each community has at least one core node, which has high degree in general. Therefore, we can partition network in line with this intuitive clue: select two nodes with higher degree as the "centers" of two communities, and then construct the community structure from each “core” node. In order to facilitate the expression described herein below, we introduce a concept of distance between two nodes in network here.

Definition 1 (*distance*) The distance between two nodes in network is the length (i.e. edge number) of the shortest path from one node to another node.

At first, it is necessary to count the degree information of each node. Based on the above results, all nodes in network can be sorted into descending order of degree. Then, the former k nodes can be picked out, denoted as $S_{top\text{-}k} = \{n_1, n_2, ..., n_k\}$. It should be noted that the k can be a specific number or a percentage of node number in whole network. According to the power-law character in complex network, the proportion of nodes with high degree is very low in general. Thence, the value of k usually is not so high. On the other hand, from a series of experiments, we find that $k = 8$ is suitable for a network with 100 nodes or less, while for a network with more than 100 nodes, $k \in [5\%, 10\%]$ will work better.

The next question is how to find out two nodes from the set $S_{top\text{-}k}$. A basic hint is that these two nodes should not be so near that they will be classified into one sub-group. That is to say, the distance between two “core” nodes should be as far as possible in network. Here, we adopt the following heuristic rule: find the diameter path (i.e., the longest one among all shortest paths) from a given network, any of which is optional if several shortest paths are all the longest. Suppose the two end-points of this path are N_{I} and N_{II} respectively. For the node N_{I}, we can calculate all distances from it to nodes in $S_{top\text{-}k}$, and find the node which is nearest to N_{I} from the set $S_{top\text{-}k}$ as a “core” node, denoted as C_{I}. Similarly, node C_{II} can also be picked out with respect to node N_{II}. It is noticeable that the above two nodes are not the indeed centers or cores in the final two communities, so they are viewed as pseudo-centers.

B. Strategy 1: SPC

Assume C_{I} and C_{II} are two “core” nodes for two parts respectively, then the shortest path cutting strategy can be used to calculate the preliminary partitions. At first, it is necessary to identify the shortest path from C_{I} to C_{II} in network G. Then, the nodes in the shortest path except C_{I} and C_{II} are removed as equivocal nodes (as shown in Figure 3). Meanwhile, a new graph G' can be achieved by removing the equivocal nodes from G. If graph G' is a disconnected graph, the algorithm stops and two preliminary partitions are yielded. Otherwise, we continue to compute the shortest path from C_{I} to C_{II} in network G', and then remove the nodes in such path until a disconnected graph can be yielded. Finally, all removed nodes are put into the set of undecided nodes.

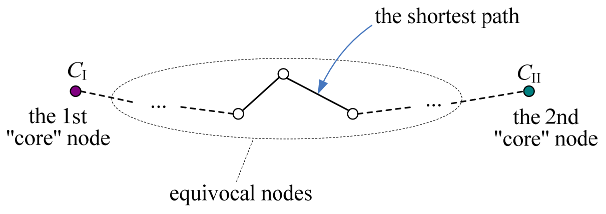


Figure 3. The illustration for removing the equivocal nodes along the shortest path between two “core” nodes.

Further, the partitioning strategy can be described in the pseudo-code form as follows.

```
Strategy SPC
Input:  A network $G$.
Output: Two embryonic communities (i.e. $EC_I$ and
        $EC_{II}$) and an undecided node set $V_{un}$.
1: {
2:   select two “core” nodes from the network according
     to the pseudo-center identification rule, denoted as
     $C_{\text{I}}$ and $C_{\text{II}}$;
3:   $V_{cut} = \varnothing$;
4:   $V_{un} = \varnothing$;
5:   while ($C_{\text{I}}$ and $C_{\text{II}}$ are connected)
6:   {
7:       Find the shortest path $p$ from $C_{\text{I}}$ to $C_{\text{II}}$ in $G$,
         whose node set is denoted as $V(p)$;
8:       $V_{cut} = V(p) - \{C_{\text{I}}, C_{\text{II}}\}$;
9:       $V_{un} = V_{un} \bigcup V_{cut}$;
10:      $G' = G - V_{cut}$; /* the associated edges in $E(G)$ are
                              also removed */
11:      $G = G'$;
12:  }
13:  view the connected sub-graph around $C_{\text{I}}$ as $EC_I$,
     and the connected sub-graph around $C_{\text{II}}$ as $EC_{II}$.
14:  return ($EC_I$, $EC_{II}$, $V_{un}$)
15: }
```

C. Strategy 2: TPD

The other strategy, called two-point diffusing (TPD), is similar to the spreading operation in L-Shell algorithm. It starts with these two points to perform expanding (also called diffusing) operation to get two embryonic communities and a set of undecided nodes. This strategy will produce several kinds of preliminary partitions. We use the modularity as the criterion to select the best one as the final output. In order to facilitate the expression, a concept about k-step diffusing is defined below.

Definition 2 (*k-step diffusing*) The k-step diffusing operation of a node is an action that walks from the given node to the farthest node, and the distance between them is exactly k steps. The k-step diffusing set of a node includes all nodes that are k steps away from the given node.

It’s not hard to find that, the 0-step diffusing set is the given node itself, and the largest diffusing set includes all nodes of network. Since the networks in real world possess the feature of small world [15], the k value in diffusing operation is not too large, and usually is a constant in the magnitude of 10^1. Here, we suppose that the largest diffusing steps of the "core" node C_{I} and C_{II} are l and w respectively. Then, the collection of diffusing sets of node C_{I} can be denoted as $DS(C_{\mathrm{I}}) = \{V_0^{\mathrm{I}}, V_1^{\mathrm{I}}, V_2^{\mathrm{I}}, ..., V_i^{\mathrm{I}}, ..., V_l^{\mathrm{I}}\}$. Similarly, $DS(C_{\mathrm{II}}) = \{V_0^{\mathrm{II}}, V_1^{\mathrm{II}}, V_2^{\mathrm{II}}, ..., V_j^{\mathrm{II}}, ..., V_w^{\mathrm{II}}\}$ for node C_{II}. Here, V_i^{I} is the *i*-step diffusing set for the "core" node C_{I}, V_j^{II} is the *j*-step diffusing set for the "core" node C_{II}. Obviously, $V_0^{\mathrm{I}} = \{C_{\mathrm{I}}\}$, $V_0^{\mathrm{II}} = \{C_{\mathrm{II}}\}$ and $V_l^{\mathrm{I}} = V_w^{\mathrm{II}} = V(G)$. Subsequently, the diffusing sets of two "core" nodes should be combined to generate two embryonic communities. As shown in Figure 4, $l \times w$-pair combinations can be obtained for $DS(C_{\mathrm{I}})$ and $DS(C_{\mathrm{II}})$. As mentioned above, the largest k is a constant value in general due to the small world character in complex network. Therefore, the value of $l \times w$ is not so great.

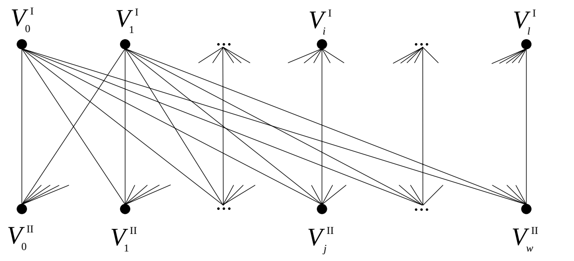


Figure 4. The illustration for combinations of *k*-step diffusing sets.

In order to generate the embryonic community structure, the diffusing sets from two "core" nodes at different steps should be combined together for further consideration. Here, the *i*-step ($0 \le i \le l$) diffusing sub-graph for the "core" node C_{I} is denoted as g_i, and the *j*-step ($0 \le j \le w$) diffusing sub-graph for the "core" node C_{II} is denoted as g_j. The whole network can be partitioned according to the following strategy, which is written in pseudo-code as follows. It is worth noting that, in the bottom loop-body the preliminary partition includes three parts: two embryonic communities and an undecided node set. Since the max-diffusing steps of C_{I} and C_{II} are l and w respectively, the final output are the sets of two embryonic communities and collection of undecided node set, and the supremum of cardinalities of these three sets are identical, i.e. $l \times w$.

Strategy TPD

```
Input:  A network $G$.
Output: The collection of two embryonic communities
        and collection of undecided node set, i.e.,
          $S(EC_I) = \{EC_0^{\mathrm{I}}, EC_1^{\mathrm{I}}, EC_2^{\mathrm{I}}, ..., EC_{l\times w}^{\mathrm{I}}\}$,
          $S(EC_{\mathrm{II}}) = \{EC_0^{\mathrm{II}}, EC_1^{\mathrm{II}}, EC_2^{\mathrm{II}}, ..., EC_{l\times w}^{\mathrm{II}}\}$ and
          $S(V_{un}) = \{V_0^{un}, V_1^{un}, V_2^{un}, ..., V_{l\times w}^{un}\}$.
1: {
2:   identify two "core" nodes from the network, denoted
     as $C_{\mathrm{I}}$ and $C_{\mathrm{II}}$;
3:   for 1 to max-diffusing steps of $C_{\mathrm{I}}$   //from 1 to $l$
4:   for 1 to max-diffusing steps of $C_{\mathrm{II}}$  // from 1 to $w$
5:   {
6:     denote i-step diffusing sub-graph of $C_{\mathrm{I}}$ as $g_i$;
7:     denote j-step diffusing sub-graph of $C_{\mathrm{II}}$ as $g_j$;
8:     if ( $g_i \cap g_j = \varnothing$ )
9:       continue;
10:    else
11:    {
12:       treat $g_i - (g_i \cap g_j)$ as the embryonic
           community of $C_{\mathrm{I}}$, denoted as $EC_{i\times j}^{\mathrm{I}}$;
13:       treat $g_j - (g_i \cap g_j)$ as the embryonic
           community of $C_{\mathrm{II}}$, denoted as $EC_{i\times j}^{\mathrm{II}}$;
14:       view $(g_i \cap g_j) + G - g_i - g_j$ as the undecided
           node set, i.e. $V_{i\times j}^{un}$;
15:    }
16:  }
17:  return ($\{EC_{i\times j}^{\mathrm{I}}\}$, $\{EC_{i\times j}^{\mathrm{II}}\}$, $\{V_{i\times j}^{un}\}$)
18: }
```

In general, g_i and g_j will not have the intersection for some cases, so the number of the real combinations is greatly less than the upper bound $l \times w$. After yielding the set $S(EC_{\mathrm{I}})$, $S(EC_{\mathrm{II}})$ and $S(V_{un})$, an experience rule will be used to assign undecided nodes in $V_{i\times j}^{un}$ to the embryonic community $EC_{i\times j}^{\mathrm{I}}$ and $EC_{i\times j}^{\mathrm{II}}$. Obviously, the assignment action will be executed for $l \times w$ times. Finally, the partition with max modularity can be picked out as the final resolution.

D. Classification for Undecided Nodes

After the undecided nodes are collected, how to classify them into a proper part is also a crucial task. The instinct experience tells us that the node in the candidate set should be classified into the closely-related partition. Therefore, we adopt the following strategy to handle the undecided nodes.

Suppose two embryonic partitions corresponding to the core C_{I} and C_{II} are EC_{I} and EC_{II} respectively, and the candidate set of undecided nodes is V_{un}. Given a node $v \in V_{un}$, which has n_1 edges connecting to EC_{I} and n_2 edges to EC_{II} in the original graph G. Then, two ratios can be calculated as below:

$$r_{\mathrm{I}} = \frac{n_1}{d(v)} \times 100\%, \qquad r_{\mathrm{II}} = \frac{n_2}{d(v)} \times 100\% \tag{2}$$

Where $d(v)$ represents the degree of node v in original network. If $r_{\mathrm{I}} > r_{\mathrm{II}}$, the node v should be added to the part EC_{I}. Otherwise, it will be classified into the second part. For the special case, i.e. $r_{\mathrm{I}} = r_{\mathrm{II}}$, the distances from node v to two "core" nodes can be viewed as the candidate criterion. If the distance from v to C_{I} is shorter than that from v to C_{II}, then v should be classified into EC_{I}, otherwise not. After each node and its corresponding edges are classified into some preliminary partition, we should update the partitions.

Subsequently, two new parts EC_{I}' and EC_{II}' can be achieved.

Similarly, the second node in the candidate set should be taken into account in the next iteration until all nodes in that set are completely handled. As a consequence, the final complete partitions can be yielded with the direction of the above rule.

V. Algorithm Complexity Analysis

In this section, the complexity of each strategy in our algorithm is separately analyzed. Since the first step and the third step are the commons in two strategies, we analyze them at first.

To identify two pseudo-centers, it is necessary to compute the statistical information about node degree. The time complexity for counting degree information is $O(m)$, and the complexity of selecting Top-*k* node set is $O(k \cdot n)$. For the *k* candidate nodes, the time complexity of determining the pseudo-centers is *k*-fold of that for generating the shortest path in network. In an unweighted graph, the shortest path can be obtained by breadth first search (BFS) algorithm, so the corresponding time complexity is $O(n+m)$. Therefore, the complexity of the first step is $O(m+k\cdot n+k\cdot(n+m)) = O(k\cdot(n+m))$, here n and m denote the number of vertices and edges respectively (as addressed in Section 2), k is a small proportion of *n* (i.e., $k \ll n$).

For classifying the undecided nodes, suppose the cardinality of the undecided node set is u, then the complexity of this step can be expressed as $O(un)$. In general, $u \ll n$, the complexity in the third step can also be denoted as $O(n^2)$.

In SPC strategy, the max step-number for breaking the connection between the "core" node C_{I} and C_{II} is n. In the while loop, the main task is to compute the shortest path from C_{I} to C_{II}. As mentioned above, the time complexity of generating the shortest path is $O(n+m)$. Therefore, if our algorithm adopts SPC strategy to partition network, the overall complexity (i.e., the complexity including step1, 2.1 and 3) is $O(k\cdot(n+m)+n\cdot(n+m)+n^2)$. For $k \ll n$, thus $O(k\cdot(n+m)+n\cdot(n+m)+n^2)$ is equal to $O(n\cdot(n+m))$. In brief, the complexity of the SPC strategy in 3SHP algorithm (i.e., 3SHP-SPC) is $O(n\cdot(n+m))$.

In TPD strategy, the outer loop will be executed for $l \times w$ times. In the execution body, it needs computation with $O(n^2)$ to form the embryonic communities. Finally, $l \times w$ kinds of combinations of two embryonic communities and an undecided node set can be yielded. For each combination, it needs the computation with complexity $O(n^2)$ to classify the undecided nodes. Therefore, the time complexity of TPD strategy in 3SHP algorithm (i.e., 3SHP-TPD which includes step 1, 2.2 and 3) is $O(k\cdot(n+m)+ \quad l\times w\cdot(n^2+n^2))$. Simply, it is $O(l\times w\times n^2)$. As mentioned in section 4.3, the value of l or w is not too large, usually in the magnitude of 10^1.

It should be noted that, our algorithm is heuristic rule-based method, so the real computing time can be much shorter than the theory value, which will be approved in the following empirical analysis.

VI. Empirical Analysis

In order to validate the effectiveness and efficiency of our algorithm, a comparison experiment has been performed on three real-world social networks. As for the network details, please refer to Table 1. The experiment is employed in the environment of Eclipse 3.2 and runs on Pentium 4 with 1.8GHz and 1 GB memory. The tool runs on Windows XP SP2, and the Java runtime environment is JRE1.6.0_05. In order to obtain the comparison data, we also implemented other two representative algorithms (i.e. L-Shell and GN) in our experiment.

TABLE I. Basic Characters of The Real-world Networks Used in Our Experiment

ID.	#Node	#Edge	Description
G_1	34	76	The well-known "karate club" study of Zachary [7,11,13].
G_2	62	159	An undirected social network of frequent associations between 62 dolphins in a community living in Doubtful Sound, New Zealand [16].
G_3	105	441	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 co-purchasing of books by the same buyers. The network is compiled by V. Krebs and is unpublished, but can found on Krebs' web site [17].

The network of Karate Club is a classical instance to verify the correctness and efficiency of some specific algorithm. For this instance, our algorithm (both SPC and TPD strategy) can achieve the best result (*Q*-value=0.36842) as shown in Figure 5. For the L-Shell and GN algorithm, the *Q*-values are 0.30748 and 0.35596 respectively. The reason for getting the optimal result lies in the placement of node 3 and 10. In the results of algorithm L-Shell and GN, the two nodes are all classified into same community as node 34. However, our TPD algorithm assigns it to the community with node 1, so our algorithm can achieve the highest *Q*-value. More importantly, our algorithm is advantageous in the escaped time, only 15 and 16 milliseconds for SPC and TPD strategy respectively.

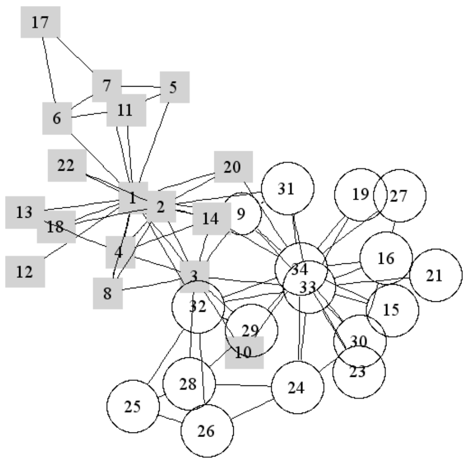


Figure 5. The partitioning result by our algorithm (both SPC and TPD strategy) for the Zachary's network.

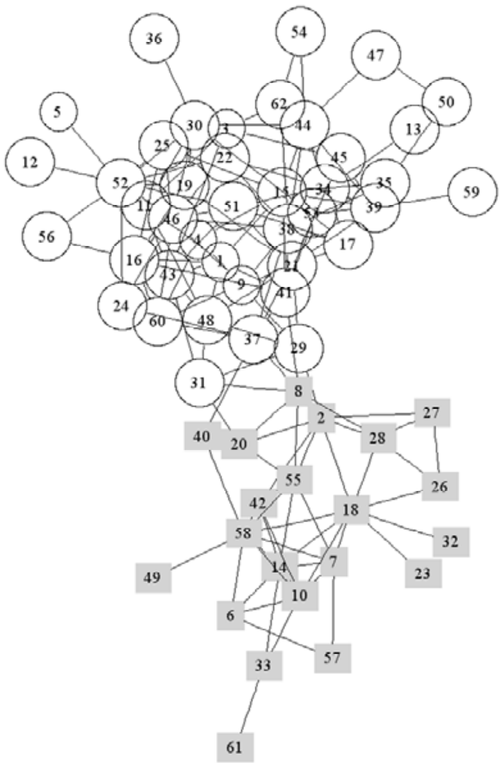


Figure 6. Results of the second network by SPC strategy.

The second example is a non-human social network, i.e. a network of dolphins. The two strategies in our algorithm can generate two different results as shown in Figure 6 and 7 respectively. While adopting SPC strategy to partition network, the final two communities are pictorially represented in Figure 6. The corresponding *Q*-value and computing time are 0.3787 and 31 ms, respectively. Figure 7 is the result of TPD strategy, its *Q*-value is 0.38986 and the computing time is also 31 ms. By contrast, GN algorithm works as well as SPC strategy but worse than TPD strategy. It consumes 110 ms to generate a partition with *Q*-value 0.3787. However, L-Shell algorithm is worse than both strategies due to its *Q*-value 0.36612 and escaped time 328 ms.

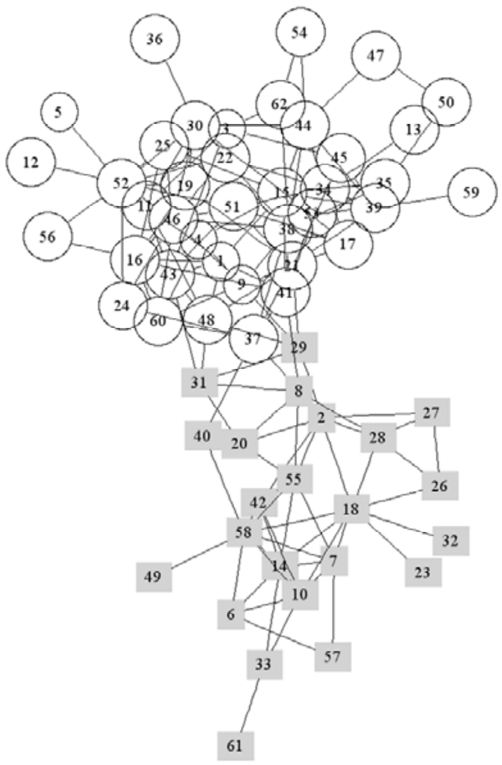


Figure 7. Results of the second network by TPD strategy.

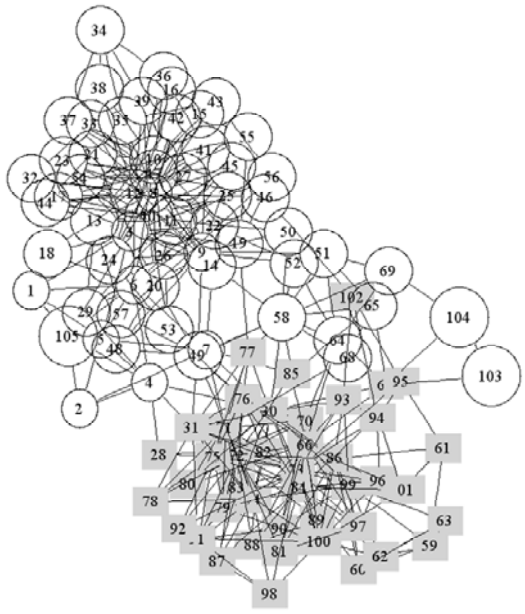


Figure 8. Results of the third network by SPC strategy.

For the third network, strategy SPC and TPD also produce different results. As shown in Figure 8 and 9, the second strategy can yield much more precise partition than the first. It computes the result with highest modularity *Q*=0.45655 in 78 ms. Unfortunately, the strategy SPC can merely compute a partition with *Q*-value=0.43920. In this case, algorithm L-Shell and GN produce very closed results, and their *Q*-values are 0.44397 and 0.44241 respectively. However, their computing time is much longer than ours, exceeding 1500 ms.

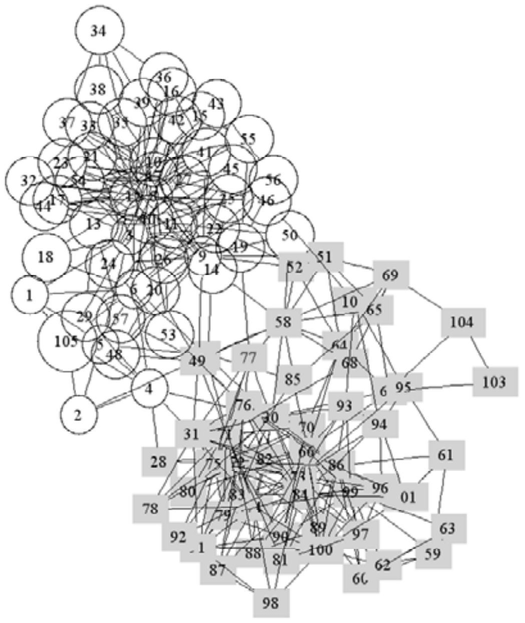


Figure 9. Results of the third network by TPD strategy.

The details about partitioning results of three networks are listed in Table 2 for comparison. Moreover, the *Q*-value comparison is also visualized in Figure 10. Based on the comparison analysis, we can draw a conclusion that our algorithm proves to be competitive in terms of the optimization ability and computational efficiency. For all three cases, strategy TPD in 3SHP algorithm can generate the highest partitioning quality for them. Meanwhile, the computing time of this strategy is shorter than algorithm L-Shell and GN. For the other strategy, i.e. SPC, it can also generate much better results. The corresponding *Q*-value is the highest for some cases or much closed to the best one. On the other hand, the computing time is as short as strategy TPD. It is worth noting that, the computing time of our algorithm doesn't have great fluctuation with the rise of network size. In other words, our algorithm has favourable scalability, i.e. the processing time doesn't fast grow with the size of network. On the contrary, the computing time of other two algorithms increase rapidly with the node number.

TABLE II. RESULT COMPARISON BETWEEN THREE ALGORITHMS FOR THE REAL-WORLD NETWORKS

Network	Algorithm	Node number of the 1st community	Node number of the 2nd community	*Q*-value	Time (ms)
G_1	L-Shell	13	21	0.30748	46
	GN	15	19	0.35596	47
	3SHP-SPC	17	17	0.36842	15
	3SHP-TPD	17	17	0.36842	16
G_2	L-Shell	24	38	0.36612	328
	GN	21	41	0.37870	110
	3SHP-SPC	21	41	0.37870	31
	3SHP-TPD	23	39	0.38986	31
G_3	L-Shell	49	56	0.44397	2203
	GN	64	41	0.44241	1594
	3SHP-SPC	62	43	0.43920	78
	3SHP-TPD	51	54	0.45655	78

Of course, the proposed algorithm is validated only by networks with the magnitude of 10^2 at present. But these three cases are well-known and have been used to check the effectiveness of algorithm in many literatures. Our algorithm generates the best result for all of them, which surely proves its scalability. Although our algorithm mainly deals with the two-part division problem, it's obvious that it can also be used for *k*-partitioning ($k > 2$) of a network through appropriate transformation.

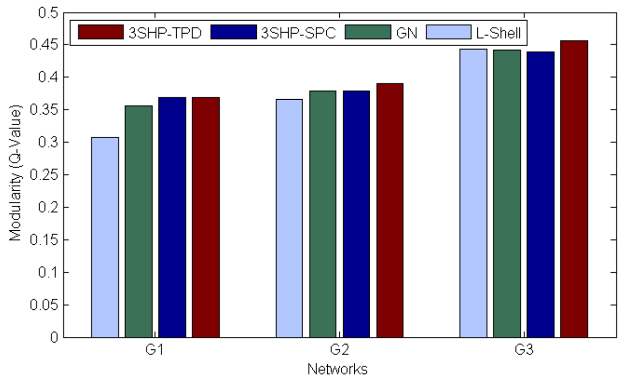


Figure 10. *Q*-value comparison of four algorithms for three real-world networks.

VII. RELATED WORK

In recent years, complex network analysis has attracted research interests, and community detection (also called graph clustering) became a hot topic in this field. Until today, quite a few methods have been proposed to solve this problem. Recently, reference [10] and [18] reviewed the existing work systemically. Here, we only present researches comparable with our algorithm.

The first one is the well-known Kernighan-Lin algorithm [7], which is based on greedy optimization for the cost function. Its basic idea is to introduce a gain function into the network partitioning problem. It reaches the best solution through inter-changing the nodes in the former designated partitions. However, it requires assigning the node number of each community before partitioning, which is hard to determine only based on people's priori knowledge [8].

Girvan and Newman proposed an approach based on link removal to detect an unknown number of communities [9]. They focused on those edges which are the most "between" communities. This algorithm can provide better partitioning result, but doesn't have advantage in computing time. In addition, it will produce so declining partitions for some specific networks (e.g. random networks) [10]. In contrast, the computing efficiency of our algorithm shows a great improvement from GN algorithm.

L-Shell algorithm proposed by Bagrow and Bollt [11] is an agglomerative clustering-based method, actually a diffusing method. It starts with a specific node until the ratio of the emerging degree, i.e. Δk_j^l, at step l to that at step $l-1$ is lower than a cut-off value α. The main difference between that algorithm and ours lies in our TPD strategy spreads the community structure from two pseudo-core nodes. In addition, L-Shell algorithm has two fatal weaknesses: (1) It needs a search procedure to find the best starting-point. (2) An iterative loop is also needed to determine the optimal threshold value, i.e. α. Fortunately, our proposed method can avoid the above time-consuming process, so it has a great advantage in computing time.

In addition, some traditional meta-search algorithms such as evolutionary algorithm are also adopted to settle this problem [19]. However, this type of algorithm usually has no advantage in computing time due to the slow convergence speed.

VIII. Concluding Remarks

Community partitioning is a key but difficult problem in social network analysis. Based on the in-depth analysis on the partitioning results, some heuristic strategies are proposed to solve the two-part division problem. At first, two nodes are treated as pseudo-centers based on the clues of the diameter path and degree information. Based on these two "core" nodes, two strategies, i.e. the shortest path cutting (SPC) and two-point diffusing (TPD), are adopted to generate two embryonic communities. Subsequently, an experience rule is used to adjust the undecided nodes. A comparison experiment is performed to validate the effectiveness and efficiency of our algorithm. The results show that our algorithm can yield excellent community partition results with very short computing time.

The result of our algorithm is promising, but there are a couple of open issues to be answered in the future work. Here, the networks in experiment are not so large, so we need collect much larger network to perform comparison analysis. On the other hand, we are currently working on modify this algorithm to handle multi-part division problem. Furthermore, using this algorithm to treat the weighted network partitioning problem also needs to be explored.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 60803046 and 61063013, Natural Science Foundation of Jiangxi Province under Grant No. 2010GZS0044, the Science Foundation of Jiangxi Educational Committee under Grant No. GJJ10433, the State Key Laboratory of Software Engineering under Grant No. SKLSE2010-08-23, and the Youth Foundation of Jiangxi University of Finance and Economics. The author is grateful to Zhenmei Zhu, Man Luo and Qiong Zhang for their help and enlightening discussions.

References

[1] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, Cambridge University Press, London, UK, 1994.

[2] J. Scott, *Social Network Analysis: A Handbook* (*2nd ed.*), Sage Publication, London, 2002.

[3] N. Gulbahce and S. Lehmann, "The art of community detection," *BioEssays*, Vol.30, 2008, pp.934-938.

[4] N. Du, B. Wang, and B. Wu, "Community detection in complex networks," *Journal of Computer Science and Technology*, Vol. 23, 2008, pp. 672-683.

[5] J. Duch and A. Arenas, "Community detection in complex networks using extremal optimization," *Physical Review E*, Vol.72, 2005, 027104.

[6] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Höfer, Z. Nikoloski, and D. Wagner, "On Finding Graph Clusterings with Maximum Modularity", *Proc. of the 33rd Intl. Workshop on Graph-Theoretic Concepts in CS* (*WG'07*), Dornburg, Germany, June 2007.

[7] B. W. Kernighan and S. Lin, "An effective heuristic procedure for partitioning graphs," *Bell System Technical Journal*, Vol. 49, 1970, pp. 291-307.

[8] X. Wang, X. Li, and G. Chen, *Complex Network Theory and Its Applications*. Tsinghua University Press, Beijing, China, 2006, pp.164-165. (in Chinese)

[9] M. Girvan and M. Newman, "Community structure in social and biological networks," *Proc Natl Acad Sci U S A*, Vol. 99, 2002, pp. 7821-7826.

[10] B. Yang, D.-Y. Liu, J. Liu, D. Jin, and H.-B.Ma, "Complex network clustering algorithms," *Journal of Software*, Vol.20, 2009, pp.54-66. (in Chinese)

[11] J. P. Bagrow and E. M. Bollt, "Local method for detecting communities," *Physical Review E*, Vol.72, 2005, 046108.

[12] M. Newman, "Detecting community structure in networks," *The European Physical Journal B - Condensed Matter*, Vol.38, 2004, pp. 321-330.

[13] M. Newman and M. Girvan, "Finding and evaluating community structure in networks", *Physical Review E*, Vol. 69, 2004, 026113.

[14] M . Newman, "Modularity and community structure in networks," *Proc Natl Acad Sci U S A*, Vol. 103, 2006, pp.8577-8582.

[15] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, Vol. 74, 2002, pp.47-97.

[16] 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 Socio-biology*, Vol. 54, 2003, pp. 396-405.

[17] V. Krebs, *Dataset: PolBooks – Krebs' Amazon Books*, http://www.orgnet.com, data can be downloaded from Website: http://vlado.fmf.uni-lj.si/pub/networks/data/mix/mixed.htm

[18] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, 2007, pp.27-64.

[19] A. Gog, D. Dumitrescu and B. Hirsbrunner, "Community detection in complex networks using collaborative evolutionary algorithms," *Proc. of ECAL'07*, LNAI 4648, 2007, pp. 886-894.