Journal of Computers Vol. 29 No. 3, 2018, pp. 165-178
doi:10.3966/199115992018062903015

A Dynamic Method to Adjust Overlapping Community Structures

Yi-Jen Su*, Wei-Lin Hsu

Department of Information Engineering and Computer Science, Shu-Te University, Taichung, Taiwan
{iansu, wlin}@stu.edu.tw

Received 13 September 2017; Revised 13 October 2017; Accepted 9 November 2017

Abstract. Identifying all cohesion subgroups in a complex network is a critical issue in social network analysis. Once new trends coming from new actors' interactions, a community detection method has to adopt an effective mechanism to adjust to structural changes in community composition. This research proposes a novel two-phase distributed method for dynamic overlapping community discovery on the MapReduce framework to improve performance. The first phase implements a static overlapping community detection. First, all neighbors of each node are collected from the complex network, and the TTT algorithm is applied to enumerate all maximal cliques. Following that, all adjacent maximal cliques are merged to complete the overlapping detection operation. In the second phase, the proposed dynamic overlapping community detection method is applied on influence area to adjust the struct of detected community structure. This approach applies MapReduce to adjust detected community's structure in order to avoid redundant analysis and enhance efficiency. The experiment's data are derived from YouTube users' interaction. The static overlapping community detection consists of six groups ranging from 50,000 to 300,000 nodes. The data for the dynamic overlapping community discovery consists of five groups ranging from 2,000 to 10,000 nodes, which are in turn dynamically added to the static overlapping communities to generate the final clustering results.

Keywords: community discovery, MapReduce, social network analysis

1 Introduction

As mobile networks become common, Internet users engage in frequent interactions with one another on social media platforms such as Instagram, Facebook, Twitter, Google+, Plurk and more. The phenomenon has encouraged virtual communities to form complex networks, each with its distinctive features. In general, a social network can be seen as a graph G(N, E) that includes a set of nodes, N, and a set of edges, E, between the nodes. Most of the time, each node represents an actor (like a user account or member) and each edge represents a binary relation between a pair of nodes.

If a service provider collects social media information from its social media services, they can identify user habits and discover user opinions through long-term social listening. Social Network Analysis (SNA) [4-5] detects communities in complex networks to assist enterprises in understanding information of specific user groups and designing suitable marketing content to promote future business.

The proposed distributed overlapping community detection method focuses on improving performance in using MapReduce to discover all maximal cliques from a complex network. Every two distinct nodes in the clique are adjacent. A maximal clique is the size of a clique that cannot be enlarged but still satisfies the structure constraints of a clique. To satisfy the constraints of a clique structure, most discovered communities become too small-scaled to correspond to facts happening in real life. In order to avoid the aforementioned drawbacks, this research adopts the Clique Percolation Method (CPM) [3] based on the subgroup structure of 1.5 club (3-clique in undirected graph) to detect all cohesion subgroups in a complex network.

* Corresponding Author

The operation of distributed static overlapping community detection includes two levels: MapReduce [2] operations and maximal cliques merging. The purpose of the 1evel-1 MapReduce operation is to collect the neighborhood information of each node in the network. The output of the 1evel-1 MapReduce operation is employed as the input for the level-2 MapReduce operation to enumerate all maximal cliques by the TTT algorithm [7, 11]. Last, following the CPM algorithm, all adjacent maximal cliques will be merged incrementally.

To respond to new coming events from new actors' interactions, this paper reanalyzed the whole social network to complete community detection increase the computation load. Therefore, this research proposes a novel dynamic community detection method that only focuses on the influence area to adjust the discovered community structures to avoid mass amount of computation and enhance the performance. Generally speaking, new events can be categorized into four kinds: (1) generation of new communities, (2) the merging of communities, (3) addition of new member nodes to some communities, and (4) the community structure does not change.

The criteria for evaluating this method's effectiveness includes consideration of the number of communities, number of overlapping nodes in communities, and average cluster coefficient. The study compares the static community detection method compares against another famous overlapping community detection method, CPM. The original dataset collected from YouTube includes 1,134,890 nodes and 2,987,624 binary relations between nodes in the social network; however, the study only employed six randomly selected sizes of datasets ranging from 50,000 to 300,000 nodes to observe the effectiveness of the proposed method. The evaluation of the dynamic overlapping community discovery was performed on five groups, each ranging from 2,000 to 10,000 nodes, which are in turn dynamically added to the static overlapping communities to generate the final clustering results.

2 Related Works

2.1 MapReduce

The MapReduce framework was first introduced by Google in 2004 to provide an easy-to-use programming framework to develop distributed applications [2]. The basic operation of MapReduce is for the JobTracker of MasterNode to divide a job into several tasks and then deliver them to the TaskTracker of SlaveNodes. Following that, both map tasks and reduce tasks are applied to complete the received tasks, and the results are returned to the JobTracker, as shown in Fig. 1. The input consists of 6 edges between 6 nodes. Given the key-value format, both the mapping and reducing operations conduct rearrangement and collection of these information, and these nodes form into two node clusters as a result of a distributed processing.

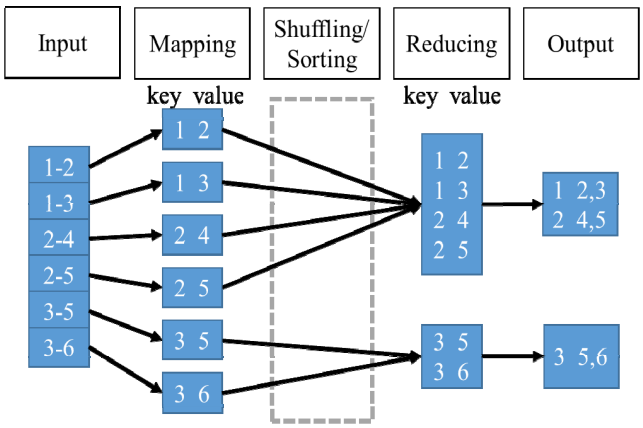

Fig. 1. The operation flow of MapReduce

Wu et al. [1] proposed a two-level MapReduce framework to realize the Maximal Clique Enumeration (MCE) algorithm. When using only the clique structure to present the discovered community, the constraint becomes too strict to detect large-scale communities. On the other hand, communities of too small a size cannot reply to real community structures in the observed cases and this will reduce the value

of future applications of the detected community structure.

2.2 Maximal Cliques

In social network analysis, a high-quality community contains high dense intra-relations and infrequent inter-relation with other communities. The maximal clique is a clique that cannot be enlarged but still satisfies the constraints of a clique structure. If using the maximal clique enumeration (MCE) algorithm [1] to list all maximal cliques in a complex network, all detected communities are cliques. For example, Fig. 2(a) employs 7 nodes and 12 edges as the input to detect all maximal cliques. Fig. 2(b) illustrates the result, which includes 3 cliques-{1,2,3,4}, {3,4,7}, and {5,6,7}.

(a) The input undirected graph (b) The subgrouping result of MCE

Fig. 2. An example of MCE operation

In 1973, Bron and Kerbosch [6] presented the BK algorithm to enumerate all maximal cliques from a complex network. However, it was Tomita et al. that designed the TTT algorithm [7] in 2006 to improve performances by applying the depth first search (DFS) with branch pruning method. The research adopted the TTT algorithm in the level-2 MapReduce operation to find all maximal cliques in an input complex network.

2.3 Clique Percolation Method

A clique has the highest intra-relation density compared to other subgroup structures. Given that the clique structure condition is too strict, it is hard to form a large-sized subgroup. The CPM algorithm incrementally merges adjacent k-cliques with the k-1 common members to enlarge the size of cohesive subgroups [3]. The CPM not only keeps the structure quality of detected communities, but also successfully enlarges the size of discovered subgroups. For example, Fig. 3(a) shows a 4-clique, {1,2,3,4}, and a 3-clique, {3,4,5}. Since the two 3-cliques, {1,3,4} and {3,4,5}, share two common members, {3,4}, the flip over operation will merge the smaller one, {3,4,5}, into the bigger one, {1,2,3,4}. Fig. 3(b) shows the final subgrouping result-a 5-node subgroup, {1,2,3,4,5}, and an isolate node, {6}.

(a) Before the CPM operation (b) After the CPM operation

Fig. 3. Example of a CPM 3-Clique operation

2.4 Cluster Coefficient

Community detection is an important research topic in SNA. The most cohesive subgroup in a complex network is a complete subgraph, also called a “clique”, where all member node pairs are connected by an

edge. In large-scale complex networks, a high-quality result of community detection holds a much higher intra-relationship density of communities than the inter-relationship density between communities [8]. The most popular method to evaluate detection result is Clustering Coefficient (CC) [10, 12]. The CC of community Ci is calculated as shown in Equation (1):

$$CC_i = \frac{|\, v_{jk}, v_k \in N_i, e_{jk} \in E_i \,|}{\frac{k_i * (k_i - 1)}{2}} \tag{1}$$

where community C_i contains a set of nodes N_i and a set of edges E_i. Meanwhile, e_{jk} presents the relationship between a pair of nodes, v_j and v_k, that belong to N_i while k_i stands for the number of nodes in community C_i.

The overall subgrouping quality CC can be evaluated by calculating the average cluster coefficient of communities, as shown in Equation (2):

$$CC = \frac{1}{k} \sum_{i=1}^{k} CC_i \tag{2}$$

where k stands for the number of detected communities in a complex network.

3 Research Method

To improve the performance of dynamic community detection, this research adopts a distributed method in designing a two-phase community detection method, which includes a static phase and a dynamic phase. In the static phase, a two-level MapReduce is adopted to propose a distributed method that completes the static overlapping community detection. All listed maximal cliques are checked for whether they are adjacent to other maximal cliques. Once the case is established, the CPM algorithm is adopted to incrementally merge them into larger size of subgroups. With these features, the proposed community detection method not only solves performance issues but also enlarges the size of discovered communities.

The static phase operation of the distributed community detection method is divided into three steps. First, the level-1 MapReduce gathers the neighborhood information of each node from the input social network. Following that, the level-2 MapReduce utilizes the collected neighborhood information to enumerate all maximal cliques. Using MCE's clique set, the CPM completes the community detection process by continually merging adjacent k-cliques.

Most of the previous research on community detection focused only on the static community detection; however, the static community structure cannot effective work with the new coming events. Therefore, a dynamic community structure adjustment mechanism is needed to avoid reanalysis of the whole network any time a new event happens. In the dynamic phase, the operations of the distributed community detection method follow an operating order that includes 5 steps. In Step 1, MapReduce is employed to identify the influence area of new coming events and it tags new interactions between nodes with new event labels. Steps 2 to 4 follow similar procedure to that of the static phase operations to retrieve all information concerning community structure changes while Step 5 utilizes subgroup merge information to adjust the community structure.

3.1 Level-1 MapReduce Operation in the Static Phase

The target of the level-1 MapReduce is to retrieve the neighborhood information of each node in the input graph. The input of Mapper is the ascending order edge set. Following the order of keys, the output is the key-value combination-but the key has to be larger than the value. Each node serves as an output in Reducer, and its neighbor nodes also follow the same node order.

Fig. 4 serves as the input undirected graph for finding the potential cohesive subgroups. Fig. 5 is the output of the level-1 MapReduce operation. The input contains all 12 edges of the input graph. To take Node 1 as an example: the map operation collects all of Node 1's neighbor nodes into an edge set that includes 4 edges. Then, the reduce operation merges the four edges into a tuple, Node 1a and its neighbors {2,3,4,5}.

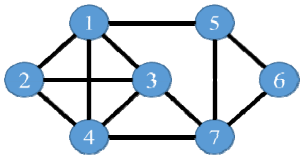

Fig. 4. The input graph of community detection

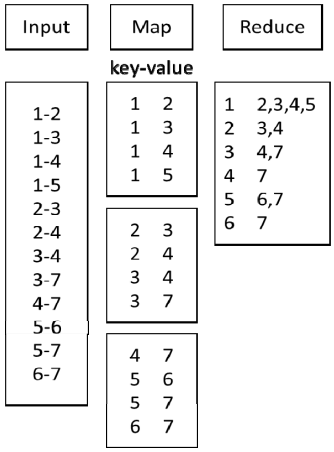

Fig. 5. The operation of Level-1 MapReduce

Algorithm 1 includes Map and Reduce operations. The input of Map is the list of all edges in the complex network. To avoid duplicate processing of the same relations in Reduce operation, all edges must follow the ascending order of the node index by the emit() function, and the output format is a key-value list.

Algorithm 1: Level-1 MapReduce

```
Input: List of Edges (k,v)
Mapper (k,v)
  if (k>v)
    emit (v,k);
    return;
  end if
  emit (k,v);  //key-value list

Reducer (k, list(v))
  output (k, list(v));
```

3.2 Level-2 MapReduce Operation in the Static Phase

The level-2 MapReduce swaps the location of the key node with its neighbor nodes to retrieve information about neighbors' neighbors for preparing the MCE work, as shown in Fig. 6. Taking Node 2 as an example, <2,<3,4>> means that Node 2 has two neighbor nodes, node 3 and node 4. To retrieve information about neighbors' neighbors, the swap operation is executed and the result is <3,<2,4>> and <4,<2,3>>. To avoid collecting the same cliques in the reduce operation, the key has to be the biggest value in the identified cliques among all enumerated maximal cliques. Therefore, the keys, including 2, 3, 5, 6, cannot be used to generate cliques.

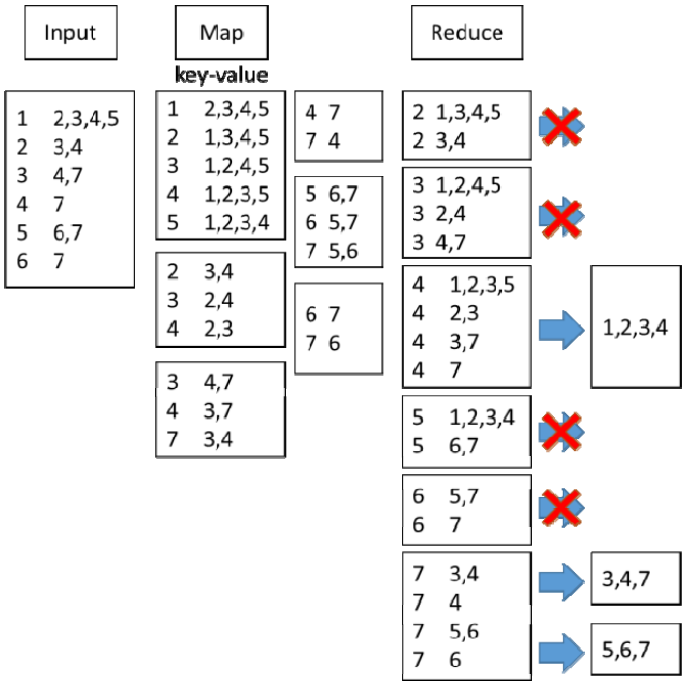

Fig. 6. The operation of Level-2 MapReduce

The TTT algorithm is adopted to list all maximal cliques, as shown in Fig. 7. The symbols in MCE operation includes □, which stands for u, the node with the largest degree. Node ● is the neighbor of node u, and node ○ does not connect to node u. After the DFS processing, node △ is adopted to avoid generation of duplicate cliques. The value of node ▽ is always higher than the value of the key node. Take Key 4 as an example. Node 4 has the neighbors {1,2,3,7}. Since Node 3 has the highest degree in the ● node set {1,2,3,7}, Node 3 is selected to be the next key. After processing the key node 1, Node 2 is the last key node. Because 7 is bigger than 4, Node 7 is marked with ▽ and does not have advance searching.

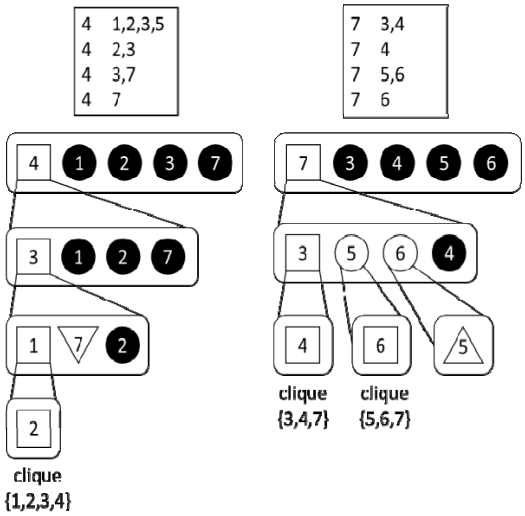

Fig. 7. Using the TTT algorithm to reach MCE

In Algorithm 2, the input is the output of level-1 MapReduce, including node k and its neighbors list(v) as the key-value list. Following the neighbors' sequence to swap the location, the reduce operation can collect all neighbors' neighbor information by emit() function and executes the TTT algorithm to list all maximal cliques. The Reducer transfers list(v) to node k's relation graph G_{sub}^{k} and stores node k's neighbors in two sets, SUBG and CAND, which will later be applied in the TTT algorithm to find out all

maximal cliques.

Algorithm 2: Level-2 MapReduce

```
Input: k, list(v) //key-value list
Mapper (k,list(v))
  emit (v, list(v)) ;
  for each v'∈list(v)
    emit(v', {k} U (list(v)-v'));
  end for

Reducer (k, list(v))
  SUBG = Γ(k); //node k's neighbors
  CAND = Γ(k);
  Q = {k};
  TTT (k, G^k_sub , SUBG, CAND, Q)
```

In algorithm 3, the TTT algorithm enumerates all maximal cliques via the DFS method. Set SUBG includes both processed nodes and unprocessed nodes while Set CAND stores only unprocessed nodes. Set Q includes nodes that have relations connecting itself to all others. Initially, SUBG is used to determine whether there still exists unexplored nodes. If the set is empty, it means no more nodes need to be explored or to perform clique output. The variable u stands for the node in SUBG that has the most number of neighbors in CAND. The set EXT_u includes Node u and nodes without relations to Node u. Next, Node q is retrieved from the set EXT_u and DFS is performed until all nodes in EXT_u have been processed, with the exception of skipping nodes whose value is larger than the key node. $SUBG_q$ and $CAND_q$ narrow down the search space to only the nodes related to Node q and insert Node q to Set Q. Both $SUBG_q$ and $CAND_q$ are recursively pass to the algorithm TTT to complete the maximal cliques search.

Algorithm 3: The Distributed TTT Algorithm

```
TTT (k, G^k_sub , SUBG, CAND, Q )
  if (SUBG = Φ)
    print (Q);
    return;
  end if
  node u ∈ SUBG & u has Max. degree;
  EXT_u=CAND-Γ(u); // Γ(u), node u's neighbors
  while (EXT_u≠ Φ)
    for each q ∈ EXTu;
      if (q>k)
      CAND = CAND-{q} ;
      end if
      Q =Q U {q};
      SUBGq = SUBG ∩ Γ(q);
      CANDq CAND ∩ Γ(q);
      TTT (k, G^k_sub , SUBG_q, CAND_q, Q);
      CAND = CAND-{q};
      Q = Q-{q};
  end for
wend
```

3.3 Merging Adjacent Maximal Cliques in the Static Phase

Once all maximal cliques are identified, all of them will be inserted into the clique table, and each record has a unique index number. When two maximal cliques are adjacent, the merge table will keep the record and assign one merge id to it. Because clique <1,2,3,4>, with id 1, and clique <2,3,7>, with id 2, are

adjacent, the merge table set contains two records with the same merge id, 1, as shown in Fig. 8. Because the clique <5,6,7>, with id 3, does not have any adjacent clique, the merge table holds only one record with the merge id 3. There are two different merge ids in the merge table. Therefore, the final result of community detection is two communities, <1,2,3,4,7> and <5,6,7>. In addition, Node 7 is present at two communities, rendering it an overlapping node.

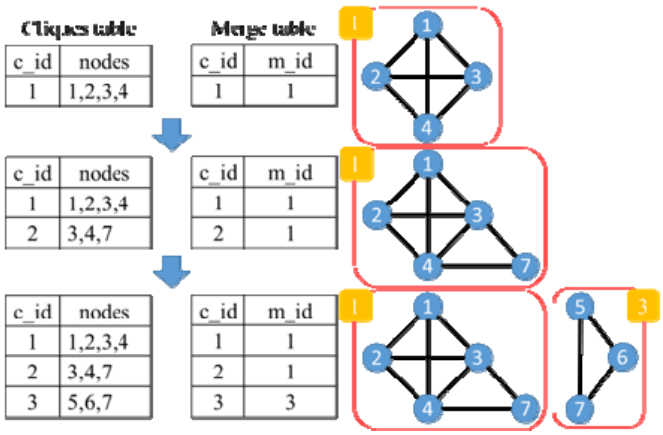

Fig. 8. Example of maximal cliques merging

3.4 Identifying Influence Area in Dynamic Phase

In the dynamic phase, in order to meet performance requirements, MapReduce is also applied in discovering influence area by new coming events from a complex network. First of all, all relations between nodes are tagged with one of the three kinds of event tags, including original event, new event, and grouped event. Each kind of tag serves as the input of the same kind of mapper of MapReduce.

Fig. 9, for example, contains two subgroups 1 and 2. The three kinds of relations illustrate new events E_n={2-4, 5-6, 6-7} as the input of Map(E_n), the key-value, original relations E_o={1-2, 1-3, 2-3, 3-4, 3-6, 4-6, 5-7} as the input of Map(E_o), and grouped relations E_g including the relations {1-2, 1-3, 2-3} of Group 1 and relations {3-4, 3-6, 4-6} as the input of Map(E_g).

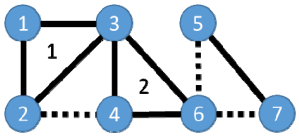

Fig. 9. An ungrouped undirected graph

The operation of Map focuses on the preprocessing the influence area as the input for the Reduce. Map(E_n) reverses the order of input relations and tags them with label E_n. Map(E_o) and Map (E_g) reverse their input relations with tag E_o and E_g. Then, the Reduce operation receives the aforementioned three kinds of relations and arranges them in order. To detect the influence area, once a set of relations shares the same key and includes edges with tag E_n, then this edge set will be reserved; otherwise, it can be ignored. In Fig. 10, there is no edge with tag E_n in the Key-1 set and Key-3 set; the Key-1 group and Key-3 group are not influenced by new events. However, Key-2, Key-4, Key-5, Key-6 and Key-7 groups are all influenced. Next, it checks for whether duplicate relations exist in the input data, abiding by the key-value reserving order to divide all events into three categories grouped events, original events, and new events. In addition, if a relation bears tag E_n and key larger than the key-value, then it does not output the relation to the next operation.

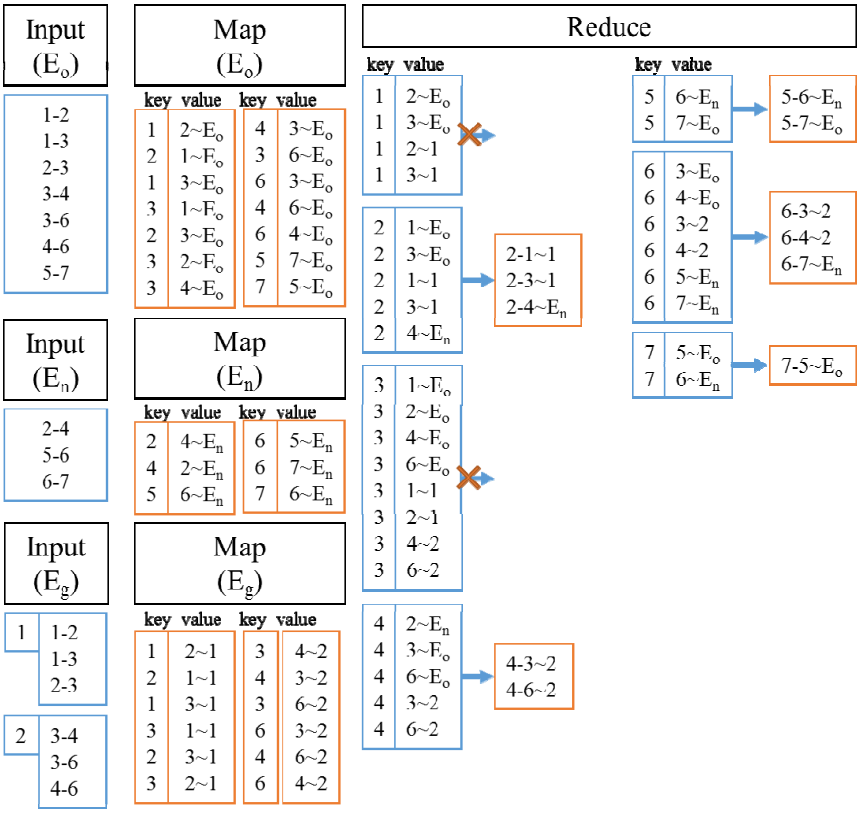

Fig. 10. Detection of influence area by MapReduce

3.5 Merging Adjacent Maximal Cliques

The last operation in the dynamic phase is to merge all adjacent maximal cliques. All detected maximal cliques are the output of the MCE operation of TTT algorithm. As shown in Fig. 11, for example, the input lists all detected maximal cliques in the influence area by new coming events and follows the order of clique id to insert all cliques to the cliques table for merging adjacent maximal cliques. The merge table records all information about subgroup merging, c_id shows the index number of the subgroup, and m_id stands for the id of the merged group.

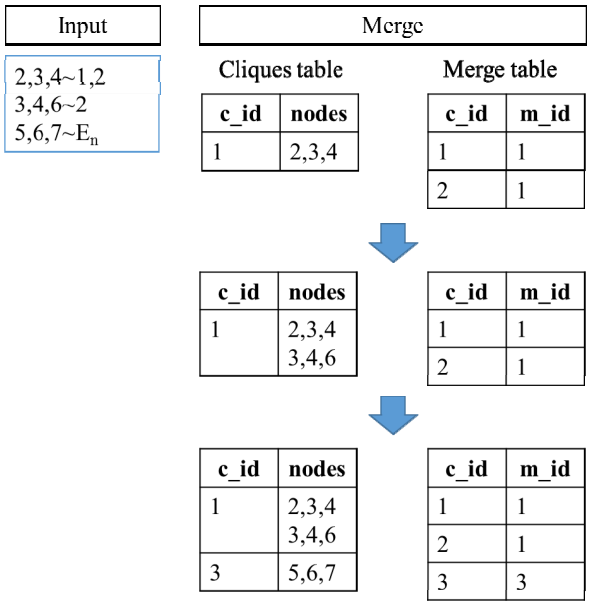

Fig. 11. The flow of adjacent maximal cliques merging

In Fig. 11, the first set of data, 2,3,4~1,2, shows that the clique contains three nodes, <2,3,4>, causing subgroup 1 and subgroup 2 to merge. The smaller id, 1, will be reserved as the id of the merged group and all member nodes to the field nodes. The Merge table keeps the merging information about clique:1 and clique:2. Given that the second clique, clique:2, contains three nodes, <3,4,6>, that share the same relation 3-4 with clique:1, clique:1 and clique:2 will be merged as a new clique with the clique id 1. The third input record, 5, 6, 7~En, bears a new event tag, En, and does not share any relation with other cliques. It will become a new clique with the clique id 3 and its information is inserted to the Merge table.

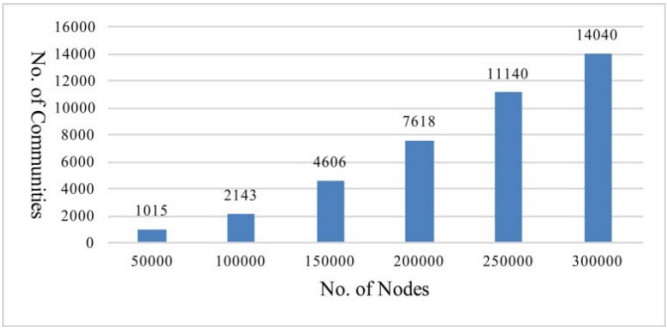

Fig. 12. Number of communities in different datasets

The output of the dynamic phase is illustrated in Fig. 13. There are two detected subgroups: <1,2,3,4,6>, with group id 1, and <5,6,7>, with group id 2.

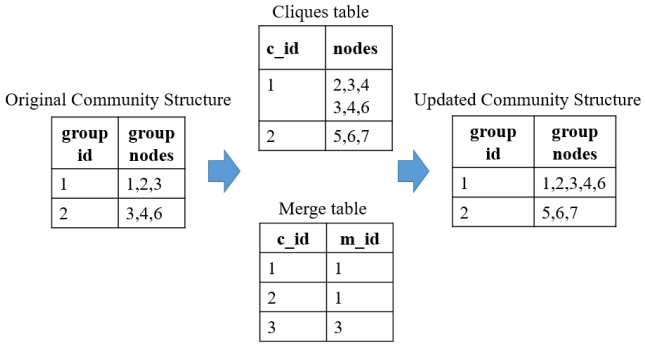

Fig. 13. The result of dynamic community detection

4 Experiment Results

The experiment's hardware environment consists of two parts, one for the MapReduce distributed experiment and the other one for the centralized experiment. The distributed environment is constructed through MapReduce and has one master node and two data nodes. The hardware specification includes the following: the CPU is Intel Core i7-3770 with 3.4 GHZ clock rate, the size of RAM is 8GB, and the HDD space is 500GB. The software environment includes the following: the operating system of each distributed experiment is Ubuntu 12.04 and Hadoop 1.2.1 while the operating system of the centralized experiment is MS Windows 7.

4.1 Experiment of Static Community Detection

The dataset is retrieved from the website SNAP (http://snap.stanford.edu), which provides the user-relation network of YouTube [9] that includes 1,134,890 nodes and 2,987,624 relations. Six datasets were randomly selected from the YouTube network. The size of the datasets ranges from 50,000 nodes to 300,000 nodes, as shown in Table 1.

Table 1. The list of the six datasets

Index	No. of Nodes	No. of Relations
1	50,000	73,151
2	100,000	165,730
3	150,000	376,664
4	200,000	672,992
5	250,000	1,036,038

The result of community detection is checked with the number of communities, the average cluster coefficient, and the execution time. In Fig. 12, it can be seen that when the size of dataset increases from 50,000 nodes to 300,000 nodes, the number of detected communities also increases, from 1,015 to 14,040. When the size of dataset is smaller than or equal to 200,000 nodes, each time the dataset receives 50,000 more nodes, the number of detected communities doubles itself. Once the size of dataset becomes larger than 200,000 nodes, the number of detected communities increases at a slower rate.

As shown in Fig. 14, when the size of dataset falls between 50,000 and 300,000, the average values of cluster coefficient (CC) of the six datasets fall between 0.919 to 0.937. If the value of CC is high, it signifies that the intra-relationship within a group is concrete. Although the CC value of detected communities is lower than the CC value of clique structure, but the subgrouping quality of these six datasets are still acceptable.

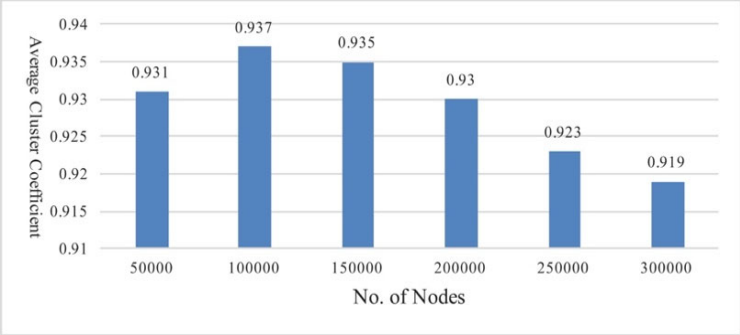

Fig. 14. Average cluster coefficient in different datasets

In Fig. 15, the CPM algorithm runs on only one computer, requires more memory to store the whole complex network's data, and demands more computation time to complete the task of community detection with single thread operation. By comparison, this research's proposed distributed community detection method can run simultaneously on three computers, requires less memory space for computation, and thus speeds up the subgrouping task. It can also be seen from Fig. 12 that when the size of dataset increases, the execution time of CPM's one-computer approach increases sharply; on the other hand, the increase in the distributed method is much less obvious.

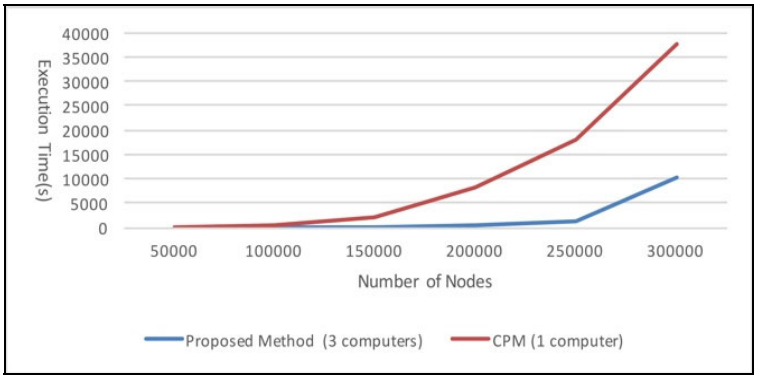

Fig. 15. Comparison of execution time

4.2 Experiment of Dynamic Community Detection

The experiment of the dynamic community detection focuses on observing changes that occur with the addition of new events. Table 2 presents six different-sized complex network datasets that each receive five different number of new events, including 2,000, 4,000, 6,000, 8,000, and 10,000. Once new events happen in the 50,000- to 150,000-node complex networks, new groups form at a faster rate. Because the structure of a smaller-scale complex networks is not stable, when new events happen, there are more new relations between nodes, in turn leading to higher relation density and formation of more cohesion subgroups. The same idea is illustrated in Table 3, in which smaller-scale complex networks also gain more nodes to groups when new events happen.

Table 2. Number of new groups after new events happen

	No. of New Events				
No. of Nodes	2,000	4,000	6,000	8,000	10,000
50,000	49	100	150	207	268
100,000	70	158	231	317	402
150,000	81	168	249	335	421
200,000	74	149	225	302	361
250,000	43	94	155	216	270
300,000	32	59	91	119	150

Table 3. Number of new nodes that are added to groups after new events happen

	No. of New Events				
No. of Nodes	2,000	4,000	6,000	8,000	10,000
50,000	279	604	1,098	1,427	1,789
100,000	474	1,029	1,641	2,204	2,738
150,000	530	1,158	1,842	2,456	2,979
200,000	525	1,128	1,828	2,423	2,921
250,000	419	874	1,410	1,875	2,304
300,000	267	499	812	1,091	1,327

Fig. 16 shows the change in average cluster coefficient in six different-sized six datasets when new events happen. Because the community structure is not stable, the datasets of 50,000 and 100,000 nodes show greater change when addition of different number of new events occur. When the complex is bigger in size, the change in average cluster coefficient when varying-sized new event sets are added is not obvious.

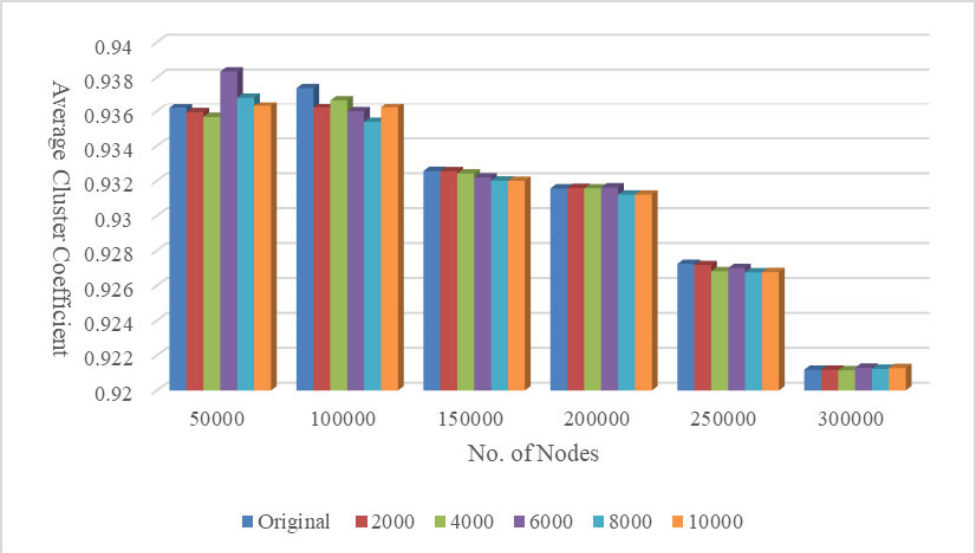

Fig. 16. The comparison of average cluster coefficient

This study executed three cases of community detection using different approaches with different number of computers, as shown in Fig. 17, including static community detection with 3 machines and dynamic community detection with 3 machine and 5 machines respectively. It is clear that the static community detection approach takes far more execution time than the other two dynamic solutions, but it can work well on complex networks of a smaller scale. For example, for complex networks with 52,000 to 160,000 nodes, because the dynamic community detection method runs on one additional MapReduce, the static community detection method yields better execution time. However, when the scale of complex network increases to 202,000 nodes and 310,000 nodes, the advantage of the dynamic method becomes more obvious. The same case is also true when it comes to number of machines. Once the number of machine increases from 3 to 5, the performance enhances sharply when the scale of complex network is larger than 152,000 nodes.

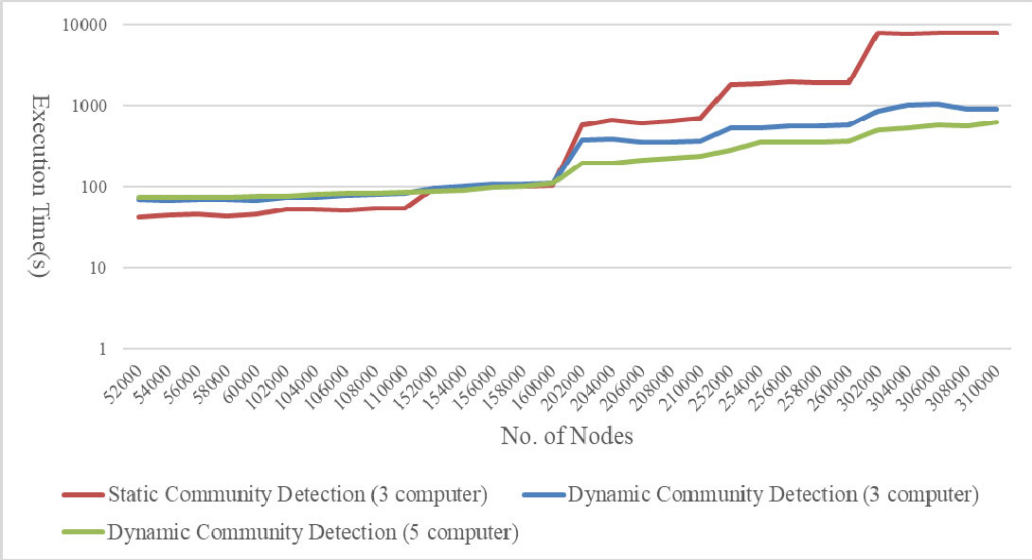

Fig. 17. The comparison of execution time between static/dynamic community detections

5 Conclusion

Past researches on community detection employed centralized approaches that easily to slow down computation processes, especially when the scale of complex network is large. Therefore, this research proposes a distributed community detection method to improve community detection. The CPM algorithm is also adopted to counter the disadvantage of MCE, which only uses maximal cliques as small size communities. The CPM incrementally merges adjacent cliques into a group to solve the problem of community structure being too strict.

This research proposes not only a distributed static community detection method but also a dynamic community detection method. The novel method can effectively avoid analysis of whole complex networks when new events happen by focusing on the influence area when adjusting the detected community structure, in turn saving mass amount of computation.

The experiment results indicate that the distributed community detection method can effectively improve performance. In addition, the average cluster coefficient is employed to measure the overall subgroup quality. As for measurements of the subgrouping quality, the lowest value of cluster coefficient was 0.919, when the dataset contained 300,000 nodes. The reduced amount of CC is still acceptable.

The experiment results also show that the dynamic community detection method yields better performance than the static method. The results prove that the dynamic method can work well on large-scale complex network. This research will continue with future works in the same direction of striving to lower down the execution time of community detection and enhancing the quality of overall subgrouping.

References

[1] B. Wu, S. Yang, H. Zhao, B. Wang, A distributed algorithm to enumerate all maximal cliques in MapReduce, in: Proc. the 4th International Conference on Frontier of Computer Science and Technology, FCST 2009, 2009.

[2] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM 51(1)(2008) 107-113.

[3] G. Palla, I. Derenyi, T. Vicsek, The critical point of k-clique percolation in the Erdos-Renyi graph, Journal of Statistical Physic 128(2007) 219-227.

[4] S. Wasserman, K. Faust, Social Network Analysis: Methods and Applications (Vol. 8), Cambridge University Press, Cambridge, 1994.

[5] J. Scott, Social Network Analysis, Sage, New York, 2017.

[6] C. Bron, J. Kerbosch, Finding all cliques of an undirected graph, Communications of the ACM 16(1973) 575-577.

[7] E. Tomita, A. Tanaka, H. Takahashi, The worst-case time complexity for generating all maximal cliques and computational experiments, Theoretical Computer Science 363(2006) 28-42.

[8] J.C. Turner, Towards a cognitive redefinition of the social group, in: H. Tajfel (Ed.), Social Identity and Intergroup Relations, Cambridge University Press, Cambridge, 1982, pp. 15-40.

[9] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, Knowledge and Information System 42(1)(2015) 181-213.

[10] L. Barabâsi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, T. Vicsek, Evolution of the social network of scientific collaborations, Physica A: Statistical mechanics and its applications 311(3)(2002) 590-614.

[11] A.S. Himmel, H. Molter, R. Niedermeier, M. Sorge, Adapting the Bron-Kerbosch algorithm for enumerating maximal cliques in temporal graphs, Social Network Analysis and Mining 7(1)(2017) 35.

[12] Q. Liao, Y. Yang, Incremental algorithm based on wedge sampling for estimating clustering coefficient with MapReduce, in: Proc. the 2017 8th IEEE International Conference on Software Engineering and Service Science, ICSESS 2017, 2017.