The Ninth International Conference on Web-Age Information Management

Scalable Community Discovery of Large Networks

Zhemin Zhu[†] Chen Wang[‡] Li Ma[‡] Yue Pan[‡] Zhiming Ding[†]

[†]*National Engineering Research Center of Fundamental Software*
Institute of Software, Chinese Academy of Sciences, Beijing, China
zhuzhemin@gmail.com zhiming@iscas.ac.cn
[‡]*IBM China Research Laboratory, Beijing, China*
{chwang, malli, panyue}@cn.ibm.com

***Abstract*— Over the past decade, community structure, a statistical property of networked systems such as social network and world wide web, has attracted considerable attention in data mining field because it enables description and prediction of complex networks. Many highly sensitive graph clustering algorithms were developed for identification of communities having dense connections internally and loose connections with others. In this context, Newman and Girvan [1] proposed modularity *Q* score for quantifying the strength of community structure and measuring the fitness of a division. The *Q* function has become an important standard recently. In this paper, combining the strengths of the Q score and multilevel paradigm [2] first developed for graph partitioning, we introduced a scalable algorithm MOME (i.e. Modularity-based Multilevel Graph Clustering) to efficiently discover communities from a network. The experimental results indicated that MOME ran extremely faster and finally achieved a division with a slightly higher *Q* score against the latest modularity-based method and its variant [3], [4], particularly when the network was of a large-scale.**

I. INTRODUCTION

Community identification is a common interest of intelligent networked systems in different areas such as World Wide Web, social network, semantic web, biochemical network, and so on. Generally speaking, a high-quality community denotes a group of vertices with a higher-than-average density of edges connecting them in a network. The vertices belonging to a tight-knit community are more likely to have common properties. Figure 1 shows a sketch of a network with such community structure. The discovered communities, for example, can be used to describe complex organizations and their associations in collaboration web, or to learn evidence for implicit modules with some degree of functional independence in metabolic networks' dynamics. Fast and scalable community detection method is practically useful to describe and predict large scale complex networked systems precisely.

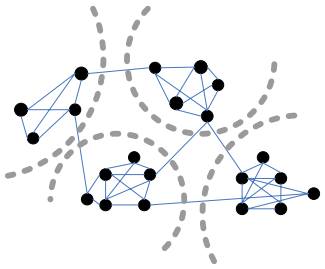

Fig. 1. An example of community structure

The problem of community discovery has been studied in many disciplines. Graph partitioning is a simpler version of this problem, and graph partitioning techniques has been widely applied in the applications like parallel computing and VLSI (i.e., Very Large-Scale Integration) design for decades. It aims at separating a graph into a prescribed number of densely connected equal-sized subgraphs, which are linked each other by the minimum number of edges. In many complex networks, however, we often have no idea in advance how many communities we wish to discover and what scale of communities should be suitable. Therefore, many efforts have been made to tackle this problem of automatically detecting communities from a network without any initial parameters like the number of discovered communities. For example, people have defined some interesting measurements or models of community structure, such as *ratio cut* [5], *normalized cut* [6], *min-max cut* [7], and modularity *Q* [1], to evaluate the goodness of discovered communities. Meanwhile, the argument on which one can better model community structure is going on as well. None of them can be universally accepted as a standard to be applicable in any cases for now. In this paper, we utilize the *Q* function for quantifying the strength of communities, because it has been proven a good candidate measure to capture the natural division.

The *Q* function is one of the popular measurements for community identification at present. A few empirical studies like [8], [9] showed that the algorithms based on the *Q* function had two practical advantages: 1) by maximizing *Q* score for a division, they could automatically discover optimal number of communities without the need of prescribing this number; 2) they tried to divide networks as natural as possible, while other measurements always tended to keep higher scores for balanced communities of approximately equal size. Nevertheless, optimizing the largest *Q* score has been proven a NP-Complete problem [10] in the strong sense.

Recently, a few general studies like [11] have been done to compare graph clustering algorithms from a computational cost perspective. Among the modularity based algorithms, hierarchical clustering (agglomerative and divisive) and spectral clustering were two common methods. Newman and Girvan [1] first proposed the *Q* function in a divisive clustering algorithm as objective metric for choosing the number of communities into which a network should be divided. Major disadvantages of the divisive clustering algorithm included high computational complexity of $O(n^3)$ (using shortest path betweenness) and comparatively low *Q* score for discovered communities. Subsequently, Newman [3], [4], Whitey and

978-0-7695-3185-4/08 $25.00 © 2008 IEEE
DOI 10.1109/WAIM.2008.13

Smyth [12] developed spectral clustering algorithms respectively. They performed better on Q score against hierarchical clustering methods because a modularity matrix of a network was formed directly in case of the Q function and used to globally optimize Q score by its most dominant eigenvectors. However, two factors affected the performance of the spectral algorithms when we were going to discover communities from a large-scale network. First, large-scale matrix led to costly calculation of matrix-vector multiplication, which was a key step in eigenvector solutions like classical power method. Next, convergence of the matrix-vector multiplication could not be guaranteed in any case. Once the multiplication did not converge, the algorithms would recursively run unless the prescribed upper limit of loops was reached.

In this paper, we present a scalable algorithm MOME (i.e. **Mo**dularity-based **M**ultilevel Graph Clust**e**ring) to efficiently identify communities from a network. MOME combines the optimization process of Q score into multilevel paradigm [2], and improves the efficacy of community detection particularly when the network is of large-scale. The experimental results indicate that for small- and medium-scale networks, it performs up to 3 orders of magnitude faster against the latest modularity-based method and its variant [3], [4]. Furthermore, it only takes about 3 minutes with a laptop to successfully identify 243 communities from the IMDB (i.e. Internet Movie Database) network[1] having more than 1 million vertices and 3 million edges, while the competing methods fail to handle the network of such scale. In addition, the Q score test shows that MOME performs slightly better as well.

The rest of this paper is organized as follows: The related works are listed in Section II. Section III gives basic knowledge of modularity Q and multilevel paradigm. The algorithm MOME is introduced in Section IV. In Section V, we conduct a couple of experiments to evaluate the performance and Q score against the latest modularity-based methods. Finally, Section VI concludes the paper.

II. Related Work

Over the past decade, many algorithms and approaches have been contributed to solve the problem of community identification. In this section, we outline the related works which resulted in the significant impacts in this area.

The modularity Q was first proposed by Newman and Girvan [1] to quantify the strength of communities and measure the fitness of a division, which has been empirically proven useful in community discovery. However, maximizing the Q score was a NP-Complete problem [10] in the strong sense. Many methods were developed to accelerate the mining process and promote the Q score of discovered communities.

Newman [9] exploited a greedy algorithm to merge vertices (or subgraphs) recursively, which would locally maximize the Q gain in each round. It was the first time to detect communities by optimizing the Q function itself. Further, Clauset et al. [8] improved the greedy algorithm with a refined data structure and claimed that it could perform in $O(nlog^2n)$ time for sparse graph where n was the number of vertices. However, the Q score had not been promoted against its predecessor.

In order to solve the problem of local maximization of the Q score in greedy algorithm, a few global optimization algorithms were utilized to help find better answers. Based on simulated annealing method, Guimera et al. [13] and Reichardt and Bornholdt [14] introduced two systems respectively to relax the condition of heuristic search for locally optimal value with distinct probability. Based on extremal optimization method, Duch and Arenas [15] presented a system to iteratively move the vertices with the lowest fitness into other communities until an optimal state with a maximal Q score was satisfied.

In addition, there was another popular way to globally optimize the Q function, i.e., spectral algorithm. Newman [3], [4] and White and Smyth [12] proposed at the same time to reformulate the problem of maximizing the Q score as an eigenvector problem involving a matrix of "Q-Laplacian". Their experimental results showed that spectral algorithms could make natural division. However, the computation on the matrix was time- and space-consuming. They were restricted to scale up to large networks.

Besides the Q function, *ratio cut* [5], *normalized cut* [6], and *min-max cut* [7] were also considered as good cost functions of community identification. A lot of methods had been proposed around these functions to detect communities from a network. However, they tended to induce balanced communities. Compared to modularity Q, they might not automatically decide the optimal number of communities in a network.

We mainly concerned cost functions based solutions in this paper. However, there were still a few simulation based approaches. For instance, before Newman and Girvan proposed modularity Q, they [1] ever presented a few divisive hierarchical clustering algorithms depending on the calculation of edge "betweenness". And the *Max-flow/Min-cut* [16] might be used to divide networks into communities as well.

III. Preliminary Knowledge

In this section, we revisit the basic knowledge of modularity Q and multilevel paradigm, both of which are the building blocks of our algorithm.

A. Modularity Q

Here, we briefly review the principle of modularity Q first presented by Newman and Girvan [1]. Readers might refer to [8], [9], [3] for details.

Modularity is a property of a network and a specific proposed division of a network into communities. Modularity function is a function of the particular division, with larger values indicating stronger community structure. Hence, we could, in principle, find good divisions of a network into communities by optimizing the modularity over possible divisions.

Given an undirected graph $G = (V, E)$, let A_{vw} be the element of adjacent matrix of graph G thus:

$$A_{vw} = \begin{cases} 1 & \text{if vertices v and w are connected;} \\ 0 & \text{otherwise.} \end{cases}$$

[1] http://www.imdb.com/interfaces

and suppose the vertices are divided into communities such that vertex v belongs to C_v. Then, the Q function can be formulated as follows:

$$Q = \frac{1}{2|E|}\sum_{v,w}(A_{vw} - \frac{d_v d_w}{2|E|})\delta(C_v, C_w),$$

where $|E|$ is the number of edges in graph G, d_v and d_w are the degrees of both vertices v and w, and δ is a function as below:

$$\delta(C_v, C_w) = \begin{cases} 1 & C_v = C_w; \\ 0 & \text{otherwise.} \end{cases}$$

To simplify the calculation of Q function in the recursive process, we deduct its formula as follows:

$$\begin{aligned} Q &= \frac{1}{2|E|}\sum_{v,w}(A_{vw} - \frac{d_v d_w}{2|E|})\delta(C_v, C_w) \\ &= \frac{1}{2|E|}\sum_{v,w} A_{vw}\delta(C_v, C_w) - \frac{1}{2|E|}\sum_{v,w}\frac{d_v d_w}{2|E|}\delta(C_v, C_w) \\ &= \frac{1}{2|E|}\sum_{c=1}^{k} E_c - \frac{1}{4|E|^2}\sum_{v}(d_v \sum_{w} d_w)\delta(C_v, C_w) \\ &= \frac{1}{2|E|}\sum_{c=1}^{k} E_c - \frac{1}{4|E|^2}\sum_{c=1}^{k}(\sum_{v} d_v\delta(C_v, c)\sum_{w} d_w\delta(C_w, c)) \\ &= \frac{1}{2|E|}\sum_{c=1}^{k} E_c - \frac{1}{4|E|^2}\sum_{c=1}^{k} D_c^2. \qquad (I) \end{aligned}$$

where E_c is a double of the edges whose both vertices belong to the community c, D_c is the total degree of the vertices within the community c, and k is the number of communities.

From the formula above, we can interpret that

$$\begin{aligned} Q = &(\text{number of edges within communities}) \\ &-(\text{expected number of such edges}). \end{aligned}$$

If no edges exist to connect vertices across clusters then Q = 1, and conversely if the number of inter-cluster edges is no better than random then Q = 0, just as claimed in [8]. Newman and Girvan found that real-world unweighted networks with high community structure generally have Q scores within a range from 0.3 to 0.7. Note that modularity Q could be easily extended to weighted graphs.

Compared to other cost functions like *ratio cut* [5], *normalized cut* [6], and *min-max cut* [7], modularity Q has two practical advantages. (1) It can be used to automatically determine the optimal number of communities. (2) It can effectively overcome the bias of balanced communities.

B. Multilevel Paradigm

Multilevel paradigm has been widely applied to solve graph partitioning problem which is a simpler version of community identification. Since Bui and Jones [2] first proposed the multilevel method, many practical graph partitioning systems like [17], [18], [19], [20] have further carried forward the idea.

As described in [21], multilevel algorithm first performs a successive approximation to an original graph and receives a sequence of graphs in which the size of each graph is always smaller than that of its predecessors. The approximation process does not stop until the size of the coarsest graph is small enough to be fast processed by common graph partitioning methods. For example, heavy edge matching is a common approximation method. In this course, both vertices bridged by the edge with the most heavy weight are collapsed into one vertex iteratively and the vertex weight is accumulated accordingly. Other common approximation methods include random matching, heavy clique matching, and light edge matching. The final step is to derive the partitioning results recursively from the coarsest graph, via its successive predecessors, finally back to the original graph. Many move-based graph partitioning algorithms like KERNIGHAN-LIN procedure [22] and FM algorithm [23] can be further exploited to recursively refine the results in the final step. In the multilevel paradigm, the above process can be divided into three phases: coarsening phase, initial partitioning phase, and uncoarsening & refinement phase, e.g., as shown in Figure 2 which has been described in [21].

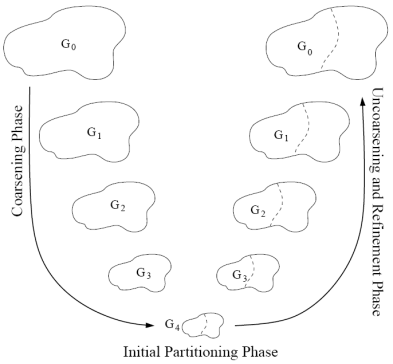

Fig. 2. Multilevel paradigm [21].

IV. ALGORITHM MOME

In this section, we introduce a scalable algorithm MOME that combines the optimization process of Q score into multilevel paradigm. As described in Section III-B, the basic structure of multilevel algorithm is very simple. It consists of 3 phases: coarsening, initial partitioning, and uncoarsening & refinement. Here, we further simplify the initial partitioning phase by regarding the vertices in the coarsest graph as an initial set of communities on the successive approximations of the original graph. Therefore, MOME only includes 2 phases of coarsening (including initial partitioning) and uncoarsening. The top framework of MOME is shown in Algorithm 1.

We will introduce the two steps of MOME respectively in detail as following.

A. Coarsening Phase

Coarsening is an important phase in MOME to identify and collapse together groups of vertices which are highly

Algorithm 1 The top framework of MOME.

Input: A weighted graph $G_0 = (V_0, E_0)$.
Output: A set of communities CS.

1) Coarsen down the graph G_0 and approximate it into a sequence of smaller graphs G_1, G_2, ..., G_m such that $|V_0| > |V_1| > |V_2| > \cdots > |V_m|$;
2) Directly regard vertices in the coarsest graph G_m as communities and project them back to the original graph G_0 by iteratively refining the communities through its successively approximations G_{m-1}, G_{m-2}, ..., G_1.

connected. Many related works, e.g., [21], have been proposed to facilitate this task. A common method for graph coarsening is to collapse together the pairs of vertices that form a matching. A matching of the graph is a set of edges, no two of which are incident on the same vertex. Vertex matchings can be computed by various methods, such as random matching, heavy-edge matching, maximum weighted matching, and approximated maximum weighted matching. In MOME, we adopt a clustering based coarsening scheme [21], which can reduce the iteration of coarsening process in power law graphs and thus compute the matchings more efficiently. The recursive step of coarsening is shown in Algorithm 2.

Algorithm 2 The recursive step of coarsening in MOME.

Input: A weighted graph $G = (V, E)$ having n vertices.
Output: A set of successive approximations GC.
Method: Coarsening(G, GC).

1) FOR each vertex v_i of G in random
2) FOR each vertex v_j in adjacency list of v_i
3) Calculate the $MergeQ$ score between v_i and v_j;
4) Select one vertex v_j having max $MergeQ$ score;
5) If $maxMergeQ > 0$
6) Merge the vertices v_i and v_j into a new vertex v;
7) Update adjacency list of coarsening graph G_c;
8) If the number of vertices in G_c is less than n
9) $GC \leftarrow GC \cup G_c$;
10) Coarsening(G_c, GC).b

In this recursive procedure, MOME first generates a random order for the vertices in the input graph. According to the random order, it visits all vertices one by one. In each round, MOME calculates $MergeQ$ scores between the current vertex v_i and its adjacent ones v_js. $MergeQ$ score denotes a modularity gain after two vertices are merged into one. Thus, the biggest one is picked out and its corresponding adjacent vertex is merged with the v_i to form a new coarse vertex v. Here, we restrict that the mergence will take place only while the biggest $MergeQ$ score is greater than zero. Therefore, MOME always contributes a positive modularity Q gain locally in each coarsening step. Finally, the adjacency list of the input graph need to be updated. After visiting all vertices in the current round, MOME will recursively call this procedure to coarsen the new approximation until the coarse graph can not be contracted any more.

In addition to the original graph, we might consider the graph in each level as a multigraph, where a group of vertices in the original graph are contracted to a vertex, bundles of edges connecting two groups are regarded as multiple edges between two vertices, and edges internal to a group are denoted as self-edges of a vertex. The core of the coarsening is to calculate $MergeQ$ scores between the adjacent vertices and merge the pairs with the largest scores in each level. Figure 3 shows an example of calculating $MergeQ$ score by merging vertices A and B into a vertex C in a certain level. Note that A and B represent two groups of vertices contracted before.

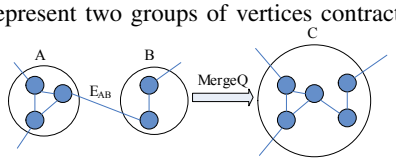

Fig. 3. An example of calculating $MergeQ$ score.

According to the Eq.(I) presented in Section III-A, we deduct the formula of calculating $MergeQ$ score as follows. Considering the example above,

$$Q_A = \frac{E_A}{2|E|} - (\frac{D_A}{2|E|})^2, Q_B = \frac{E_B}{2|E|} - (\frac{D_B}{2|E|})^2.$$

Once vertices A and B are merged, then

$$\begin{aligned} Q_C &= \frac{E_C}{2|E|} - (\frac{D_C}{2|E|})^2 \\ &= \frac{E_A + E_B + 2E_{AB}}{2|E|} - (\frac{D_A + D_B}{2|E|})^2. \end{aligned}$$

Therefore, we can calculate the $MergeQ$ score of vertices A and B as below,

$$\begin{aligned} MergeQ &= Q_C - Q_A - Q_B \\ &= \frac{1}{|E|}(E_{AB} - \frac{D_A D_B}{2|E|}), \end{aligned}$$

where

$$E_{AB} = \sum_{v \in A, w \in B} A_{vw}, D_A = \sum_{v \in A} d_v, D_B = \sum_{w \in B} d_w.$$

(A is adjacency matrix of a network and d is degree of a vertex.)

Calculating $MergeQ$ score is time-consuming. In MOME, we develop an improved adjacency list with a hash heap used to fast locate the pair of vertices we need. For efficacy, the degree of each vertex D_v in the original graph (or the sum of the degrees of vertices contracted into a vertex D_C in a certain level) is recorded in the adjacency list as well. It has been proven to speed up the calculation of $MergeQ$ score in experiments. Due to limited space, we omit the description of the data structure here.

B. Uncoarsening & Refinement Phase

The success of the multilevel paradigm is primarily due to good coordination between the coarsening and the uncoarsening & refinement phases. Here, we adopt a greedy version of

FM algorithm [23] to refine the initial partitioning performed by the coarsening procedure. The illustration of uncoarsening & refinement in MOME is shown in Algorithm 3.

Algorithm 3 The procedure of uncoarsening in MOME.

Input: A set of successive approximations GC.
Output: A set of communities.
Method: Uncoarsening(GC).

1) FOR each graph G in GC
2) Find out the borderline of communities in G, and store the vertices into a set $Borderline$;
3) For each vertex v in $Borderline$
4) Calculate v's $MoveQ$ scores, and select a community C with maximal value;
5) If $maxMoveQ > 0$
6) Move the vertex v into C;
7) Update $Borderline$ with v's adjacent vertices;
8) If the iterative times reach predefined threshold
9) break;
10) Return the original graph and its division.

We stepwisely project the initial partitioning back to the original graph via a set of successive approximations in GC, during which the refinement processes are performed to refine the communities. In each round of uncoarsening, MOME first identifies the borderlines of communities projected from the previous approximation. Then, it calculates $MoveQ$ score for each vertex v on borderlines. $MoveQ$ score denotes a modularity gain after one vertex in a community is moved to another community. Similar to the calculation of $MergeQ$, the biggest one is picked out, and the vertex v is moved to the corresponding community C when the score is greater than zero. Therefore, MOME always contributes a positive modularity Q gain locally in each uncoarsening step as well. Finally, the adjacency list of the input graph need to be updated.

Discovering borderlines for communities is time-consuming because we have to test each vertex by frequently scanning the adjacency list. To improve the performance of discovering borderlines in each round, we iteratively reuse the identified ones in the previous approximation graph G_{i+1} to help find out the new ones of the current approximation graph G_i. For the graph G_i, all vertices on the new borderlines must be contracted in the borderlines of the graph G_{i+1}. Accordingly, we can further narrow the search within the scope of those projected vertices contracted as the borderlines of the graph G_{i+1}, which reduces much more redundant scanning over the adjacency list.

Similar to the coarsening process, the kernel of the uncoarsening & refinement procedure involves calculating $MoveQ$ scores between the adjacent vertices and moving the ones with the largest scores to the corresponding communities in each level. An example of calculating the $MoveQ$ score has been given in Figure 4.

Based on the Eq.(I), we give the formula of calculating $MoveQ$ score likewise. Considering the example above,

$$Q_A = \frac{E_A}{2|E|} - (\frac{D_A}{2|E|})^2, Q_B = \frac{E_B}{2|E|} - (\frac{D_B}{2|E|})^2.$$

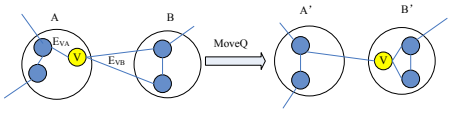

Fig. 4. An example of calculating $MoveQ$ score.

Once the yellow vertex V moves from A to B, then

$$Q_{A'} = \frac{E_A - 2E_{VA} - E_V}{2|E|} - (\frac{D_A - D_V}{2|E|})^2,$$

$$Q_{B'} = \frac{E_B + 2E_{VB} + E_V}{2|E|} - (\frac{D_B + D_V}{2|E|})^2.$$

Therefore, we can calculate the $MoveQ$ score as below,

$$\begin{aligned} MoveQ &= Q_{A'} + Q_{B'} - Q_A - Q_B \\ &= \frac{E_{VB} - E_{VA}}{|E|} + \frac{1}{2|E|^2}(D_A D_V - D_V^2 - D_B D_V), \end{aligned}$$

where E_{VA} is the number of edges between the vertex V and the community A, and E_{VB} is the number of edges between the vertex V and the community B.

C. Optimization

Observed from the coarsening procedure, MOME collapses vertices together using globally random-locally greedy strategy. In case that vertices on borderlines of real communities are visited in early stage, it is likely to lead to a few bad merging results that include vertices across communities. For example in Figure 5, suppose that the yellow vertex V is first visited in a level. Thus, its adjacent vertex A is picked out to be merged as their $MergeQ$ score is maximum, though A does not fall into V's community in nature. Under this condition, coarsening phase might affect the final quality of graph partitioning.

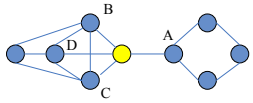

Fig. 5. An example of bad merging result.

We propose an optimization technique to improve globally random-locally greedy strategy in MOME. In Line 5 of Algorithm 2, we add one more condition to secure that only "similar" vertices (e.g., their similarity is beyond a predefined threshold or they are the most similar pair around a vertex) are permitted to merge. Here, the similarity is defined using a widely accepted idea in social network that two persons are similar if both share many friends. This restriction can efficiently avoid visiting vertices on the borderlines too early.

An example of calculating similarity has been given in Figure 6. Suppose vertices A and B are picked out because their $MergeQ$ score is maximum among all pairs. Also, N is the set of adjacent vertices around A excluding B. (For example, the circle with dashed line in N denotes A's adjacent vertices in a certain level) Here, we need to evaluate the similarity between A and B by considering A's adjacent vertices (i.e., N) in the graph. Based on the Eq.(I), we present the formula as follows.

Name	Description
SPEC	Spectral algorithm presented in [4]
SPEC-tuned	Spectral algorithm improved by Newman for fast convergence of matrix-vector multiplication
MOME	Modularity-based multilevel graph clustering
MOME-optimized	Modularity-based multilevel graph clustering using the optimization technique in Section IV-C

TABLE I

THE DICTIONARY OF SHORT NAME OF DISTINCT IMPLEMENTATIONS AND ALGORITHMS.

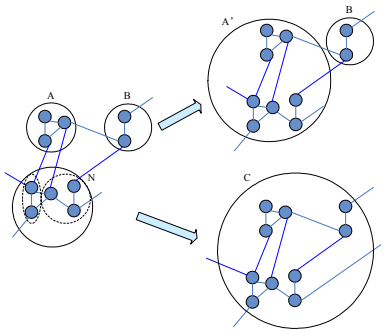

Fig. 6. An example of calculating similarity.

$$Q_A = \frac{E_A}{2|E|} - (\frac{D_A}{2|E|})^2, Q_B = \frac{E_B}{2|E|} - (\frac{D_B}{2|E|})^2,$$

$$Q_N = \frac{E_N}{2|E|} - (\frac{D_N}{2|E|})^2.$$

First, we need to quantify the strength for A's community excluding B. Suppose they are merged into a new vertex A' accordingly. Thus,

$$\begin{aligned} Q_{A'} &= \frac{E_{A'}}{2|E|} - (\frac{D_{A'}}{2|E|})^2 \\ &= \frac{E_A + E_N + 2E_{AN}}{2|E|} - (\frac{D_A + D_N}{2|E|})^2. \end{aligned}$$

Next, we further quantify the strength for A's community including B. They are merged into a new vertex H likewise. Thus,

$$\begin{aligned} Q_C &= \frac{E_C}{2|E|} - (\frac{D_C}{2|E|})^2 \\ &= \frac{E_A + E_B + E_N + 2E_{AB} + 2E_{AN} + 2E_{BN}}{2|E|} \\ &\quad -(\frac{D_A + D_B + D_N}{2|E|})^2. \end{aligned}$$

Finally, we can infer similarity sim that

$$\begin{aligned} sim &= Q_C - Q_{A'} \\ &= \frac{E_B + 2E_{AB} + 2E_{BN}}{2|E|} - \frac{2D_AD_B + 2D_BD_N + D_B^2}{4|E|^2}. \end{aligned}$$

V. EXPERIMENTS

In this section, we compare MOME with the latest modularity-based spectral algorithms [3], [4] in efficacy and quality. Prof. Newman at University of Michigan kindly provided us the source codes of their spectral algorithm and its improvement. Our algorithms are implemented in C++. For convenience to identify distinct implementations and algorithms, we uniformly name them in Table I. All the experiments are performed on a Pentium Celeron M 1.5GHz Laptop with 1GB RAM. The operating system platform is Window XP.

A. Real World Datasets

To make an overall comparison, we conduct the experiments on distinct real world networks shown in Table II, whose scales of vertices vary from hundreds to millions.

Network	vertices	edges	Application
Zachary's karate club	34	78	Social network
Football games	115	616	Social network
NIPS coauthorships	1,063	4,177	Social network
Protein interactions	1,458	1,948	Biological network
KDD citations	27,400	352,504	Social network
WWW	325,729	1,090,107	World Wide Web
IMDB	1,019,836	3,489,755	Social network

TABLE II

THE DESCRIPTION OF THE REAL WORLD DATASETS WE USED AND THEIR SCALES.

B. Performance Evaluation

First, we deliver performance comparison amongst the algorithms indicated in Table I, which has been illustrated in Figure 7. Note that Y axes is in $\log_{10} T$ scale for the convenience of observation. Seen from the chart, for small and medium networks, i.e, Football games, NIPS coauthorships, Protein interactions, and KDD citations, MOME and MOME-optimized perform up to 3 orders of magnitude faster than others. And for very large networks, i.e., WWW and IMDB, both still work efficiently while the others stop due to memory exhausting. Particularly, MOME and MOME-optimized only take about 3 and 10 minutes respectively to finish community discovery on the IMDB network, which is the largest network in our experiments with more than 1 million vertices and 3 million edges.

Here, the huge contrast in runtime between SPEC and SPEC-tuned attracts our attention. For the network of Football games, SPEC-tuned runs to completion in 47 milliseconds

Networks	SPEC	SPEC-tuned	MOME	MOME-optimized	clusters by SPEC	clusters by MOME
Zachary's karate club	0.4188	0.4188	**0.4198**	**0.4198**	4	**4**
Football games	0.602	0.602	**0.606**	**0.606**	10	**10**
NIPS coauthorships	0.854	0.854	**0.883**	**0.885**	54	**27**
Protein interactions	0.798	0.798	**0.822**	**0.822**	63	**31**
KDD citations	0.639	N/A	**0.661**	**0.667**	28	**24**
WWW	N/A	N/A	**0.947**	**0.946**	N/A	**312**
IMDB	N/A	N/A	**0.834**	**0.835**	N/A	**243**

TABLE III
MODULARITY Q SCORES COMPARISON.

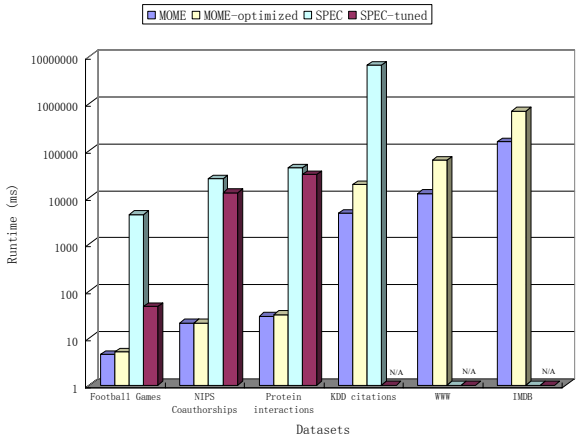

Fig. 7. Performance comparison.

at nearly 100 times faster than SPEC, while for NIPS coauthorships, both finish partitioning at approximately equal rate in about 13 seconds and 25 seconds respectively. After consulting with Newman, we learned that the performance of power method used in SPEC would fluctuate in terms of the convergence rate for matrix-vector multiplication. To achieve fast convergence in SPEC-tuned, Newman further utilized CLAPACK, a C language version of LAPACK of linear algebra package, to do the multiplication. Our experimental results also indicated that SPEC-tuned performs more stable by comparing to SPEC.

In our experiments, for the example of KDD citations, we also find that SPEC seems to be more scalable than SPEC-tuned. The reason is that SPEC implements power method based on adjacency list data structure, while SPEC-tuned has to physically maintain a matrix (2-dimensional array) in memory for the CLAPACK. It is extremely costly in space when we use matrix instead of adjacent list for graph clustering.

C. Quality Measure

Next, we further measure the quality of discovered communities and compare their Q scores from the real world networks.

MOME can split the Zachary's karate club into the 2 right-on communities just like "forecast". Furthermore, we may continue to discover 4 sub-communities by maximizing the modularity Q, which possibly indicate the potential ingroups. The division of Zachary's karate club has been shown in Figure 8. The vertices of 1 and 33 represent the club's administrator and its principal karate teacher respectively. The friendships in the 4 sub-communities are illustrated by the thick lines.

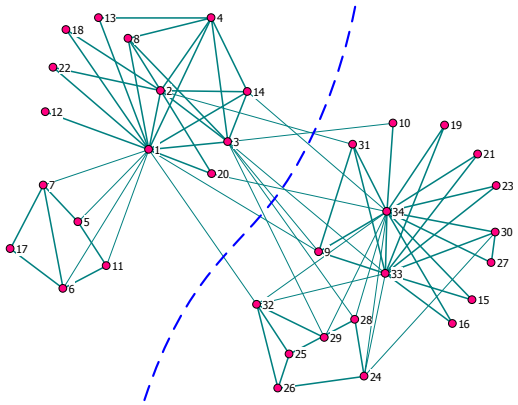

Fig. 8. Zachary's karate club.

Figure 9 shows the discovered communities from the Football games by MOME. For convenience to understand the difference between real conferences and discovered communities, we represent the teams in the same conference with a uniform mark, i.e., same color, in the figure. Each discovered community is depicted as a group of vertices connected by thick lines. Seen from the figure, about 91% teams are accurately divided into the real conferences. The rest of teams seem to be the noise points in the dataset because they would play much more games with the teams from other conferences than expectation.

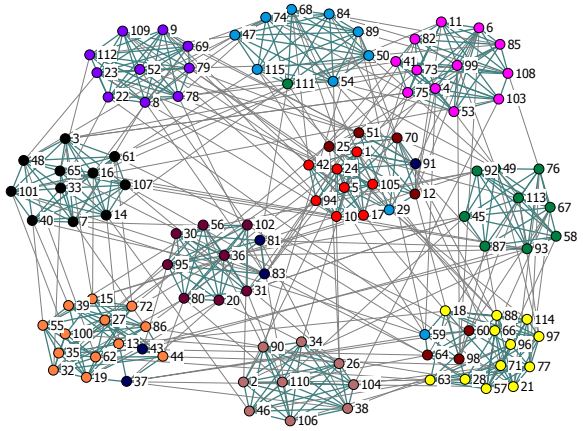

Fig. 9. Football games.

For the rest of the real world networks, we can not provide an exact division to evaluate the accuracy of those algorithms. Also, those networks are too large to be clearly shown in the figures. However, a high Q score might empirically denote a good division of network. Accordingly, we measure the quality of discovered communities by comparing their Q scores. Furthermore, the number of communities is attached as reference. Table III shows the overall comparison. Seen from this table, for small and medium networks, MOME and MOME-optimized would make the divisions slightly better than the spectral algorithms do. But for large networks, both perform overwhelmingly efficiently.

We further evaluate the stability between MOME and MOME-optimized. This experiment was run 20 times for both methods under the same condition and environment. Figure 10 indicates that MOME-optimized performs more stable on the Q score when we divide the Football games network into communities. The optimization technique in Section IV-C is proven by practical experiments, which is able to improve the bias of the random visiting order, which avoids coarsening the vertices on the borderlines of communities too early.

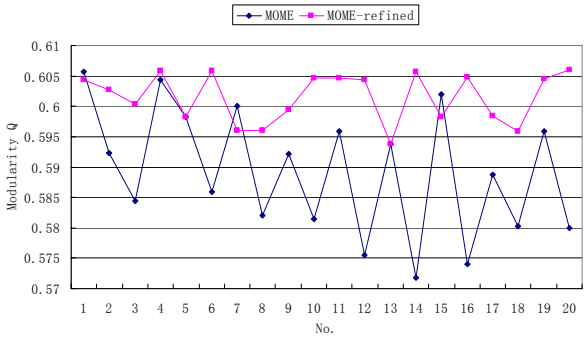

Fig. 10. Stability test.

VI. CONCLUSION

In this paper, based on the cost function of modularity Q, we proposed an efficient multilevel algorithm MOME to extract community structures from networks, especially for very large ones. Furthermore, we investigated that the optimization of the Q function could be expressed by a multilevel problem. The experimental results showed that MOME had a better performance and higher quality than other competing methods.

REFERENCES

[1] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, p. 026113, 2004. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0308217

[2] T. N. Bui and C. Jones, "A heuristic for reducing fill-in in sparse matrix factorization." in *PPSC*, 1993, pp. 445–452.

[3] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E*, vol. 74, p. 036104, 2006. [Online]. Available: doi:10.1103/PhysRevE.74.036104

[4] ——, "Modularity and community structure in networks," *PROC.NATL.ACAD.SCI.USA*, vol. 103, p. 8577, 2006. [Online]. Available: doi:10.1073/pnas.0601602103

[5] Y.-C. Wei and C.-K. Cheng, "Towards efficient hierarchical designs by ratio cut partitioning," 1989.

[6] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

[7] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon, "A min-max cut algorithm for graph partitioning and data clustering," in *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 107–114.

[8] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, p. 066111, 2004. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0408187

[9] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, p. 066133, 2004. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0309508

[10] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, and D. Wagner, "Maximizing modularity is hard," 2006. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:physics/0608255

[11] L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas, "Comparing community structure identification," Oct 2005. [Online]. Available: http://arxiv.org/abs/cond-mat/0505245

[12] S. White and P. Smyth, "A spectral clustering approach to finding communities in graph." in *SDM*, 2005.

[13] R. Guimera, M. Sales-Pardo, and L. A. N. Amaral, "Modularity from fluctuations in random graphs and complex networks," *Physical Review E*, vol. 70, p. 025101, 2004. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0403660

[14] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Physical Review E*, vol. 74, p. 016110, 2006. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0603718

[15] J. Duch and A. Arenas, "Community detection in complex networks using extremal optimization," *Physical Review E*, vol. 72, p. 027104, 2005. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0501368

[16] G. W. Flake, S. Lawrence, and C. L. Giles, "Efficient identification of web communities," in *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2000, pp. 150–160.

[17] B. Hendrickson and R. Leland, "A multilevel algorithm for partitioning graphs," in *Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*. New York, NY, USA: ACM Press, 1995, p. 28.

[18] Walshaw and Cross, "Mesh partitioning: A multilevel balancing and refinement algorithm," *SIJSSC: SIAM Journal on Scientific and Statistical Computing, apparently renamed SIAM Journal on Scientific Computing*, vol. 22, 2000. [Online]. Available: citeseer.ist.psu.edu/walshaw00mesh.html

[19] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, 1998.

[20] ——, *MeTis 4.0: Unstrctured Graph Partitioning and Sparse Matrix Ordering System*, 1998. [Online]. Available: http://www.cs.umn.edu/?metis

[21] A. Abou-Rjeili and G. Karypis, "Multilevel algorithms for partitioning power-law graphs," 2006.

[22] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Sys. Tech. J.*, vol. 49, no. 2, pp. 291–308, 1970.

[23] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *25 years of DAC: Papers on Twenty-five years of electronic design automation*. New York, NY, USA: ACM Press, 1988, pp. 241–247.