Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 934301, 13 pages
http://dx.doi.org/10.1155/2015/934301

Research Article

A Parallel Community Structure Mining Method in Big Social Networks

Songchang Jin,[1,2] Philip S. Yu,[2] Shudong Li,[1] and Shuqiang Yang[1]

[1]*College of Computer, National University of Defense Technology, Changsha, Hunan 410073, China*
[2]*Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA*

Correspondence should be addressed to Songchang Jin; jinsongchang87@gmail.com

Received 5 July 2014; Accepted 2 August 2014

Academic Editor: Haipeng Peng

Community structure plays a key role in analyzing network features and helping people to dig out valuable hidden information. However, how to discover the hidden community structures is one of the biggest challenges in social network analysis, especially when the network size swells to a high level. Infomap is a top-class algorithm in nonoverlapping community structure detection. However, it is designed for single processor. When tackling large networks, its limited scalability makes it less effective in fully utilizing server resources. In this paper, based on infomap, we develop a scalable parallel nonoverlapping community detection method, Pinfomr (parallel Infomap with MapReduce), which utilizes the MapReduce framework to solve the two problems. Experiments on artificial networks and real datasets show that our parallel method has satisfying performance and scalability.

1. Introduction

A few common properties in many complex networks have been discovered: small-world property, scale-free feature, and community structure pattern [1–4]. Community structure is playing a key role in the formation and function of these networks [5]. However, it is one grave challenge in complex systems [6].

Current social networks have jumped to millions even billions of nodes [7]. Take Facebook for example, its monthly active user has reached 1.16 billion [8]. However, due to computational costs, traditional community discovery algorithms are willing, but unable to tackle such huge complex networks. So, it is necessary to implement a fast and scalable approach to detect communities in big social networks.

Network partitioning is NP-complete [9]. Partitioning a network into approximately equal sized components while minimizing the number of edges between different components is extremely important in parallel computing [10]. For example, parallelizing many applications involves the problem of assigning data or processes evenly to processors, while minimizing the communication traffic. However, when the network size reaches a certain level, direct segmentation on the original networks is not realistic, and there exist deficiencies of convergence rate of traditional algorithms.

Nowadays, mainstream servers are configured with high performance hardware. Empirical studies [11] have showed that infomap [12] is a top-class standalone algorithm for nonoverlapping community detection. However, due to the limitations of technological level, processing capability of single core has encountered a bottleneck and the scalability of infomap is suffered as a consequence, that is, because it only utilizes one core or processor of the server. Besides, computing resource waste is an additional product of infomap running on multiprocessor server. How to improve the scalability of infomap and make full use of servers is an awkward subject.

Information science is shifting from computing-intensive to data-intensive [13] with the advent of the era of big data. Some novel parallel computing frameworks shine, in which MapReduce [14] is one of the best. In this paper, based on our previous work [15], we present a new scalable parallel community detection method coalescing several existing excellent techniques, such as infomap, k-shell decomposition, multilevel network partitioning, and MapReduce. A high-level description of our approach is as follows. First, we divide the whole network into a number of partitions and

the number of partitions is far less than that of community structures. To speed up the process, we develop an enhanced multilevel partitioning method. Next, with MapReduce, we run parallel method to mine the community structures simultaneously within the partitions. Finally, we collect the community structures together to form a final result.

Main contributions of this paper are as follows: (1) we propose a new model to mine community structure in big social networks. (2) We integrate k-shell decomposition theory with multilevel k-way partitioning algorithm to deal with peripheral nodes. (3) We implement a scalable and parallel infomap to uncover community structures and to improve resource utilization rate.

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

2. Preliminary Knowledge

2.1. Relevant Concepts. In this paper, we only study undirected networks, which can be mathematically described as G, consisting of node set V and edge set E; n represents the number of nodes, $v_i \in V$ represents a node, and $d(v_i)$ means its degree; m represents the number of edges and $e_{i,j}$ is the edge between v_i and v_j, where $0 < i \neq j \leq n$.

Infomap is based on information-theory. So some information-theoretic concepts are briefly reviewed here. In information theory, the information contained in a distribution is called entropy. For a discrete random variable $X = \{x_1, x_2, \ldots, x_n\}$ with a probability distribution $P(X)$, its entropy is

$$H(X) = -\sum_{x \in X} p(x) \log p(x). \tag{1}$$

Mutual information calibrates the shared information between two distributions, X and Y. We define $P(X, Y)$ as the joint probability of X and Y. $P_x(X)$ and $P_y(Y)$ are defined as marginal probability distribution of X and Y, respectively. Then, mutual information of X and Y is

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \frac{p(x, y)}{p_x(x)\, p_y(y)}. \tag{2}$$

Normalized mutual information (NMI) is often used for evaluating clustering result, information retrieval, feature selection, and so forth. Value range of NMI is [0, 1] and when X and Y are the same, NMI$(X; Y)$ equals 1.0. Consider

$$\mathrm{NMI}(X;Y) = \frac{I(X;Y)}{\sqrt{H(X)\, H(Y)}}. \tag{3}$$

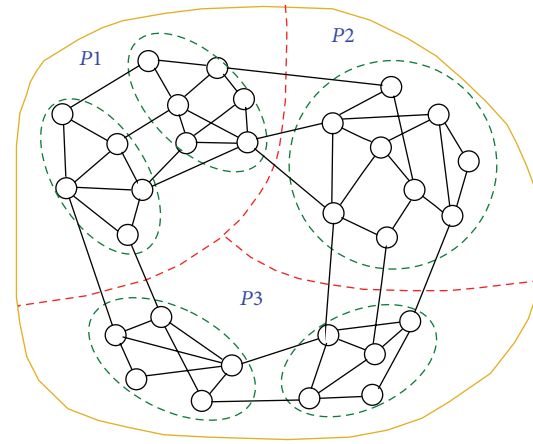


Figure 1: A network with 5 communities and 3 partitions.

2.2. K-Shell Decomposition Theory. K-shell decomposition is a well-established method for analyzing the structure of large-scale networks [16–18]. In particular, it provides a method for identifying hierarchies in a network. It is assumed that importance of a node is not related to its degree but its location. The process assigns an integer index, ks, to each node, representing its location within the successive layers (k-shells) in the network. The ks index is a robust measure and the node ranking is not significantly influenced in the case of incomplete information. The k-core of a network G is the maximum subnetwork of G whose degree is no less than k. The k-shell of G is the set of all nodes belonging to the k-core of G but not to the $(k + 1)$-core.

Nodes are assigned to k-shells based on their remaining degree, which is obtained by successive pruning of nodes with degree smaller than the ks value of the current layer. The decomposition process starts by removing all nodes with degree $d = 1$. After that, some nodes may be left with one link. We then prune the system iteratively until there is no node left with $d = 1$ in the network. The removed nodes, along with the corresponding links, form a k shell with index $ks = 1$. In a similar fashion, we iteratively remove the next k-shell, where $ks = 2$, and continue to remove higher k shells until all nodes are removed. As a result, each node is associated with one ks index, and the network can be viewed as the collection of all k shells. ks value of a node can be very different from its degree. In Figure 2, we can see that v_9 has 7 neighbors with $ks(v_9) = 1$. Figure 5 is the result of Figure 2 of which peripheral nodes are processed.

2.3. Multilevel k-Way Partitioning Method. Partitioning the node set V of a network G into k disjoint subsets $\{V_1, V_2, \ldots, V_k\}$ is called a k-way partitioning of V. Each subset and the edges within the subset constitute a *partition* of G. Figure 1 shows a simple network with 5 communities surrounded by the dotted circles and 3 partitions.

Definition 1 (effective edge lost ratio). An edge whose endpoints are in the same community, that is, intracommunity edge, is called an effective edge. If the endpoints of an effective

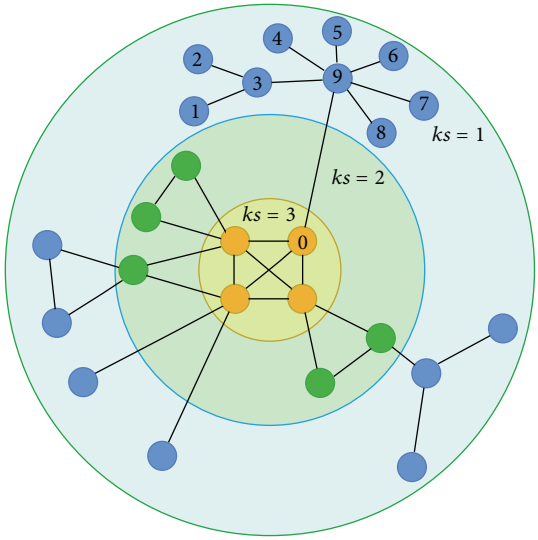


Figure 2: Schematic representation of k-shell decomposition.

edge are divided into different partitions, then we call it an effective edge lost. The effective edge lost ratio is the percentage of the effective edge lost divided by the total number of edges in the network.

In Figure 1, $e_{3,4}$ is an effective lost, whereas $e_{1,2}$ is not. It is apparent that effective edge lost plays a more important role in the community detection than the edges connecting nodes in different communities and being cut off by partition boundaries.

A number of high-quality and computationally efficient graph partitioning methods have been proposed and multilevel graph partitioning algorithms [9, 19, 20] are currently considered to be a start-of-the-art method and being extensively used. Here, we optimize the multilevel k-way partitioning method proposed by Abou-Rjeili and Karypis [21] to partition the power law networks.

From Figure 3, we can see that multilevel k-way partitioning method consists of coarsening phase, initial partitioning phase, and uncoarsening and refinement phase. Instead of trying to partition directly on the original graph G_0, multilevel partitioning algorithms first obtain a sequence of successive approximations, such as G_1, G_2, and G_3 in the coarsening phase, of the original graph. Each of these approximations represents smaller than the size of the original graph. This process continues until a level of approximation is reached where the graph contains only a few hundreds of nodes. At this point, partitioning algorithms begin to compute partitions of that graph, corresponding to the five small partitions of G_3 in the initial partitioning phase, and since the graph is quite small, even simple algorithms are able to take it over and get reasonably good results, such as K-L [22]. And there is a parameter used to control the balance of partitions. The final step, uncoarsening and refinement phase, is to map the partitions of the smallest graph G_3 onto the original graph G_0 and to derive final partitions.

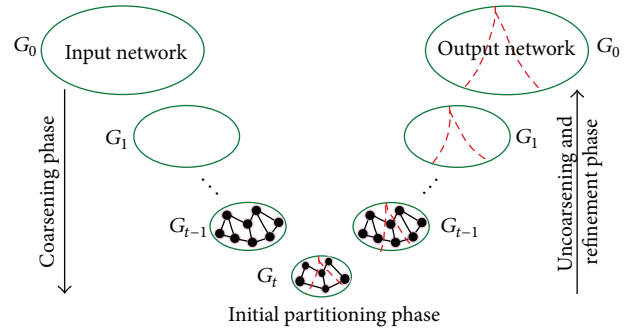


Figure 3: A schematic diagram of multilevel k-way partitioning method.

2.4. Infomap. In this paper, we continue our work on the information theoretic community detection model-infomap. First, we briefly review the model. It utilizes the duality between compressing a data set and detecting and extracting significant patterns or structures within the data, which is a statistical concept known as minimum description length statistics [23]. A random walk, represented as a Markov process, is used as a proxy for information flow. For a community-structured network, when a random walker enters a community, it tends to stay in it for a long time and the possibility of moving out into another community is low.

In an undirected network, the random walk has a state $x(t) \in V$ at time t, indicating where it is. Then in next step, $t + 1$, the walker will move to v_j chosen at random from neighbors of v_i. To describe the state of random walker, a 2-level description model with Huffman coding is proposed. The first level encodes the communities and the second level encodes the nodes within the communities. Then we can use "community ID + node ID" to uniquely describe a particular node in the network. Huffman codes are prefix-free coding scheme and are optimally efficient for symbol-by-symbol encoding. It saves space by assigning short codewords to common events or objects and long codewords to rare ones, just as Morse code uses short codes for common letters and longer codes for rare ones. So, the path of the random walker can be described as a coding sequence.

Figure 4 is an example for illustrating the 2-level description method. Assuming there are 2 communities divided like Figure 4(b), then the code sequence for the random walk in Figure 4(a) is "***0*** *111 00 10 111 010 10 011 110 00 10 110* ***1011 1*** *00 01 10 00 11 10 01 1*." The underlined word "***0***" in bold format means random walk starting from C1. The underlined word "***1011 1***" in bold format means random walk leaving from C1 and entering C2. The description length of the sequence will be 50 bits and about 2.63 bits per step. But, in Figure 4(c), we will need 57 bits and 3.0 bits per step. Community division is obviously more reasonable in Figure 4(b) than in Figure 4(c), and the average description length in the former one is shorter than in the latter one. From the perspective of information theory, we know that smaller entropy corresponds to smaller uncertainty. Corresponding to the community detection, smaller entropy means smaller indistinctness or clearer community structure.

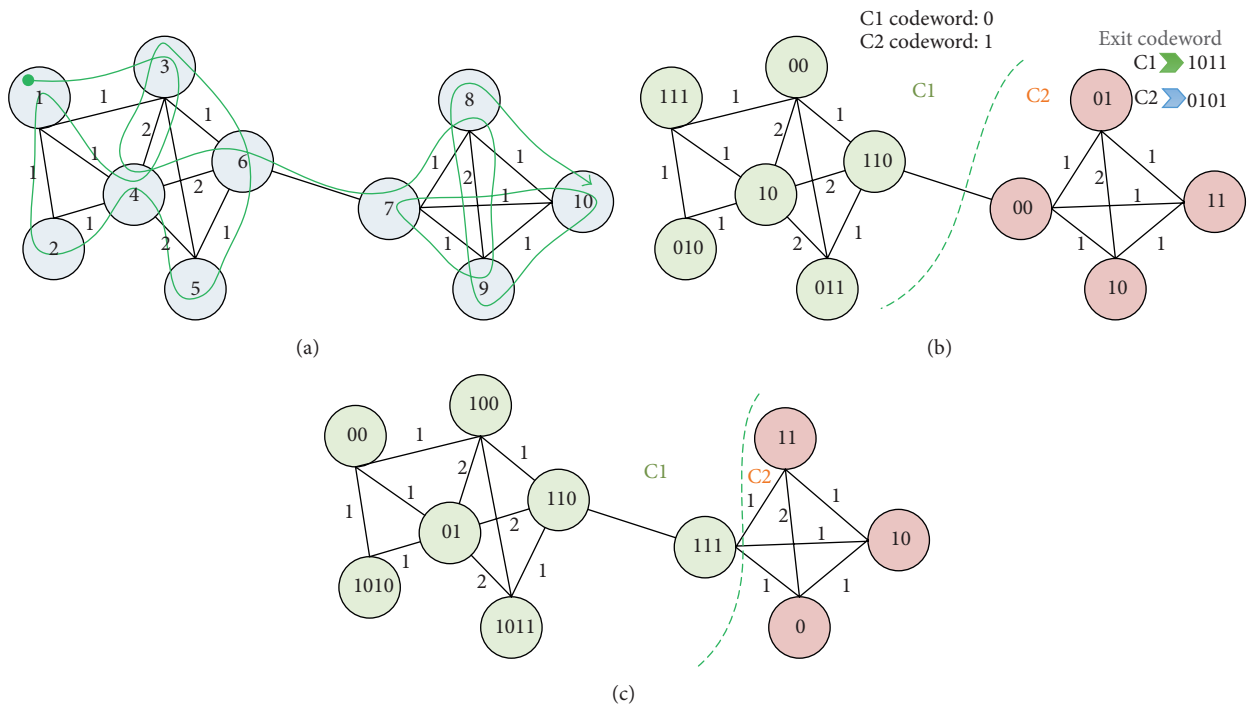


Figure 4: Random walk and 2-level Huffman coding on a network with two communities.

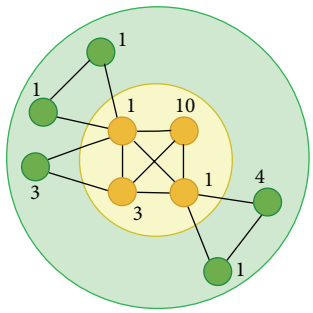


Figure 5: The result of Figure 2 without peripheral nodes. Integers indicate node weights.

3. Parallel Community Detection Method

3.1. Problem Statement. Assuming there is an optimal community division, M^*, in a community-structured network G. With M^*, the network G is divided into num^* communities, and the lower limit of average description code length is $L(M^*)$. According to the Shannon source coding theorem [24] and the Kraft's inequality [25], we know that, for any division pattern M, the average codeword length per source symbol, $L(M)$, for an optimal prefix-free code satisfies

$$H(X) \le L(M^*) \le L(M) \le H(X) + 1. \tag{4}$$

Obviously, calculating an endless random walk on a network to get $L(M)$ is not realistic. Fortunately, when randomly walking on a network endlessly, we will get a steady visit frequency for each node, and we can calculate that easily with many methods, such as PageRank. With the steady visit frequency distribution, we will be able to describe the state of the random walker easily. For $x(t+1) \in \{\text{neighbor}(x(t))\}$, the probability of $x(t+1)$ and $x(t)$ being in the same community is p_{within} and the probability of being in different communities is $q_{k,\text{out}}$, where $x(t+1)$ belongs to community k. Then the $L(M)$ can be described as

$$L(M) = q_{\text{out}} H(Q) + \sum_{i=1}^{\text{num}} p^i_{\text{within}} H\left(P^i\right), \tag{5}$$

where q_{out} means the probability of moving out from the current community and $q_{\text{out}} = \sum_{i=1}^{\text{num}} q_{i,\text{out}}$. $H(Q)$ is the average description length of nodes in all communities, and it can be expressed as

$$\begin{aligned} H(Q) &= \sum_{i=1}^{\text{num}} H_i(Q) \\ &= -\sum_{i=1}^{\text{num}} \frac{q_{i,\text{out}}}{\sum_{j=1}^{\text{num}} q_{j,\text{out}}} \log \frac{q_{i,\text{out}}}{\sum_{j=1}^{\text{num}} q_{j,\text{out}}}. \end{aligned} \tag{6}$$

With the probability p_a or $p(a)$ to visit the node v_a, p^i_{within} represents the probability of staying in the current community during the next step, and $p^i_{\text{within}} = \sum_{v_a \in C_i} p_a + q_{i,\text{out}}$. $H(P^i)$

```
(1) method Map(nid n, node v)
(2)   p ← p_v/|v.adjacencyList| //(8)
(3)   emit(nid n, v) //pass along network structure
(4)   for all nodeid m ∈ v.adjacencyList do
(5)     emit(nid m, p) //pass pagerank contribution to neighbors
(6)   endfor
(7) method Reduce(nid m,[p_1, p_2, ...])
(8)   v ← φ
(9)   for all p ∈ counts[p_1, p_2, ...] do
(10)    if isNode(p) then
(11)      v ← p //recover local network structure
(12)    else
(13)      s ← s + p //sums pagerank contributions
(14)    endif
(15)  endfor
(16)  p_v ← s
(17)  emit(nid m, v)
```

Algorithm 1: Steady visiting probability vector calculation on MapReduce (VPC).

expresses the infor mation entropy of the visiting probability of the nodes in the community C_i, which can be written as

$$H\left(P^i\right) = -\frac{q_{i,\text{out}}}{q_{i,\text{out}} + \sum_{v(b)\in C_i} p(b)} \times \log \frac{q_{i,out}}{q_{i,\text{out}} + \sum_{v(b)\in C_i} p(b)} - \sum_{v(a)\in C_i} \frac{p(a)}{q_{i,\text{out}} + \sum_{v(b)\in C_i} p(b)} \times \log \frac{p(a)}{q_{i,\text{out}} + \sum_{v(b)\in C_i} p(b)}. \tag{7}$$

For the NP-complete challenge, we cannot achieve the global optimal division pattern M^* by direct computing on a big social network. However, we can archive a set of local optima to approximate M^* by partitioning the network into small subnetworks (partitions) and tackling them independently with MapReduce. Then, the issue will become how to discover optimal division pattern M_i^* in partition P_i and get final $M^* = \bigcup_{i=1}^{k}\{M_i^*\}$. For P_i, it would be sufficient to calculate the $L(M_i)$ for different M_is and pick up the one with shortest description length as M_i^*. Finally, we get a community set $C = \{C_1, C_2, \ldots, C_k\}$, where $C_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,k1}\}$ corresponds to P_i, $\cup_{i=1}^{k_i} C_i = V$ and $C_i \cap C_j = \emptyset$, where $0 < i \neq j \leq n$ and $|P| \ll |C|$.

3.2. Procedure of the Parallel Community Detection. For the convenience of illustration, we adopt Figure 1 to start this section and assume that the amount of partitions is far less than that of communities in big social networks. There are 3 stages in the parallel community detecting process.

In the first stage, we calculate the steady visiting probability of all nodes (shown in Algorithm 1). Here, we modify the traditional PageRank, which is used to deal with directed networks and run a iterative MapReduce-based version to get the global steady visit probability vector. In each iteration, visit probability of v_a is (since there is no teleport and link sink in undirected networks, we set $\tau = 0$)

$$p_a = \tau \times \frac{1}{n} + (1-\tau) \times \sum_{v_i \in \{\text{neighbor}(v_a)\}} \frac{p_i}{d_i}. \tag{8}$$

Second, we use multilevel k-way partitioning method enhanced by k-shell decomposition method to divide a big social network into k approximately equal sized disjoint partitions ($P1$, $P2$, and $P3$ in Figure 1). Edges cut off by partition boundaries will be discarded. As networks studied here are sparse and with community structure, *edge cut (lost) ratio* will be low. However, partitioning method has a decisive influence on the final community detection effectiveness which will be explained with experiments in Section 4. A matching of a network is a set of edges and no two edges in it share a same node. To coarsen a network, a commonly used method is to collapse the node pairs forming the matching, such as random matching, heavy-edge matching, and maximum weighted matching. However, it shrinks at a slow rate and does not consider the relative importance of nodes in different locations. We all know that there is a large number of low-degree and low *ks* value nodes in power law networks, so we can turn this characteristic into revenue. Here, we use the k-shell decomposition to merge the peripheral nodes with high speed and more accurate performance during the coarsening phase (shown in Algorithm 2).

In the last stage, parallel community detection method is carried out on all partitions (such as the 3 partitions in Figure 1) and all partitions are tackled independently. When the parallel process finished, each partition generates

```
(1) set int id[n + 1] = {0, 1, 2, ..., n}
(2) while true, do
(3)   for all node v_i ∈ V, do
(4)     if d[i] == 1, then
(5)       tmpV.add(v_i) //a list to store nodes with ks = 1
(6)     endif
(7)   endfor
(8)   if tmpV.isEmpty(), then
(9)     break //process finished
(10)  endif
(11)  for all node v_j ∈ tmpV, do
(12)    if flag[j]! = true, then //v(j) hasn't been annexed
(13)      k = getID(v_j.getDiffNeighbor()) //get rid off v(j) itself
(14)      id[j] = k //assign new id to v(j) annexed by v(k)
(15)      w_k = w_k + 1
(16)      updateNeighbor(v_k, j) //replace v(j) with v(k) in v(k)'s neighbor
(17)      d[k] = d[k] − 1
(18)      d[j] = d[j] − 1 = 0
(19)      flag[j] = flag[k] = true
(20)    endif
(21)  endfor
(22) endwhile
```

Algorithm 2: Amalgamation of peripheral nodes in coarsening phase.

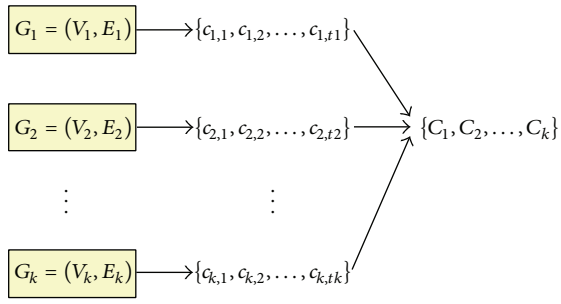


Figure 6: A schematic diagram of MapReduce process for community detection.

a community set (such as the 2 communities in $P1$ in Figure 1). Combining all the community sets together will be the final result. Figure 6 gives a straightforward statement of this process and more detail is shown in Algorithm 3. At the beginning of this stage, we treat each node as a community and then use a bottom-up approach with greedy scheme to (1) find out community pairs that can minimize $\Delta L(M)$ and (2) merge them to form new communities.

4. Experiments and Analysis

In this section, we conduct several experiments and analyze the results. All experiments are running on the Hadoop-1.1.1 cluster of Antivision Software Ltd. The cluster consists of 20 PowerEdge R320 servers (Intel Xeon CPU E5-1410 @2.80 GHz, memory 8 GB) with 64-bit NeoKylin Linux OS, and servers are connected by a Cisco 3750G-48TS-S switch. Data sets are shown in Table 1, including artificial networks and real networks.

All artificial networks used here are generated by LFR benchmark. In LFR, some parameters give us a direct control on network properties: network size (n), degree distribution (γ, $d_{\max}$, avg(d)), and community structure (β, mix) [26]. γ and β are exponents for degree and community size distributions, which range between $[2, 3]$ and $[1, 2]$, respectively. Mix is the ratio of edges connecting nodes from different communities divided by collective edges of all communities. For the average and balance, we set $\gamma = 2.5$ and $\beta = 1.5$ for artificial networks.

4.1. Accuracy Experiments. In accuracy experiments, we compare our method, Pinfomr, with two top-class methods, Louvain algorithm [27] and OSLOM algorithm [28], on different data sets and with different partition numbers. The data sets used are $D0$, $D1$, and $D2$, and result is shown in Figure 7, where $|P|$ means partition number. The situation when $C = 1$ or $|C| = n$ is defined by us as no community structure and NMI in this case is set to 0, but the case when $|C|$ is close to n is discarded. Taking for instance Louvain in Figure 7(a), when mix = 0.75, $|C|/n = 0.373$ and we conclude that a community just contains 3 nodes averagely. From the design of LFR, we know that when mix value is too high, such as higher than 0.75, there will be no obvious community structures, and the network will not be a power law network but more like a random network which is not the focus here.

```
(1) Initialization-global variables
(2)   nc = |V_t| //number of communities
(3)   L_0 = 0
(4)   for i from 1 to |V_t|, do
(5)     cid[i] = i //community ID of v_i
(6)     L_0 = L_0 + H(v_i)
(7)   endfor
(8) method Map(node v,adjacencyList)
(9) while δ < 0, do
(10)  for i ∈ [1, |V_t|], do
(11)    lock[i] = false //able to merge in current iteration
(12)  endfor
(13)  for i ∈ [1, |V_t|], do
(14)    j = cid[random(|V_t|) + 1] //randomly select a node(community)
(15)    curNeighbor[ ] = getAdjacencyList(c_j)
(16)    δ = 0 //decrease of L
(17)    maxCID = 0 //community id which leads to minimum δ
(18)    for c_s in curNeighbor[ ], do
(19)      if L_{s,k} − L_0 < δ //if merge c_s and c_k
(20)        maxCID = s
(21)        δ = L_{s,k} − L_0
(22)      endif
(23)    endfor
(24)    if maxID ≠ 0 & δ ≠ 0, then
(25)      lock[s] = lock[k] = true
(26)      nc = nc − 1
(27)      L_0 = L_0 − δ
(28)      pid[s] = j
(29)    endif
(30)  endfor
(31) endwhile
(32) for i ∈ [1, nc], do
(33)   emit(i, getAdjacencyList(c_i))
(34) endfor
```

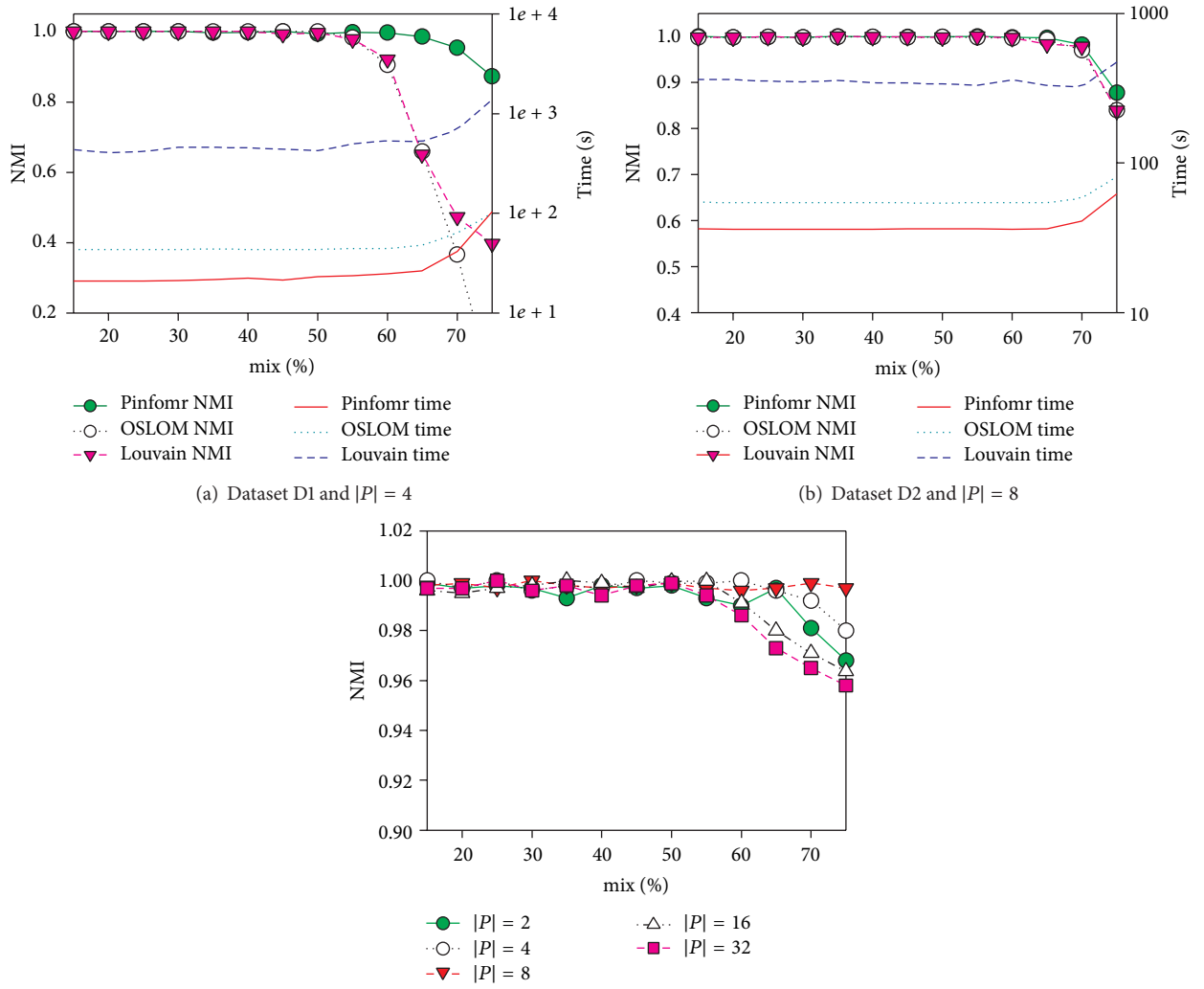
Algorithm 3: The map procedure of community detection on a subnetwork G_t.

Table 1: Data sets used in experiments (increment of mix is set to 0.05. $M = 10^6$).

Data set	n	m	avg(d)	γ	β	Size(C)	mix
LiveJournal	3,997,962	34,681,189	/	/	/	/	/
Youtube	1,134,890	2,987,624	/	/	/	/	/
Orkut	3,072,441	117,185,083	/	/	/	/	/
D0	0.1 M	/	45	2.5	1.5	[25, 120]	[0.1, 0.75]
D1	0.2 M	/	40	2.5	1.5	[25, 100]	[0.1, 0.75]
D2	0.4 M	/	40	2.5	1.5	[25, 100]	[0.1, 0.75]
D3	80,000	/	45	2.5	1.5	[25, 120]	[0.1, 0.75]
D4	0.2 M, 0.4 M, 0.8 M, 1.6 M	/	45	2.5	1.5	[25, 100]	0.45
D5	3.2 M, 6.4 M, 10 M	/	45	2.5	1.5	[25, 100]	0.45

Figure 7 indicates that Pinfomr is more stable and accurate than the others in uncovering community structures in power law networks. For running time, we can see that for the same data set, Louvain consumes the longest time and Pinfomr needs the shortest time. OSLOM requires a little more time than Pinfomr when mix parameter is not too big.

From Figure 7(c), we know that the NMI decrease as partition numbers increase, but the performance is excellent and stable when mix is less than 0.75, and NMI will maintain at about 1.0. Our results show that Pinfomr is able to achieve better results in a shorter period of time, although accompanied by some loss of performance.



(a) Dataset D1 and $|P| = 4$

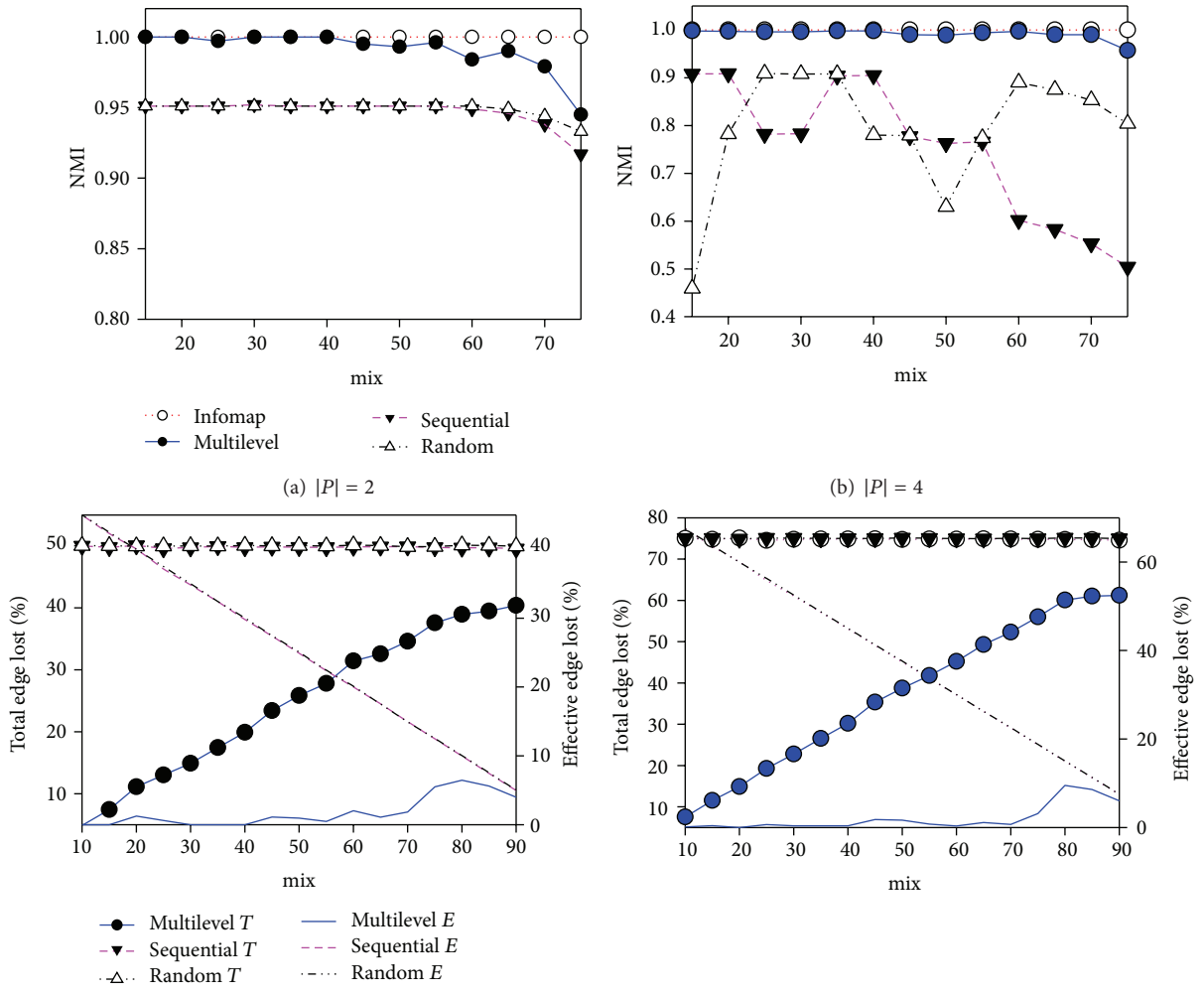
(b) Dataset D2 and $|P| = 8$

(c) Dataset D0 with different $|P|$

FIGURE 7: Accuracy and running time tests on different data sets.

4.2. Partitioning Experiment. In previous section, we have mentioned that the quality of partition will play a vital role in the final performance of parallel community detection. Therefore, we conduct experiment in this section to test the impact and effectiveness of different partitioning methods on Pinfomr.

We use two simple partitioning methods to compare with the improved multilevel k-way partitioning method. One is a sequential partitioning method dividing the network according to the storage order of the nodes and edges on the HDFS. The other one is a random matching partitioning method by randomly choosing nodes to generate a matching. For example, assuming that we bisect $G = (V, E)$ with $n = 20,000$ and $m = 300,000$ into $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, (1) when using sequential partitioning method, the first 10,000 nodes will be collected to form V_1 and links within V_1 will form E_1. The other nodes are left for V_2 and links within V_2 form E_2. (2) If we use the random matching method, we will randomly select 5,000 node pairs into V_1 and all links within V_1 will form E_1, and the process for G_2 is similar to G_1. Dividing a connected network into subnetworks or partitions will cause edge loss. Excellent partitioning methods will always try to walk through the slits between communities and avoid cutting off the effective edges. Here, we use the data set $D1$ to test performance of different partitioning methods with $|P| = 2$ and $|P| = 4$. In Figures 8(a) and 8(b), we can see that multilevel k-way partitioning method is stable and results of Pinfomr on it are very close to the results of infomap and also very close to the real results. From Figures 8(c) and 8(d), we get that, for multilevel k-way partitioning method, total edge loss ratio increases linearly along with the increase of mix parameter. It is easy to understand that, from the meaning of the mix parameter, effective edge loss ratio always remains at low level before mix rising up to 0.70. Manifestations of sequential partitioning method and random partitioning method are also easy to explain. Distribution of edges of LFR generated networks is random and uniform, regardless of storage order. As a result, the total edge loss ratio will



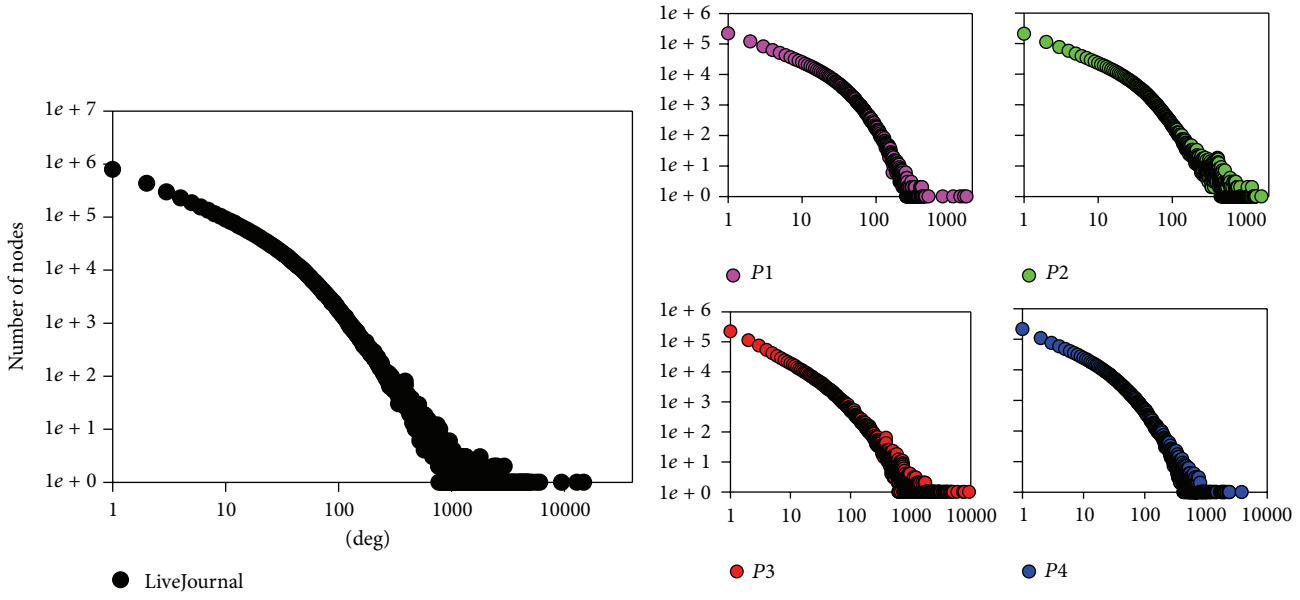
(a) $|P| = 2$

(b) $|P| = 4$

(c) $|P| = 2$, T, and E mean "total edge lost" and "effective edge lost," respectively

(d) $|P| = 4$

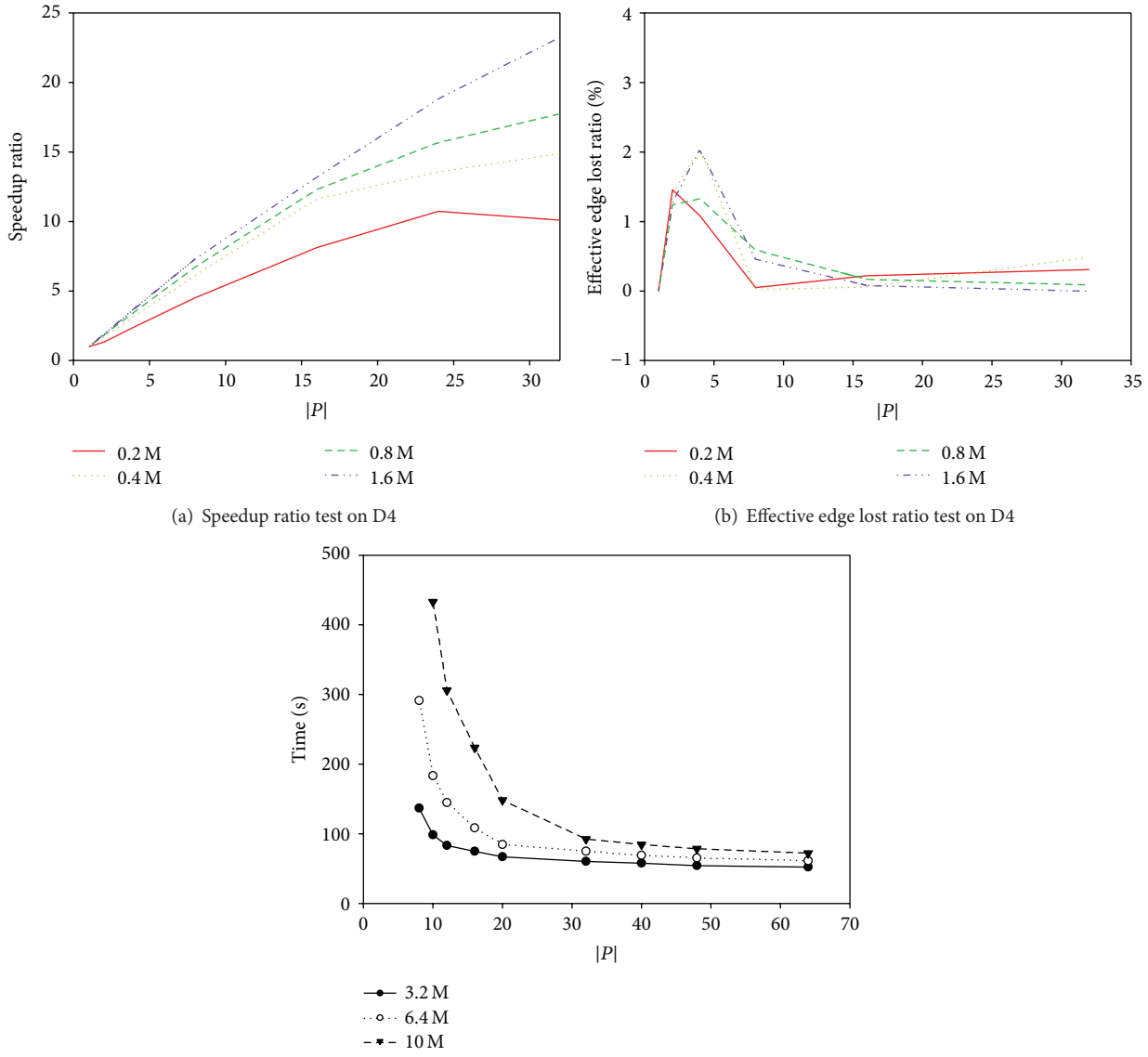
Figure 8: Performance and edge loss ratio of different partition methods on data set D3.



(a) Degree distribution of LiveJournal

(b) Degree distribution of subnetworks of LiveJournal

Figure 9: Degree distribution test on LiveJournal data set.



(a) Speedup ratio test on D4

(b) Effective edge lost ratio test on D4

(c) Running time test on D5

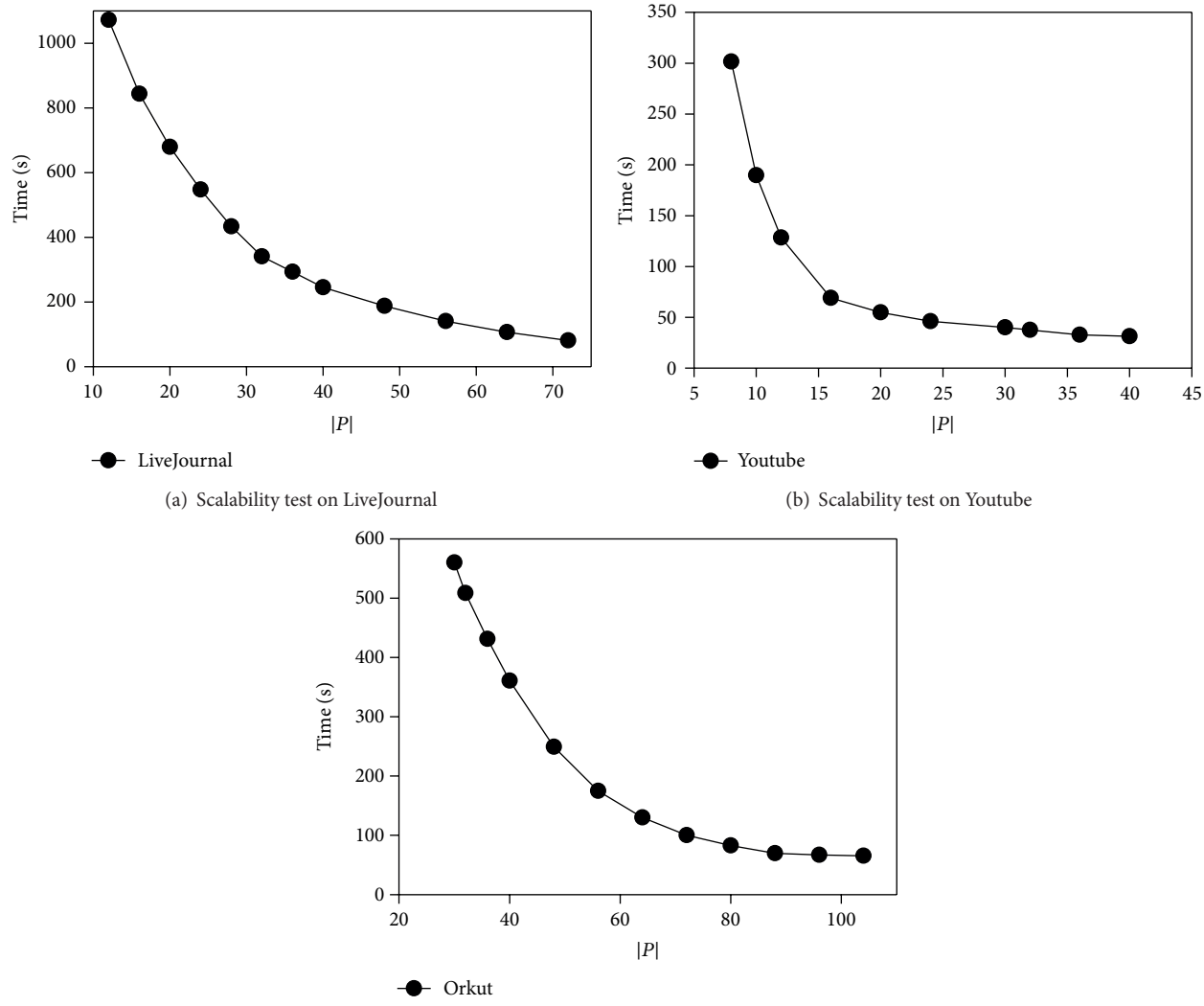
Figure 10: Scalability tests on data sets D4 and D5.

remain at about $(|P| - 1)/|P|$. Effective edge loss decreases linearly with the increase of mix for the same reason when total edge loss ratio increases linearly with the increase of mix of multilevel partitioning method in Figures 8(a) and 8(b).

In addition, we conduct a degree distribution test on a real network-LiveJournal to verify performance of the improved multilevel partitioning method. The network is divided into 4 partitions by the improved multilevel k-way partitioning method, and the degree distributions corresponding to the original network and the 4 subnetworks are shown in Figure 9. Comparative observation indicates that the subnetworks got from the improved multilevel k-way partitioning method are able to maintain the degree distribution characteristics of the original network.

4.3. Scalability and Performance Experiment. Our study aims to uncover community structures in big social networks and improve resource utilization as much as possible. Here, we unify the two problems together by means of MapReduce. With a small portion of expense of performance, we will achieve the goal. In this section, we will test the scalability and performance of the parallel community detection method, and data sets used are D4, D5, LiveJournal (http://snap.stanford.edu/data/com-LiveJournal.html), Youtube (http://snap.stanford.edu/data/as-skitter.html), and Orkut (http://snap.stanford.edu/data/com-Orkut.html).

For a certain network in Figures 10(a) and 10(b), when $|P|$ increases, the speedup ratio will increase but the acceleration will become slow, since MapReduce needs some time to



(a) Scalability test on LiveJournal

(b) Scalability test on Youtube

(c) Scalability test on Orkut

Figure 11: Scalability tests on real networks.

initiate before map tasks start to run and transmit data from map phase to reduce phase. For a certain $|P|$, as network size grows, the speedup ratio will become higher. For Figure 10(c), we just present the running time of parallel community detection method on MapReduce because the capacity of the servers cannot deal with such large networks on one server or with $|P| = 1$.

Finally, we apply the same process onto the real networks. Experiments on real networks shown in Figure 11 also confirm that our parallel community detection method has excellent scalability. From the results, we can conclude that, when $|P|$ increases, the subnetwork size assigned to each map task will be smaller, and the total edge lost ratio will increase, which will further reduce the subnetwork size. From Figures 10 and 11, we can get the following: in the case of constant data size, the running time and $|P|$ are linear approximation when $|P|$ is small. When partition number is small, the running time is affected by the number of partitions significantly. When the partition number $|P|$ reaches the "critical point" (Figure 11(c) Orkut, $|P| = 72$ and Figure 11(b) Youtube, $|P| = 20$), running time is less affected by the changes of partition number and shows "long tail effect" to some extent. The reason is that the cost of MapReduce is basically fixed. For a larger social network with the same number of map tasks, Mapreduces initial time accounts for a smaller proportion of the total running time. When partition number increases and the total running time decreases, the proportion of the initial time is not negligible. It makes our method exhibiting "long tail effect" in different data sets.

5. Conclusion

Community detection has become an important research topic in social networks. Traditional algorithms on community mining cannot effectively adapt to the current big social network scenarios [29, 30]. Infomap is excellent standalone

community detection method and, by means of multilevel k-way partitioning method enhanced by k-shell deposition, we are able to develop a parallel community discovery method on MapReduce framework. Related experiments verified the validity of the proposed work in this paper, and it may possess some reference meaning for social network analysis and social community mining with the big data techniques. Next, well try to use some overlapping partitioning methods to further improve the community detection accuracy.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to express their sincere gratitude to Zhang Yuchao from Beijing Institute of System Engineering for providing great assistance through the entire research process, Lancichinetti A. from Amaral Lab of Northwestern University for supporting their work unselfishly with implementation of some algorithms, and Chen Siming from University of Illinois at Chicago for his careful review, comments, and feedback on this paper. In addition, this research is supported by the National High-Tech R&D Program of China (nos. 2012AA012600, 2012AA01A401, and 2012AA01A402), the National Natural Science Foundation of China (no. 61202362), the State Key Development Program of Basic Research of China (no. 2013CB329601), and Project funded by the China Postdoctoral Science Foundation (no. 2013M542560).

References

[1] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002.

[2] C. Lee and P. Cunningham, "Community detection: effective evaluation on large social networks," *Journal of Complex Networks*, vol. 2, no. 1, pp. 19–37, 2013.

[3] D. J. Watts and S. H. Strogatz, "Collective dynamics of "small-world" networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[4] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, "Complex networks: structure and dynamics," *Physics Reports*, vol. 424, no. 4-5, pp. 175–308, 2006.

[5] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.

[6] M. E. J. Newman and E. A. Leicht, "Mixture models and exploratory analysis in networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, no. 23, pp. 9564–9569, 2007.

[7] G. R. Chen, X. F. Wang, and X. Li, *Introduction to Complex Networks: Models, Structures and Dynamics*, Higher Education Press, Beijing, China, 2012.

[8] S. Cooper, "The Largest Social Networks in the World Include Some Big Surprises [Business Insider]," January 2014, http://www.businessinsider.com/the-largest-social-networks-in-the-world-2013-12.

[9] T. N. Bui and C. Jones, "A heuristic for reducing fill-in in sparse matrix factorization," in *Proceedings of the 6th SIAM Conference on Parallel Processing for Scientific Computing (PPSC '93)*, pp. 445–452, 1993.

[10] K. Andreev and H. Räcke, "Balanced graph partitioning," *Theory of Computing Systems*, vol. 39, no. 6, pp. 929–939, 2006.

[11] A. Lancichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," *Physical Review E*, vol. 80, no. 5, Article ID 056117, 2009.

[12] M. Rosvall and C. T. Bergstrom, "Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems," *PLoS ONE*, vol. 6, no. 4, Article ID e18209, 2011.

[13] X. Ding, Advancement of operating system to manage critical resources in increasingly complex computer architecture. (Electronic Thesis or Dissertation), 2010, https://etd.ohiolink.edu/.

[14] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[15] S. Jin, A. Li, S. Yang, W. Lin, B. Deng, and S. Li, "A MapReduce and information compression based social community structure mining method," in *Proceedings of the IEEE 16th International Conference on Computational Science and Engineering (CSE '13)*, pp. 971–980, IEEE, Sydney, Australia, December 2013.

[16] M. Kitsak, L. K. Gallos, S. Havlin et al., "Identification of influential spreaders in complex networks," *Nature Physics*, vol. 6, no. 11, pp. 888–893, 2010.

[17] D. Zhao, L. Li, H. Peng, Q. Luo, and Y. Yang, "Multiple routes transmitted epidemics on multiplex networks," *Physics Letters A*, vol. 378, no. 10, pp. 770–776, 2014.

[18] D. Zhao, H. Peng, L. Li, Y. Yang, and S. Li, "An efficient patch dissemination strategy for mobile networks," *Mathematical Problems in Engineering*, vol. 2013, Article ID 896187, 13 pages, 2013.

[19] G. Karypis and V. Kumar, "Analysis of multilevel graph partitioning," in *Proceedings of the ACM/IEEE Conference on Supercomputing*, p. 29, ACM, 1995.

[20] L. Li, D. Alderson, J. C. Doyle, and W. Willinger, "Towards a theory of scale-free graphs: definition, properties, and implications," *Internet Mathematics*, vol. 2, no. 4, pp. 431–523, 2005.

[21] A. Abou-Rjeili and G. Karypis, "Multilevel algorithms for partitioning power-law graphs," in *Proceedings of the Parallel and Distributed Processing Symposium*, p. 10, 2006.

[22] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970.

[23] P. D. Grnwald, I. J. Myung, and M. A. Pitt, Eds., *Advances in Minimum Description Length: Theory and Applications*, MIT press, New York, NY, USA, 2005.

[24] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[25] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[26] G. K. Orman, V. Labatut, and H. Cherifi, "Towards realistic artificial benchmark for community detection algorithms evaluation," *International Journal of Web Based Communities*, vol. 9, no. 3, pp. 349–370, 2013.

[27] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, "Finding statistically significant communities in networks," *PLoS ONE*, vol. 6, no. 4, Article ID e18961, 2011.

[28] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, Article ID P10008, 2008.

[29] M. Li, "Fractal time series—a tutorial review," *Mathematical Problems in Engineering*, vol. 2010, Article ID 157264, 26 pages, 2010.

[30] M. Li, W. Chen, and L. Han, "Correlation matching method for the weak stationarity test of LRD traffic," *Telecommunication Systems*, vol. 43, no. 3-4, pp. 181–195, 2010.