Synergistic Partitioning in Multiple Large Scale Social Networks

Songchang Jin*, Jiawei Zhang†, Philip S. Yu†, Shuqiang Yang‡ and Aiping Li‡
*College of Computer, National University of Defense Technology
Changsha, Hunan 410073, China
Email: jinsongchang87@gmail.com
†Department of Computer Science, University of Illinois at Chicago
Chicago, Illinois 60607
Email: jzhan9@uic.edu, psyu@uic.edu
§College of Computer, National University of Defense Technology
Changsha, Hunan 410073, China
Email: sqyang9999@126.com, apli1974@gmail.com

***Abstract*—Social networks have been part of people's daily life and plenty of users have registered accounts in multiple social networks. Interconnection among multiple social networks adds a multiplier effect to social applications when fully used. With the sharp expansion of network size, traditional stand-alone algorithms can no longer support computing on large scale networks while alternatively, distributed and parallel computing become a solution to utilizing the data-intensive information hidden in multiple social networks. As such, synergistic partitioning, which takes the relationships among different networks into consideration and focuses on partitioning the same nodes of different networks into same partitions. With that, the partitions containing the same nodes can be assigned to the same server to improve the data locality and reduce communication overhead among servers, which are very important for distributed applications. To date, there have been limited studies on multiple large scale network partitioning due to three major challenges: 1) the need to consider relationships across multiple networks given the existence of intricate interactions, 2) the difficulty for stand-alone programs to utilize traditional partitioning methods, 3) the fact that to generate balanced partitions is NP-complete. In this paper, we propose a novel framework to partition multiple social networks synergistically. In particular, we apply a distributed multilevel *k*-way partitioning method to divide the first network into *k* partitions. Based on the given anchor nodes which exist in all the social networks and the partition results of the first network, using MapReduce, we then develop a modified distributed multilevel partitioning method to divide other networks. Extensive experiments on two real data sets demonstrate that our method can significantly outperform baseline independent-partitioning method in accuracy and scalability.**

I. Introduction

With the rapid development of computer science and technology, social network service has been thriving for several years, and has been playing increasingly important role in people's daily life. Social network allows individuals to create public profiles, to establish connections with a list of users with whom to share photo/video/music/blogging, and to send/receive messages across the connections within the system, etc [1, 2]. Plenty of users have signed up for several social networks focusing on different kinds of content, and are maintaining different relationships in different networks simultaneously.

In recent years, social network size drastically increases. A recent report from BI Intelligence indicates that, among the largest social networks, Facebook still has the largest user population at 1.16 billion monthly active users (MAUs), which is followed by YouTube with 1 billion, and China's social media network, Qzone, takes the third position with 712 million MAUs [3].

Co-processing refers to the information integration among multiple social networks. For example, current friend recommendation function derives information from a single network environment. However, through co-processing, information from multiple networks can be integrated and the efficiency and accuracy of two related friends can be greatly enhanced. The ever-expanding user base creates more challenges for co-processing the large amount of data stored in the multiple social networks environment. In our previous work, we have developed a novel community detection method, MCD (Mutual Community Detector) [4] which uses the heterogeneous information in multiple networks, but this method is limited to small networks. How to generalize MCD to large scale heterogeneous networks has been a burning question for us.

Users maintaining accounts in multiple networks are defined as *anchor users*. Making full use of the interactions of anchor users in different social networks will create multiplier effect for social applications. However, it is troublesome for large sized networks (notably the number of links) to be stored in main memory. Distributed and parallel computing on distributed file system will be more efficient than stand-alone computing on local disk.

As of now, there have been intensive research on partitioning within single network environment [5]. But problems about partitioning within multiple relevant network environment, especially within multiple large scale relevant networks, has less been studied. in this paper, we take into account data locality while partitioning multiple social networks so that the same partition on different social networks can be assigned to the same server. This will decrease the communication overhead among servers.

However, synergistic partitioning across multiple large scale social networks is very difficult for the following chal-

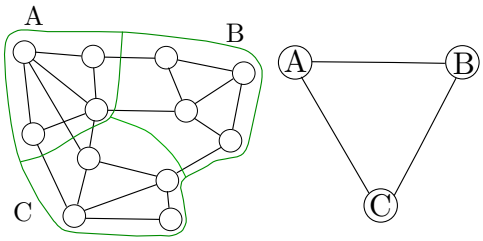


Fig. 1. A sample network that is divided into 3 balanced partitions and each partition has 4 nodes on the left. Corresponding abbreviated network is shown on the right, each node represents a partition and links mean that there exist connections between different partitions in the original network.

lenges: 1) social network, distinct from generic data, usually contains intricate interactions, and multiple heterogeneous networks mean that the relationships across multiple networks should be taken into consideration. 2) Network size implies it is difficult for stand-alone programs to apply traditional partitioning methods and it is a difficult task to parallelize the existing stand-alone network partitioning algorithms. 3) For distributed algorithms, load balance should be taken into consideration and how to generate balanced partitions is another challenge.

To address the challenges, we develop a network structure based distributed network partitioning framework. In it, 1) we identify the anchor nodes among the multiple networks. 2) We select a network as the *datum network*, then divide it into k balanced partitions and generate $\langle$*anchor node ID, partition ID*$\rangle$ pairs as our objective. 3) Based on the objective, we coarsen the other networks (called as *synergistic networks*) into smaller ones. 4) We divide the smallest networks into k balanced initial partitions, and try to assign same kinds of anchor nodes into the same initial partition as many as possible. Here, anchor nodes of same kind means that they are divided into same partition in the datum network. 5) we project the initial partitions back to the original networks.

The main contributions of this paper are as follows: 1) To the best of our knowledge, this is the first work to study synergistic partitioning problem in multiple large scale social networks. 2) We develop a new framework to tackle the problem. 3) We develop a distributed multilevel network coarsening algorithm. 4) We propose a modified label propagation algorithm to generate initial partitions.

The rest of the paper is organized as follows: Section II is about related work. In Section III, we formulate the problem. Section IV is about the framework and the detail information. We conduct experiments in Section V and conclude our work in Section VI.

II. Related Work

Mathematically, social networks can be represented as graphs. In the past years, researchers have developed many graph partition algorithms and methods for single networks. But graph partition is NP-complete [6], so, it is unlikely to always find out optimal partitions with a polynomial-time algorithm. Thus, almost all the practical algorithms on network partition are heuristics. Besides, to divide a network into k partitions, traditional methods usually generate a bisection, and take out recursive bisection method on it to get k partitions. In this section, we will briefly review some classical network partition algorithms.

Researchers have developed many sequential algorithms for network partition, which can be referred into local improvement method and global method. Local improvement method takes a partition of a network as input, and tries to decrease the cut size with some local search methods. To perform a local improvement method, initial partitions should be generated by some other methods beforehand, and the performance is strongly affected by the quality of initial partitions.

One of the ground-breaking network partition methods is Kernighan-Lin (KL) method [7], which aims to divide a network into 2 balanced partitions. In it, an initial balanced bisection should be given, and then iterations will be performed to try to find a sequence of node pair exchanges that can lead to a reduction in the cut size. Time expenditure of each iteration in KL method is $O(|n|^3)$. KL method is so important that plenty of local improvement methods are variations of it. For example, Dutt optimized the KL method and made it possible to complete KL process in $O(|E|max\{log|n|, d_{max}\})$ time, where d_{max} means the maximum of node degree [8]. Fiduccia-Mattheyses (FM) [9] is another KL inspired method. For FM method, it does not choose node pairs as in KL method but single nodes to improve edge-cut in each iteration which may result in unbalanced partitions, and each iteration can be finished in $O(|E|)$ time.

Unlike the KL and FM methods, Simulated annealing (SA) method [10] do not use greedy search but is based on statistical mechanics. It is a general purpose local search method, which tries to randomly select a neighbor node to estimate if it can improve the result of solution. But related experiments show that SA is not suitable for networks that are sparse or has some local structure [11].

Global method takes the whole graph and number of partitions - k as input, and generates a k-way partition. Most of these methods first bisect the graphs, and then apply bisection step recursively until k node subsets generated. Global methods can be divide into 2 categories according to whether the spatial location information of nodes are included in the networks, geometric methods and coordinate-free methods [5].

Recursive coordinate bisection (RCB) [12] is the prime example of geometric algorithms. The algorithmic thought of the RCB method is very simple, a coordinate axis should be chosen at the beginning, then it tries to find an orthogonal plane of the axis that can divide the node set V into 2 equal sized partitions. Inertial method [13] proposed by Farhat and Lesoinne elaborates the RCB method. In the inertial method, the axis with *minimum angular momentum* [14] of the node set is chosen instead of coordinate axis in the RCB method. Leland and Hendrickson then combined the KL method with inertial method together to improve the bisection generated by the inertial method.

In some networks, space location information of nodes is not embedded or can not get high-quality partitions even if given. In this situation, some algorithms only considering the combinational network structure have been proposed. The recursive graph bisection (RGB) method [15] begins by finding a pseudo peripheral node in the network, i.e., one of a pair of nodes that are approximately at the greatest network distance

from each other in the network. Finally, the nodes are sorted with respect to these distances and then the RGB method divides the nodes into two equal sized sets.

Pothen, Simon and Liou developed an eigenvector based method - recursive spectral bisection (RSB) method [16]. The RSB method sorts its nodes with respect to eigenvector coordinates and then divides the sorted nodes into two partitions. Based on RSB, Barnard and Simon first proposed a multilevel network graph partition method - multilevel-RSB to speed up the RSB method. The multilevel method consists of three phases: coarsening phase, partitioning phase and uncoarsening phase, which we will dilate in Section IV. The multilevel idea in the multilevel-RSB method has a profound influence on subsequent network partition methods, and lots of multilevel algorithms have been proposed by researchers, such as multilevel-KL methods [17, 18], multilevel k-way partition [19], multilevel graclus method [20], et al.

The KL method and SA method have been proved to be P-complete [21]. However, several parallel approximations of KL have been developed based on multi-processors (not multi-servers). The earliest KL based parallel network method which also is the earliest parallel method for network bisection is proposed by Gilbert and Zmijevski [22]. Savage and Wloka developed a Mob heuristic method [21]. Özturan et al proposed a new network bisection method based on iterative local node exchange between processors. Based on local improvement and a parallel multilevel evolutionary algorithm, Sanders and Schulz developed a balanced network partition method - KaFFPaE [23]. In 2013, Rahimian et al proposed a distributed balanced graph partition method - ja-be-ja based on local search and SA techniques [24]. Some other parallel network partition methods can be found in [5] and most of parallel methods are essentially parallelization of well-known sequential algorithms, which usually are developed on multi-processor but stand-alone servers.

III. Problem Formulation

The problem we are trying to address in this paper is synergistic partitioning in multiple large scale social networks. In this section, we formulate the problem and describe the assumptions used duing the work.

A. Terms and Definitions

Definition 1: (*Social network*) A social network can be mathematically described as $G = (V, E)$, where V and E represent the node set and the edge set of G, respectively. An integer $n = |V|$ is used to record the size of V and $m = |E|$ represents the number of edges. $v(i)$ represents node i and corresponding degree is $d(v(i))$. $e_{i,j} \in E$ means the link between $v(i)$ and $v(j)$.

Definition 2: (Anchor node and Non-anchor node) Users involved in multiple social networks simultaneously are called as anchor users, and named as anchor nodes in network set $GS = \{G^1, G^2, \ldots, G^t\}$. All the anchor nodes constitute the anchor node set A, and $A = \cap V^i, 1 \leq i \leq t$. The left nodes are called non-anchor nodes, constituting the non-anchor node set - NA. So, $A \cap NA = \emptyset$ and $A \cup NA = \cup V^i, 1 \leq i \leq t$.

Definition 3: (Input space) Input space InS means the network set and the anchor node set in this paper, so $InS = (GS, A)$. From Ins, we are able to get all the structural information about the multiple social networks.

Definition 4: (Balanced network partition) For an undirected network $G = (V, E)$, balanced (or uniform) network partition refers to the problem of partitioning V into equal sized components and the number of links across boundaries of different components is minimized. All the nodes and links within a component constitute a *partition*. If the number of components is given as k, then the problem is called as balanced k-way network partitioning problem.

B. Problem Formulation

To take full advantage of users' relationships in multiple social networks, cooperative analysis is required. However, in view of the facts that even a single social network is so large that the network data can not be stored in the main memory, but may need to placed onto local disk or remote distributed filesystem [25]. In the distributed data placement case, for the existence of anchor nodes connecting multiple social networks together closely, network partitioning methods have to take the relationships among the social networks into account.

Links in social networks can reflect people's interactions in the real world to a great extent. So, we can assume that anchor users' interpersonal relationships in different social networks should be similar. If we take these anchor nodes as samples, and assuming these samples are evenly distributed, it would be reasonable to believe that there are a lot of similar structures revolving around the samples hidden in the networks. Furthermore, if we expand the similar structures to the whole networks, we will be able to get potential similar partition pairs. Finally, we can align the partitions of all the networks and put similar partitions onto same servers. As a result, data locality will increase and communication volume between servers will reduce remarkably, which are very important in distributed applications. For example, in the ideal case of cooperative partitioning dividing two social networks into k partitions, applications only need searching local subnetworks and do not need searching the subnetworks stored in other servers, and the communication volume may decrease from $O(m^2)$ to $O(m)$. Fig.2 shows a simple diagram of result comparison between synergistic partitioning method and independent partitioning method.

Mathematically, synersitic partition problem can be described as: For an input space $InS = (GS, A)$, try to find a partition set $PS = \{P^1, P^2, \ldots, P^t\}$, where the amount of anchor nodes divided into same partitions in different networks - $SumAN$ is maximized and the edge-cut size EC is minimized. Let $pidx(G, v(i))$ and $P(G, j)$ be the partition ID of $v(i)$ and partition j in network G, respectively. $w_{i,j}$ represents the weight of link $e_{i,j}$. In the networks studied in this paper, $w_{i,j} = 1$ if $e_{i,j} \in E$, and else $w_{i,j} = 0$.

$$count(v(i)) = \begin{cases} 1 & \text{if all } pid(G^j, v(i))\text{s are the same, where } 1 \leq j \leq t. \\ 0 & \text{others.} \end{cases} \quad (1)$$

$$SumAN = \sum_{v(i) \in A} count(v(i)) \quad (2)$$

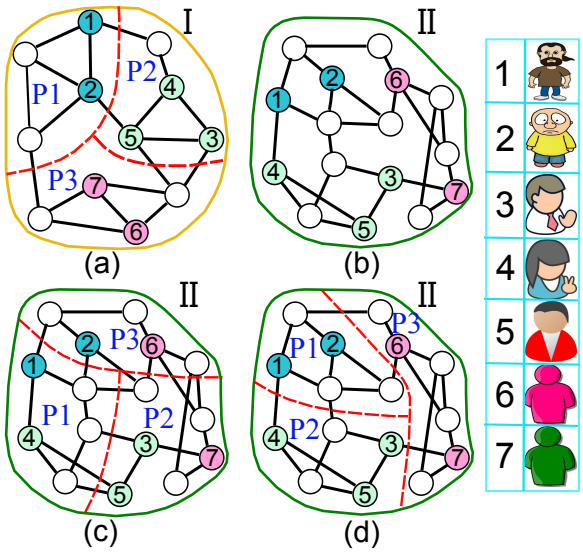


Fig. 2. Diagrammatic sketch of synergistic partitioning method and independent partitioning method on two social networks - network I (datum network) and network II (synergistic network). Anchor users and corresponding node IDs are shown on the right. (d) shows the result of synergistic partitioning method taking the partition result of anchor nodes in datum network into consideration. (c) is the result of an independent partitioning method. Compared the partition results of network II shown in (c) and (d) with the result of network I in (a), similarity of partitions between (a) and (d) is higher than that between (a) and (c).

$$EC(G) = \sum_{v(i),v(j)} w_{i,j} \tag{3}$$

Where $v(i) \in P(G,u)$, $v(j) \in P(G,w)$ and $u \neq w$.

$$EC = \sum_{i=1}^{t} EC(G^i) \tag{4}$$

IV. Proposed Framework

In this section, we describe the heuristic framework for synergistic partitioning among multiple large scale social networks, and we call the framework - SPMN.

A. The Basic Idea

Data processing in SPMN can be roughly divided into two stages: datum generation stage and network alignment stage.

When got the anchor node set A, the framework will apply a distributed multilevel k-way partitioning method onto the datum network to generate k balanced partitions. During this process, the anchor nodes are ignored and all the nodes are given the same treatment. We call this process datum generation stage. When finished, partition result of anchor nodes will be generated, we store them in a set - $Map\langle anidx, pidx\rangle$, where $anidx$ is anchor node ID and $pidx$ represents the partition ID the anchor node belongs to.

After the datum generation stage, synergistic networks will be partitioned into k partitions according to the $Map\langle anidx, pidx\rangle$ to make the synergistic networks to align to the datum network, and during this process $max\{SumAN\}$ and $min\{EC\}$ are the objectives. We call this process network alignment stage.

B. Distributed Multilevel k-way partitioning

Algorithms guaranteed to find out near-optimal partitions in a single network have been studied for a long period, which have been discussed in Section II. But most of the methods are stand-alone, and performance is limited by the server's capacity. Inspired by the multilevel k-way partitioning (MKP) method proposed by Karypis and Kumar [19, 26] and based on our previous work [25], we try to use MapReduce [27] to speedup the MKP method. As the same with other multilevel methods, MapReduce based MKP also includes three phases: coarsening, init partitioning and un-coarsening. Fig.3 gives a vivid description about MKP method.

Coarsening phase is a multilevel process and a sequence of smaller approximate networks $G_i = (V_i, E_i)$ are constructed from the original network $G_0 = (V, E)$, where $|V_i| < |V_{i-1}|$, $i = 1, 2, ..., s$. To construct coarser networks, node combination and edge collapsing should be performed. The task can be formally defined in terms of *matchings* [28]. A matching is a set of node pairs $M = List\langle i, j\rangle, i \neq j$ and $e_{i,j} \in E$, and in which each node can only appear for no more than once. For a network G_i with a matching M_i, if $\langle j, k\rangle \in M_i$ then $v(j)$ and $v(k)$ will form a new node $v(q) \in V_{i+1}$. The weight of $v(q)$ equals to the sum of weight $v(j)$ and $v(k)$, besides, all the links connected to $v(j)$ or $v(k)$ in G_i will be connected to $v(q)$ in G_{i+1}. The total weight of nodes will remain unchanged during the coarsening phase but the total weight of edges and number of nodes will be reduced. Define $W(T)$ to be the sum of edge weight in T and $N(T)$ to be the number of components in T. It will be that:

$$W(E_{i+1}) = W(E_i) - W(M_i) \tag{5}$$

$$N(V_{i+1}) = N(V_i) - N(M_i) \tag{6}$$

Analysis in [29] shows that for the same coarser network, smaller edge-weight corresponds to smaller edge-cut. With the help of MapReduce framework, we use a local search method to implement an edge-weight based matching (EWM) scheme to collect larger edge weight during the coarsening phase. For the convenience of MapReduce, we design a new network representation format: each line contains essential information about a node and all its neighbors (NN), such as node ID, node weight (VW), edge weight (w), et al. The whole network data are distributed in distributed filesystem, such as HDFS [30], and each data block only contains part of node set and corresponding connection information. Function $map()$ takes a data block as input and search locally to find node pairs to match according to the edge weight, $reduce()$ is in charge of node combination, renaming and sorting. With the new node IDs and matching, a simple MapReduce job will be able to update the edge information and write the coarser network back onto HDFS. The complexity of EWM is $O(|E|)$ in each iteration and pseudo code about EWM is shown as follow.

Input:
A network, G_h;
Maximum weight of a node, $maxVW = n/k$;
Output:
A coarser network, G_{h+1};
1: $map()$ function:
2: **for all** node i in current data block **do**

```
    if match[i] == −1 then
      maxIdx = −1;
      sortByEdgeWeight(NN(i));
      for all v_j ∈ NN(v_i) do
        if match[j] == −1& VW(i) + VW(j) < maxVW then
          maxIdx = j;
        end if
        match[i] = maxIdx;
        match[maxIdx] = i;
      end for
    end if
  end for
  reduce() function:
  new newNodeID[n + 1];
  new newVW[n + 1];
  set idx = 1;
  for all i ∈ [1, n] do
    if i < match[i] then
      set newNodeID[match[i]] = idx;
      set newNodeID[i] = idx;
      set newVW[i] = newVW[match[i]] = VW(i) + VW(match[i]);
      idx + +;
    end if
  end for
```

After several iterations, a coarsest undirected weighted network G_s consisting of only hundreds of nodes will be generated. For the network size of G_s, stand-alone algorithms with high computing complexity will be acceptable for init partitioning. Meanwhile, the weights of nodes and edges of coarser networks are setted to reflect the weights of the finer network during the coarsening phase, so G_s contains sufficient information to intelligently satisfy the balanced partition and the minimum edge-cut requirements. Plenty of traditional bisection methods are quite qualified for the task. In this paper, we adopt the KL method with an $O(|E|^3)$ computing complexity to divide G_s into two partitions and then take recursive invocations of KL method on the partitions to generate balanced k partitions.

Un-coarsening phase is inverse processing of coarsening phase. With the initial partitions and the matchings of the coarsening phase, it is easy to run the un-coarsening process on the MapReduce cluster.

C. Synergistic Partitioning Process

In this section we talk about the synergistic partitioning process on the synergistic networks with the knowledge of partition results of anchor nodes from datum network. The synergistic partitioning is also a MKP process but quit different from general MKP methods.

In the coarsening phase, anchor nodes are endowed with higher priority than non-anchor nodes. When choosing nodes to pair, we assume that anchor nodes and non-anchor nodes have different tendencies. Let G^d be the datum network and $G^s = \{G^{s1}, G^{s2}, \ldots, G^{s(t-1)}\}$ be the synergistic network set.

For an anchor node $v(i)$, it would prefer to combine with another anchor node $v(j)$ which has the same partition ID in the datum network, i.e., $pidx(G^d, v(i)) = pidx(G^d, v(j))$ where $v(i) \in A, v(j) \in A$ and $i \neq j$. Second, if there is no appropriate anchor node, it would try to find a non-anchor node to pair. When planing to find a non-anchor node to pair, the anchor node, assuming to be $v(i)$, would like to find a correct direction, and it would prefer to match with the non-anchor node $v(j)$, which has lots of anchor nodes as neighbors with the same $pidx$ with $v(i)$. When combined together, the new node will be given the same $pidx$ as the anchor node. To improve the accuracy of synergistic partitioning among multiple social networks, an anchor node will never try to combine with another anchor node with different $pidx$.

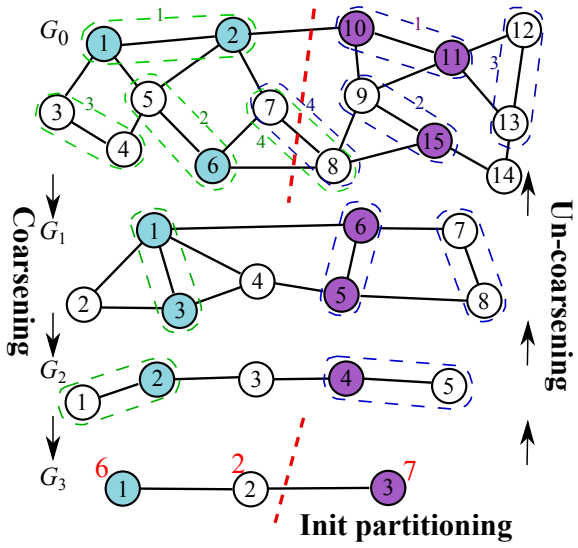


Fig. 3. Diagrammatic sketch of synergistic partitioning process dividing a synergistic network into two partitions according to the partition results of anchor nodes in datum network. In coarsening phase, the networks are stored in two servers, $V_1^i = \{v^i(j) | j \leq |V^i|/2\}$ are stored on a sever and the others are on the other server. Anchor nodes are with colors, and different colors represent different partitions. Node pairs encircled by dotted chains represent the matchings. Numbers on chains mean the order of pairing.

For a non-anchor node, it would prefer to make a pair with an anchor node neighbor which belongs to the dominant partition in the non-anchor node's neighbors. Here, dominant partition in a node's neighbors means the number of anchor nodes with this partition ID is largest. Next, a non-anchor node would choose a general non-anchor node to pair. At last, a non-anchor node would not like to combine with an anchor node being part of the partitions which are in subordinate status. After combined together, the new node will be given the same $pidx$ as the anchor node. To ensure the balance among the partitions, about 1/3 of the nodes in the coarsest network are unlabeled.

Fig.3 shows a diagrammatic sketch of synergistic partitioning process. Take G_0 for example, nodes $v_1 \sim v_7$ and the corresponding links information are stored on the same server and the other nodes are stored on another server. Nodes in pairs $\langle v_1, v_2 \rangle$ and $\langle v_{10}, v_{11} \rangle$ are all with the same $pidx$, so they should be tackled first. v_6 and v_{15} choose a correct direction to make a pair. Then, v_3 can not pair with v_1, so it chooses v_4 to combine. Finally, after searching locally, v_7 can not find a local neighbor to pair, but has to make a pair with its remote neighbor v_8. The situation of v_8 is the same as v_7.

As discussed in *part A* of Section IV, $max\{SumAN\}$, $min\{EC\}$ and trying to balance the size of partitions are the objectives in synergistic partitioning process. However, when put together, it is impossible to achieve them simultaneously. So, we try to make a compromise among them and develop a heuristic method to tackle the problems. First, according to the conclusion - smaller edge-weight corresponds to smaller edge-cut and the pairing tendencies, we propose a modified EWM (MEWM) method to find a matching in the coarsening phase, of which the edge-weight is as large as possible. At the end of the coarsening phase, there is no impurity in any node, meaning that each node contains no more than one type of anchor nodes. Besides, a "purity" vector attribute and a "pidx" attribute are added to each node to represent the percentage of each kind of anchor nodes swallowed up by it and the "pidx" of the new node, respectively. Then, during the init partitioning phase, we treat the anchor nodes as labeled nodes and use a modified label propagation algorithm to deal with the non-anchor nodes in the coarsest network. At the end of the init partitioning phase, we will be able to generate balanced k partitions and to maximize the number of same kind of anchor nodes being divided into same partitions. Finally, we project the coarsest network back to the original network, which is the same as traditional MKP process. The pseudo code of coarsening phase in synergistic partitioning process is as follow.

Input:
A network, G_h;
A map consists of anchor nodes with $pidx$s, $Map\langle anidx, pidx\rangle$;
Maximum weight for a node, $maxVW = n/k$;
Output:
A coarser network, G_{h+1};

$map()$ function:
for all v_i in current data block **do**
  if $match[i] == -1$ **then**
    set $flag = false$;
    $sortByEdgeWeight(NN(v_i))$;
    if $v_i \in Map\langle anidx, pidx\rangle$ **then**
      for all $v_j \in NN(v_i)$ & $match[j] == -1$ **do**
        if $v_j \in Map\langle anidx, pidx\rangle$ & $Map.get(v_i) == Map.get(v_j)$ &$VW(i) + VW(j) < maxVW$ **then**
          $match[i] = j$; $match[j] = i$;
          $flag = true$; $break$;
        end if
      end for
      if $flag == false$, no suitable anchor node **then**
        for all $v_j \in NN(v_i)$& $match[j] == -1$ &$VW(v_i) + VW(v_j) < maxNW$ **do**
          $indirectNeighbor[] = NN(v_j)$;
          $sortByEdgeWeight(indirectNeighbor)$;
          for all $v_k \in indirectNeighbor$ **do**
            if $v_k \in Map\langle anidx, pidx\rangle$ & $Map.get(v_i) == Map.get(v_k)$ **then**
              $match[i] = j$; $match[j] = i$;
              $flag = true$; $break$;
            end if
          end for
          if $flag == true$ **then**
            $break$;
          end if
        end for
      end if
    else
      $sortByEdgeWeight(NN(v_i))$;
      for all $v_j \in NN(v_i)$ & $v_j \notin Map\langle anidx, pidx\rangle$ & $VW(i) + VW(j) < maxVW$ & $match[j] == -1$ **do**
        $match[i] = j$; $match[j] = i$; $break$;
      end for
    end if
  end if
end for
$reduce()$ function:
new $newNodeID[n+1]$;
new $newVW[n+1]$;
set $idx = 1$;
for all i in $newNodeID[]$ **do**
  if $i < match[i]$ **then**
    $newNodeID[match[i]] = idx$;
    $newNodeID[i] = idx$;
    $newVW[i] = VW(i) + VW(match[i])$;
    $newVW[match[i]] = VW(i) + VW(match[i])$;
    $idx++$;
  end if
end for
new $newPurity[idx+1]$;
new $newPidx[idx+1]$;
for all $i \in [1, idx]$ **do**
  $newPurity[i] = \frac{purity[i]*VW(i)+purity[j]*VW(j)}{VW(i)+VW(j)}$;
  $newPidx[i] = max\{pidx[i], pidx[match[i]]\}$;
end for

The label propagation algorithm (LPA) [31] is a graph-based semi-supervised learning algorithm. It can be used to predict the information of unlabeled nodes by a few of labeled nodes, and has many good natures such as simple, effective, speedy, favorable scalability and reliability. Inspired by the idea of local degree center based label propagation algorithm (LDC-LPA)[32], we develop a modified LPA method to deal with the weighted nodes and weighted links in the coarsest network. In each iteration in LPA, normalized probabilistic transfer matrix $\overline{T}_{n\times n}$ should be given, then we can update the probabilities matrix of each node $Y_{n\times k}$ by ($y_{i,j}$ means the probability of v_i being labeled as j):

$$Y \leftarrow \overline{T}Y \tag{7}$$

In consideration of weighted nodes and weighted links, we have to modify the transfer matrix $\overline{T}$. In the coarsest network, each node is a node set, but only labeled nodes are able to propagate their labels. Besides, each link contains plenty of links, but it maybe that parts of the links are from the labeled nodes. Take Fig.4 as example, v_c has four neighbors with different VWs and $purity$s, and the links between v_c and its neighbors are with different EWs. We assume the percentage of links from the labeled nodes of a node, e.g. v_a, to the unlabeled node, e.g. v_c, equals to the purity of the node. Then, we can define $T_{i,j}$ as:

$$\overline{T}_{i,j} = P(j \to i) = \frac{EW_{i,j} * purity_i}{\sum_{v_i \in neighbor(j)} EW_{i,j} * purity_i} \tag{8}$$

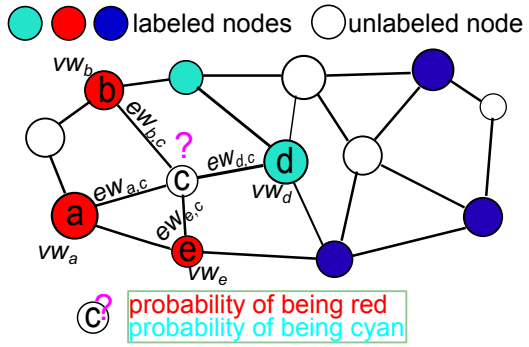


Fig. 4. The model of label propagation. There are three kinds of labels in the network. For an unlabeled node, such as v_c, we can consider it with its local environment, and assign potential labels to it with different probabilities according to the closenesses of links, such as $e_{a,c}, e_{b,c}, e_{d,c}, e_{e,c}$. Finally, an unlabeled node will be given the label with the largest probability.

TABLE I. DATA SETS DESCRIPTION

Foursquare			Twitter			Anchor Node
n	m	$\overline{d}$	n	m	$\overline{d}$	n
1,064,011	111,812,450	105	1,180,762	88,381,472	75	413,182

To ensure the convergence of the modified LPA, we use the initialization method of LDC-LPA to initialize the probability vector for each labeled node and randomly assign probabilities to the unlabeled nodes. At the end of the iterations, a stable probability distribution matrix Y will be generated. We can sort Y by the first label and collect the top n/k nodes together to form the first partition. Then delete the collected nodes from Y and repeat the process for the other labels. Finally, we will get k approximately equal sized partitions.

V. EXPERIMENT AND ANALYSIS

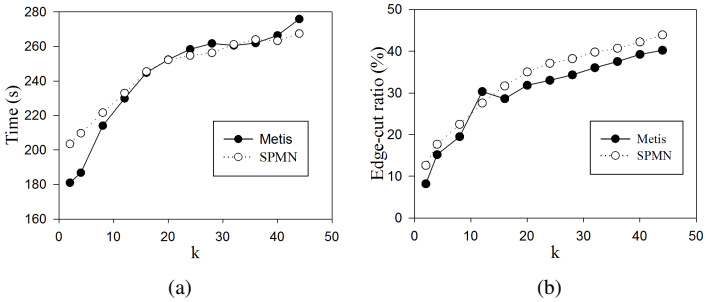
In this section, we conduct extensive experiments to evaluate the work proposed in this paper.

A. Experimental Environment and Data Set

In this paper, all experiments are running on a Hadoop-1.1.1 cluster of Antivision Software Ltd., which consists of 20 PowerEdge R320 servers (Intel Xeon CPU E5-1410 @2.80GHz, memory 8GB) with 64-bit NeoKylin Linux OS connected by a Cisco 3750G-48TS-S switch.

Data sets used are parts of two real social networks: Foursquare and Twitter, which are crawled from May 7th to June 9th. In Foursquare, we crawl 1,064,011 users, among them about 413,182 have their accounts in Tiwtter as well. Numbers of followers and users of these 413,182 twitter users follow in Twitter are 1,030,855,018 and 187,295,465 respectively. To compare with the baseline method, we sample a smaller sized subnetwork of Twitter, which has about the same size with Foursquare, to perform experiments. The details about data sets can be found in Tab.1.

The baseline method used to compare with the SPMN framework is an independent network partitioning method - Metis, which is the state-of-the-art method for single network partitioning by now. With Metis, we first partition all the networks independently. Then for each partition $P_i^d, 1 \leq i \leq k$ in G^d, we search in each synergistic network $G^{sj} \in G^s, 1 \leq j \leq t-1$, the partition $P_j^{sl} \subset V^{sl}$ containing the largest number of same kind of anchor node with P_i^d will be chosen to align with P_i^d.



(a) (b)

Fig. 5. Comparation of SPMN and Metis on Foursquare network which is chosen as datum network.

To evaluate the accuracy and the balance performance of the framework SPMN, we compute the NMI (Normalized Mutual Information) [33] of anchor nodes in the datum network and synergistic networks. To measure the edge-cut, we edge-cut ratio to represent the percetage of edges cut off by the partition boundaries.

B. Experiments and Results

In this section, first, we simply test the performance of SPMN on datum network (Foursquare network) and the result is shown in Fig.5. From Fig.5(a), we can see that SPMN consumes more time than Metis when the partition number k is small. But when k increases, growth of running time slows down. When k climbs to 32, Metis becomes slower than SPMN. By and large, the increments in both methods are very small. The reason is as follows. The first phase consumes the most time in MKP methods. For SPMN, time consumptions in the first and last phases have nothing to do with k and would be constants. Besides, they are running on MapReduce, so time consumption should be less than Metis. However, the coarsening phase depends two jobs (node pairing and edge renaming) and several iterations, and job intialization in MapReduce would take up great part. The init partitioning phase takes recursive iterations to generate intial partitions and the number of iterations is linear to k which is about the same with Metis. For Metis, only the first phase doesn't concern partition number. If network size increases steadily, proportion of job initialization will decrease and the total time consumption in SPMN will be less than Metis. From Fig.5(b), we can see that the edge-cut size in SPMN is larger than in Metis, that is because Metis adds a refine process during the uncoarsening phase to further optimize the partitions.

Second, we verify the validity and practicability of SPMN on synergistic network (Twitter network). To increase the number of data sets and guarantee the connectedness in all networks, we deliberately and randomly ignore parts of anchor nodes and construct another three data sets which are only different in the number of anchor nodes. D1 has 413,182 anchor nodes, D2 has 328,014 anchor nodes, D3 has 203,491 anchor nodes and D4 has 109,842 anchor nodes. To test the scalability performance of SPMN, we enlarge the synergistic network by

TABLE II. SYNERGISTIC NETWORKS FOR SCALABILITY TEST ($\times 10^6$)

	D1	D5	D6	D7	D8	D9
n	≈ 1.18	1.5	1.8	2.1	2.4	2.7
m	≈ 88.4	≈ 127.4	≈ 152.8	≈ 178.0	≈ 203.8	≈ 230.2

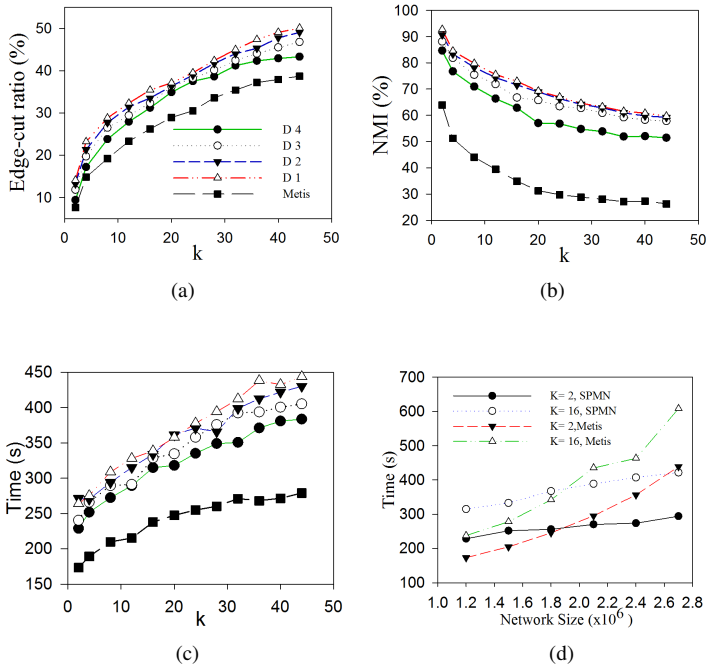


Fig. 6. Performance of SPMN on Synergistic Network. (a) Edge-cut test; (b) NMI test on the partitions of anchor nodes in datum network and synergistic network; (c) Running time test; (d) Scalability test on synergistic networks.

adding randomly selected non-anchor node set based on D1, and all the non-anchor nodes are from our original Twitter data set which totally contains 1,030,855,018 nodes. The data sets are described in Tab.2. With the partition results of anchor nodes in datum network, we conduct experiments on D1$\sim$D9 and results are shown in Fig.6.

First, we compare SPMN with Metis. From Fig.6(a) and Fig.6(c), we can see that Metis cuts off less links and runs faster than SPMN during the network partitioning process. As discribed in previous section, Metis partitions the datum network and synergistic network independently, which means that it treats all the nodes as non-anchor nodes. SPMN has to consider the distribution of anchor nodes during the partitioning process. Moreover, Metis' extra refining process during the uncoarsening phase makes it more effective in controlling the edge-cut size and SPMN's job intializations account for large proportion of the total time consumption.

From the results of NMI and scalability tests shown in Fig.6(b) and Fig.6(d), we can see that SPMN is more effective than Metis. Higher NMI means that more anchor nodes are assigned to correct partitions and the mapping structure between partitions of different neteworks is more distinct. In distributed applications, partitions mapped together will be assigned to the same servers. In this situation, data locality will be improved and communication traffic among different servers will be reduced, both of which are very important for large scale data analysis. In traditional MKP methods, coarsening phase comsumes the largest part of total time consumption. Metis is a stand-alone method and can not effectively accelerate the coarsening phase. However, SPMN, by employing the MapReduce computing model, is able to finish the process with relatively less time consumption. So, in terms of scalability, SPMN performs more excellent than Metis.

Next, we focus on the performance of SPMN itself. In the Fig.6(a), we can see that for a certain synergisitc network, the edge-cut size increases but the speed slows down as partition number grows. For the same partition number k, the more the anchor nodes are, the more the links would be cut off. Besides, for the same k, no significant difference in edge-cut ratio is observed among different synergistic networks. For NMI in Fig.6(b), the value decreases as k increases. When k increases, the probability of assigning an anchor node to incorrect partitions will be higher, then the NMI value will decrease. When k keeps increasing, speed of NMI slows, that is because the other partitions will share responsibility for the incorrect partition results.

For the running time test, compared with EWM scheme, taking the anchor nodes distribution into account in MEWM will add extra workload to coarsening phase and will consume more time to finish the coarsening task. Moreover, for a certain synergistic network, higher k means that more time is required for the LPA to generate intial partitions. For the scalability test shown in Fig.6(d), we can see that magniitude of increase in time consumption is negligible as k increases. This is attributed to two reasons: 1) there are sufficient servers to process workload in the coarsening phase. 2) the size of the coarsest networks generated in the coarsening phase are similar, meaning time consumtion in initial partitionning phase of those networks are close as well. Plus, the uncoarsening phase does not distinguish anchor nodes and non-anchor nodes.

VI. CONCLUSION

The synergistic partitioning we propose, combined with distributed computing, has proven to be an effective approach toward coprosessing interconnected information in multiple social networks environement. We have seen siginificant improvenment in NMI and scalability when comparing our approach with the state-of-the-art independent partitioning method. Currenly, synergistic partitioning still has the pitfall of enlargening the size of edge-cut. Adding a refinement process to the uncoarsening phase could be a potential solution to this problem. Application using synergistic patitioning is able to partition multiple social networks considering the relaitonship among them, which is not taken into account by traditional methods. In this way, the coprocessing in large scale social networks can be greatly advanced. For example, the accuracy and performance of production recommendation will be improved.

ACKNOWLEDGMENT

This research is supported by the National High-Tech R&D Program of China (No.2012AA012600, 2012AA01A401, 2012AA01A402), National Natural Science Foundation of China (No. 61202362), State Key Development Program of Basic Research of China (No. 2013CB329601) and Project funded by China Postdoctoral Science Foundation (No. 2013M542560).

REFERENCES

[1] A. Marion and O. Omotayo, "Development of a social networking site with a networked library and conference chat," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 2, no. 8, pp. 396–401, Aug. 2011.

[2] D. Boyd and N. Ellison, "Social network sites: Definition, history, and scholarship," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, pp. 210–230, 2007.

[3] S. Cooper, "The planets 24 largest social media sites, and where their next wave of growth will come from," Business Insider, NY, Tech. Rep., Nov. 2013. [Online]. Available: http://www.businessinsider.com/a-global-social-media-census-2013-10

[4] J. Zhang and P. Yu, "Mcd: Mutual clustering across multiple heterogeneous networks," *KDD2014*, submitted for publication.

[5] P. Fjällström, "Algorithms for graph partitioning: A survey," *Linköping electronic articles in computer and information science*, vol. 3, no. 10, Sep. 1998.

[6] T. Feder, P. Hell, S. Klein, and R. Motwani, "Complexity of graph partition problems," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*. ACM, 1999, pp. 464–472.

[7] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell system technical journal*, vol. 49, no. 2, pp. 291–307, 1970.

[8] S. Dutt, "New faster kernighan-lin-type graph-partitioning algorithms," in *Computer-Aided Design, 1993. ICCAD-93. Digest of Technical Papers., 1993 IEEE/ACM International Conference on*. IEEE, 1993, pp. 370–377.

[9] C. Fiduccia and R. Mattheyses, "A linear-time heuristic for improving network partitions," in *Design Automation, 1982. 19th Conference on*. IEEE, 1982, pp. 175–181.

[10] S. Kirkpatrick, C. Gelatt, M. Vecchi *et al.*, "Optimization by simmulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[11] R. Williams, "Performance of dynamic load balancing algorithms for unstructured mesh calculations," *Concurrency: Practice and experience*, vol. 3, no. 5, pp. 457–481, 1991.

[12] M. Berger and S. Bokhari, "A partitioning strategy for nonuniform problems on multiprocessors," *Computers, IEEE Transactions on*, vol. 100, no. 5, pp. 570–580, 1987.

[13] C. Farhat and M. Lesoinne, "Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics," *International Journal for Numerical Methods in Engineering*, vol. 36, no. 5, pp. 745–764, 1993.

[14] C. Truesdell, *A first course in rational continuum mechanics*. Academic Press, 1992, vol. 1.

[15] H. Simon, "Partitioning of unstructured problems for parallel processing," *Computing Systems in Engineering*, vol. 2, no. 2, pp. 135–148, 1991.

[16] A. Pothen, H. Simon, and K. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 3, pp. 430–452, 1990.

[17] B. Hendrickson and B. Leland, "A multi-level algorithm for partitioning graphs," *SC*, vol. 95, p. 28, 1995.

[18] A. Gupta, "Fast and effective algorithms for graph partitioning and sparse-matrix ordering," *IBM Journal of Research and Development*, vol. 41, no. 1.2, pp. 171–183, 1997.

[19] G. Karypis and V. Kumar, "Multilevel k-way partitioning scheme for irregular graphs," *Journal of Parallel and Distributed computing*, vol. 48, no. 1, pp. 96–129, 1998.

[20] I. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1944–1957, 2007.

[21] J. Savage and M. Wloka, "Parallelism in graph-partitioning," *Journal of Parallel and Distributed Computing*, vol. 13, no. 3, pp. 257–272, 1991.

[22] J. Gilbert and E. Zmijewski, "A parallel graph partitioning algorithm for a message-passing multiprocessor," *International Journal of Parallel Programming*, vol. 16, no. 6, pp. 427–449, 1987.

[23] P. Sanders and C. Schulz, "Think locally, act globally: Perfectly balanced graph partitioning," *arXiv preprint arXiv:1210.0477*, 2012.

[24] F. Rahimian, A. Payberah, and S. G. et al, "Ja-be-ja: A distributed algorithm for balanced graph partitioning," in *Self-Adaptive and Self-Organizing Systems (SASO), 2013 IEEE 7th International Conference on*. IEEE, 2013, pp. 51–60.

[25] C. Aggarwal, Y. Xie, and P. Yu, "Gconnect: a connectivity index for massive disk-resident graphs," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 862–873, 2009.

[26] G. Karypis and V. Kumar, "Parallel multilevel series k-way partitioning scheme for irregular graphs," *Siam Review*, vol. 41, no. 2, pp. 278–300, 1999.

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

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

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

[30] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–10.

[31] X. Zhu, Z. Ghahramani, J. Lafferty *et al.*, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, vol. 3, 2003, pp. 912–919.

[32] Q. Chen, T. Wu, and M. Fang, "Detecting local community structures in complex networks based on local degree central nodes," *Physica A: Statistical Mechanics and its Applications*, vol. 392, no. 3, pp. 529–537, 2013.

[33] C. Studholme, D. Hill, and D. Hawkes, "An overlap invariant entropy measure of 3d medical image alignment," *Pattern recognition*, vol. 32, no. 1, pp. 71–86, 1999.