A parallel self-organizing overlapping community detection algorithm based on swarm intelligence for large scale complex networks

A Parallel Self-Organizing Overlapping Community Detection Algorithm Based on Swarm Intelligence for Large Scale Complex Networks

Hanlin Sun[a,b], Wei Jie[c], Jonathan Loo[c], Lizhe Wang[d,*], Sugang Ma[a,b], Gang Han[e], Zhongmin Wang[a,b], Wei Xing[a,b]

[a]*School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, China*
[b]*Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing, Xi'an University of Posts and Telecommunications,China*
[c]*School of Computing and Engineering, University of West London, UK*
[d]*Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, China*
[e]*Department of Electronic Science and Technology, Northwestern Polytechnical University, China*

Abstract

Community detection is a critical task for complex network analysis. It helps us to understand the properties of the system that a complex network represents and has significance to a wide range of applications. Though a large number of algorithms have been developed, the detection of overlapping communities from large scale and (or) dynamic networks still remains challenging. In this paper, a Parallel Self-organizing Overlapping Community Detection (PSOCD) algorithm ground on the idea of swarm intelligence is proposed. The PSOCD is designed based on the concept of swarm intelligence system where an analyzed network is treated as a decentralized, self-organized, and self-evolving systems, in which each vertex acts iteratively to join to or leave from communities based on a set of predefined simple vertex action rules. The algorithm is implemented on a distributed graph processing platform named Giraph++; therefore it is capable of analyzing large scale networks.

*Corresponding author

Email addresses: sunhanlin@xupt.edu.cn (Hanlin Sun), wei.jie@uwl.ac.uk (Wei Jie), Jonathan.Loo@uwl.ac.uk (Jonathan Loo), lizhe.wang@gmail.com (Lizhe Wang), msg@xupt.edu.cn (Sugang Ma), hangang668866@163.com (Gang Han), zmwang@xupt.edu.cn (Zhongmin Wang), xingwei@xupt.edu.cn (Wei Xing)

The algorithm is also able to handle overlapping community detection well because a vertex can naturally joins to multiple communities simultaneously. Moreover, if some vertexes and edges are added to or deleted from the analyzed network, the algorithm only needs to adjust community assignments of affected vertexes in the same way as its finding joining communities for a vertex, i.e., it inherently supports dynamic network analysis. The proposed PSOCD is evaluated using a number of variety large scale synthesized and real world networks. Experimental results indicate that the proposed algorithm can effectively discover overlapping communities on large-scale network and the quality of its detected overlapping community structures is superior to two state-of-the-art algorithms, namely Speaker Listener Label Propagation Algorithm (SLPA) and Order Statistics Local Optimization Method (OSLOM), especially on high overlapping density networks and (or) high overlapping diversity networks.

Keywords: overlapping community detection, community structure analysis, complex network analysis, swarm intelligence, parallel network analysis

---

1. Introduction

A lot of complex systems, such as the World Wide Web, mobile communication networks, online social networks, power grids, traffic road networks and so on, are often modeled as complex networks to investigate. A complex network usually shows some interesting properties such as high network transitivity, power-law degree distribution, small world, scale free, the existence of community structures, and much more. The study of community structures can help us to understand those systems at a mesoscopic level, just between the macroscopic level in which the whole system is considered and the microscopic level in which each node is analyzed individually. In addition, the analysis of community structures has significance to many applications. For example, community structure analysis can be used in social networks (e.g. Facebook) which presents relationships between members. The analysis of such networks will help to design reliable friend recommendation systems. As another example, community structure analysis can be used in detecting communities of customers with similar purchasing interest in e-business networks. This can lead to setting up efficient product recommendation systems and thus improving business opportunities for product retailers.

An exact definition of a community depends on the underlying problem and its application, thus there is no a unanimous definition. For example, the definition could be based on degrees of vertexes [1], k-cliques[2], k-clans[2], k-clubs [2], etc. Filippo et. al. [1] gave out the definition of strong sense community and weak sense community according to member connection strength. Michele et. al. [3] proposed a number of meta definitions. Intuitively, a community is a group of vertexes in a network that has more edges (connections) among its members but comparatively has less edges between its members and the rest of the network vertexes. This simple concept is the core of nearly all community definitions.

The properties of very complex networks induce three staple challenges for a community detection algorithm: (1) overlapping community structure detection, especially from a high overlapping density network of which a large percent of vertexes are overlapping vertexes, and (or) from a high overlapping diversity network of which an overlapping vertex belongs to a great number of communities; (2) large scale network analysis, e.g., the number of vertexes and edges could reach the scale of several millions and even more; and (3) dynamic changing of the analyzed networks topology, i.e., a number of vertexes and edges could appear or disappear frequently. The problem about large scale and dynamic networks is how to find community structures within them quickly with as less effort as possible. Designing efficient algorithms to meet these problems remains challenging.

In this paper, we develop the Parallel Self-Organizing Community Detection (PSOCD) algorithm based on the idea of swarm intelligence (SI) to further near to a final solution. Swarm intelligence is the collective behavior of decentralized and self-organized systems, either natural or artificial. An SI system generally consists of a large number of simple individuals who can only perform simple actions and interact with nearby neighbors as well as with the system existing environment. Intelligence will emerge as a consequence of the sum of these simple actions and interactions. The main innovations and contributions of this paper are:

(1) We proposed the PSOCD algorithm applying concepts of SI. In PSOCD, an analyzed network is modeled as a SI system in which each vertex as an individual decides its own actions, i.e. leaving its original communities or joining into new communities, depending on a set of predefined simple action rules. Eventually an optimal community structure will emerge whilst each vertex acts iteratively. A vertex is naturally allowed to join to multiple communities, thus it is able to find overlapping community structures. The

algorithm inherently supports dynamic network analysis very well; in that, if new vertexes and edges are added to the analyzed network, or existing vertexes and edges are deleted, it only needs to adjust community associations of affected vertexes in the same way as finding their previous joining communities.

(2) We implemented the PSOCD in a distributed manner on the parallel graph processing platform Giraph++[4], leveraging the properties of SI, namely distributed, self-organizing, and self-evolving. As a result, it is capable of handling large scale network analysis.

(3) We found that the extended modularity or modularity density [5] for evaluating overlapping community structure quality should not be used as a metric for performance comparison of different algorithms. The reason is that the incorporation of overlapping properties in the metrics has great impact on its values, thus may lead to a quite opposite conclusion.

(4) We applied the PSOCD to analyze structural communities of three large scale real world networks and got reasonable results, which are closer to the functional communities reported in previous studies than those discovered by the two compared state-of-art algorithms.

The remainder of this paper is structured as follows: In section 2, some most related overlapping community detection algorithms are briefly reviewed. In section 3, the design of the PSOCD algorithm is outlined. A current implementation of the proposed algorithm on the platform Giraph++ and its computational complexity are described in section 4. In section 5, the evaluation results of the algorithm for a number of synthesized and real large complex networks are presented, and the limits of the current implementation are discussed. Finally, section 6 concludes the paper.

2. Related works

Community structure analysis of complex networks has attracted much interest, and a number of algorithms originating from different fields, such as physics, statistics, data mining, evolution computation and many more have been proposed. There are many different strategies behind these community detection algorithms, such as divisive hierarchy, agglomerative hierarchy, random walking, information diffusion, spectrum analysis, statistical inference and so on. Several comprehensive reviews of these methods have been conducted, for example, a survey of community discovery methods was provided with a special focus on techniques designed by statistical physicists [6]. Meta

definitions of a community in complex network were given and majority community discovery methods were summed up based on their own definitions [3]. Overlapping community structure analysis algorithms were reviewed in [7] and [8], while those for social network analysis were reviewed in [9]. The performance of a number of algorithms were compared in [8] and [10]. In this section, we briefly review some algorithms most related to our work.

2.1. LPA

The label propagation algorithm (LPA) is currently the fastest algorithm for community structure analysis, with a near-linear computational complexity. The idea is that, as information propagates on a network with a community structure, it will have a high probability flowing within a community. At first, each vertex is assigned a label, indicating the community to which it belongs, then each vertex sends its label to its neighbors and selects a label received from neighbors, e.g., the label observed most frequently, as its new label. By iteratively propagating labels among neighboring vertexes, the community structure will gradually emerge. Assuming that a vertex is able to hold more than one label, LPA can be extended for overlapping community detection. There are a number of improved algorithms based on the LPA approach for overlapping community detection, such as COPRA(Community Overlap PRopagation Algorithm)[11], MLPA(Multi-Label Propagation Algorithm)[12], BMLPA(Balanced MLPA)[13], SLPA(Speaker-Listener LPA)[14], LPAcw(LPA with consensus weight)[15], DLPA(Dominant LPA)[16], etc. They are differentiated by the way of label propagating strategies and new label selections.

We take the SLPA as a comparison algorithm, which is said having good performance as detecting overlapping communities[8]. The algorithm mimics human communication behavior as propagating labels, i.e. preferring to spread most frequently discussed opinions. Specifically, each vertex has a memory and spreads randomly one of its current labels to its neighbors with a probability proportional to the occurrence frequency of the label in memory, and takes the most popular label observed as a new label. Those labels with probabilities exceeding a given threshold are kept by a vertex and transformed to communities finally.

There are parallel extensions of LPA [17, 18] to further speeding up their execution. However, the problem of LPA algorithms is that its result is unstable due to the inner randomness of the algorithm.

2.2. Game Theory

The algorithms based on game theory try to simulate a procedure through which the community structure of a network evolved to the current state. In such an algorithm, a vertex is viewed as a rational or selfish individual, and decides its own community associations according to a defined utility function, that consists of two parts, a gain function and a loss function. While a Nash equilibrium is reached, at which state no individual can increase its utility by changing its strategy (community associations) unilaterally, a community structure could be deduced. If an individual is allowed to join into multiple communities, the algorithm could be used for overlapping community detection. A Nash equilibrium is guaranteed to exist if the gain and loss functions are locally linear. However, finding a Nash equilibrium under such restricts is NP-hard. In practice, a local equilibrium is used instead, in which each individual plays its local optimal strategy. The algorithms in [19] and [20], PSGMAE(Pearson correlation GAME) [21], NGGAME(Neighborhood similarity GAME)[21], and SID(Social Information Diffusion) [22] are just some examples based on game theory. Their differences lie within the design of the utility functions they employ. Specifically, the COFOGA in [23] defines a utility function for a coalition (community) and uses the game theory as an approach for coalition merging, i.e. an individual in the formation game is a coalition. Since a utility function should be local linear to assure the game being potential, the community properties contained in the utility function should be able to be expressed as local linear functions.

2.3. Swarm Intelligence

Extensively, the LPA algorithms and those based on game theory are similar as the one ground on swarm intelligence as we have described previously, in the way that each vertex decides its community associations by itself, but the ideas behind are different: the LPA relays on the fact that a label is propagating within a community with high probability; the game theory aims to find a Nash equilibrium state by each selfish agent selecting proper strategies iteratively; while the swarm intelligence assumes an emergence of optimal result from collective simple actions of a large number individuals. Therefore, the final best community structures found may have different qualities.

Actually, a swarm intelligence algorithm is more referred to an algorithm inspired by natural bio-systems, and ranges over a number of types, including the Genetic Algorithm (GA) and the Ant Colony Optimization (ACO) algorithm. Both of the two algorithms were used for community analysis. The

GA-NET+[24] is GA based and used for overlapping community detection by working on a link network translated from a normal vertex network, in which a vertex and an edge represent an edge and a vertex of the original network, respectively. Due to the limitation of the representation method for evolutionary individuals, however,the GA based algorithm can not be used for large scale network analysis. In [25], the AntCBO algorithm was proposed for overlapping community detection. Intrinsically, this algorithm is a type of LPA but the label propagation among neighboring vertexes is achieved by 'ants' in the ACO algorithm framework.

In [26] the authors proposed an overlapping community detection algorithm named COGS (Community Optimization Graph Swarm), that is explicitly said using swarm intelligence. In COGS, an analyzed network is treated as a swarm intelligence system and a vertex interacts with its neighbors to find the so called Friendship-Group (a type of partial community). Then the algorithm finds Friendship-Groups that should be merged based on the LPA idea through propagating community labels among such groups. Our algorithm is different from COGS in that the COGS uses SI to find Friend-Groups, only a partial step in community detection, while the PSOCD uses SI as a framework to imitate a procedure through which the community structure of a network evolves to its current state. The authors also implemented a multi-thread parallel version COGS[27].

2.4. Local Expansion

In local expansion algorithms, a seed vertex or vertex-set is given firstly as an initial community, and then neighboring vertexes of member vertexes join to the seed community if their joinings can improve the community quality. As there is no such a vertex left, a complete community is found. A new seed (vertex or vertex-set) is then selected from those vertexes not joined to any community yet and a new community is discovered in the same way as previously described. Given a seed community, if all other vertexes can try to join to it, but not only those not joined to any community yet, this strategy can be used for the detection of overlapping communities. The algorithms IS(Iterative Scan)[28], RaRe(Rank Removal)[28], IS^2(Improved IS)[29], LFM[30], fast LFM[31], DOCS(Detecting Overlapping Community Structures)[32], MOSES(Model-based Overlapping Seed ExpanSion) [33], and OSLOM(Order Statistics Local Optimization Method)[34] are just some examples. However, the results of such an algorithm heavily depends on the quality of the selected seeds.

The OSLOM is another comparison algorithm we used in this paper, that is reported having good performance for overlapping community detection as well[8]. It should be also mentioned that the OSLOM is capable of detecting communities in networks accounting for edge directions, edge weights, overlapping communities, hierarchies and community dynamics. The algorithm uses a fitness function that expresses the statistical significance of a cluster (community) with respect to random fluctuations to evaluate the cluster quality. The statistical significance of a cluster is defined as the probability of finding the cluster in a random null model, i.e. in a class of networks without community structure. To reduce the impact of stochasticity of initial seed selections, the algorithm is usually repeated several times to obtain several community structures, and then the final best outputs are selected from all detected clusters. One major drawback of OSLOM is its heavy computational cost as calculating significance for each cluster.

To the best of our knowledge, the problem of overlapping community structure analysis remains a challenge, though there are a large number of algorithms have been proposed. We should continue to further explore and develop algorithms that can handle large scale and dynamic networks and be able to detect overlapping community structure accurately and effectively.

3. The proposed PSOCD algorithm

The conception structure of the proposed algorithm is shown in Fig. 1. It mainly consists of four phases, namely, partitioning, initializing, evolving and collecting. In the partitioning phase, the algorithm divides an analyzed large scale network into a number of smaller sized sub-networks, which will be loaded into different worker nodes and be processed in parallel in subsequent phases. In the following initializing phase, PSOCD finds an initial community for each vertex within its located sub-network. These initial communities provide a foundation for the evolving phase. In the third phase, the algorithm evolves communities of each sub-network by changing community associations of each local vertex iteratively. Note that during parallel processing, messages are passed among these sub-networks to keep states of the evolving system be in consistent. An optimal community structure of the network will gradually emerge after a number of evolutions. Finally, in collecting phase, the community structure of the analyzed network is collected from the evolved sub-networks.

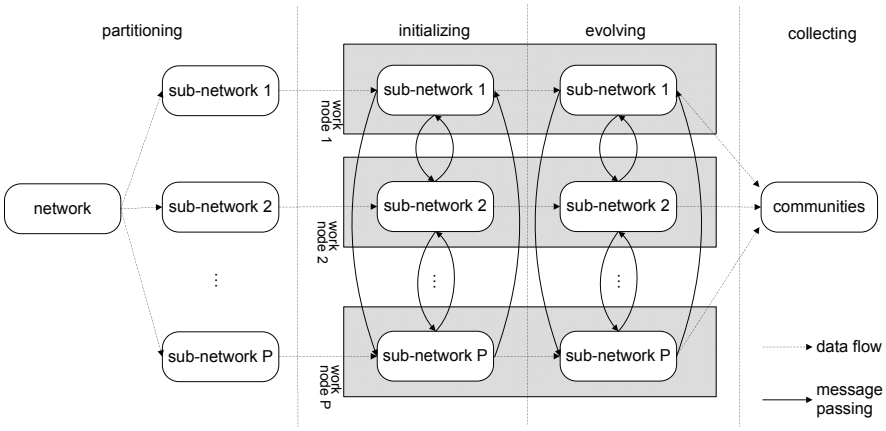

Figure 1: The conception structure of PSOCD.

3.1. Partitioning Phase

The first phase of analyzing a large scale network is to divide it into a number of smaller sub-networks with approximately the same size. Moreover, the connections among the identified different sub-networks should be minimal. The target of equally sized sub-networks roughly equalizes the processing time for each sub-network. Minimal connections among the identified sub-networks is aimed for because the connections between sub-networks have a significant influence on the performance of the subsequent evolving phase. For example, during its evolution, a vertex should notify its neighbors changes of its communities. It is easy to notify local neighbors located in the same sub-network, but if a neighbor is hosted by another sub-network, the vertex must send community- change messages across a communication network. Therefore, a vertex is preferred to be assigned to a sub-network with as less connections with other sub-networks (i.e., less external neighbors) as possible. Currently, the 'Metis' algorithm[35] is used as the partition method, which could produce a partition structure with minimum edges across partitions.

3.2. Initializing Phase

As mentioned earlier, member vertexes of a community have denser connections among them, but comparatively less connections with members of other communities. There is no more denser connected part in a network than a k-clique, of which each member is connected to all the rest members

ALGORITHM 1: InitializeCommunities(sub-network)

```
for (each vertex v in the sub-network) do
    if (v has been initialized) then
        continue
    end if
    if (v and its two uninitialized neighbors n1 and n2 form a 3-clique) then
        create a new community nc containing v, n2 and n2;
    else
        create a singleton new community nc containing v;
    end if
    save nc in the local community structure lcs and associate it with its members;
    tag members of nc as being initialized;
end for
```

Figure 2: Finding an initial community for each vertex within a sub-network.

of the k-clique. In other words, a maximum k-clique represents the strongest sense of community. The PSOCD takes a smallest k-clique, 3-clique, as the initial core community for the three member vertexes. For simplicity, the algorithm finds a 3-clique for an uninitialized vertex of a sub-network with its two uninitialized local neighbors as their initial community. If such a 3-clique does not exist for a vertex, then the vertex forms a community with only itself as a member, i.e. a singleton community. The initializing algorithm for a sub-network is shown in Fig. 2. Please keep in mind that the algorithm is executed in parallel by different worker nodes that are responsible for processing different sub-networks. We will explain in section 4.1 how communities are saved in our implementation to reduce communication cost thus get performance gain. Here, we simply consider that each vertex keeps its own state, namely its neighbors, neighbors' neighbors, joining communities, neighbors' joining communities, and so on.

To find if a 3-clique with two neighbors exists, a vertex must be able to check if its two neighbors are connected mutually to each other. Hence a vertex must know its neighbors' neighbors while executing the initializing algorithm. This is achieved by each vertex sending each of its neighbors its all neighbors firstly. The sending procedure is shown in Fig. 3. The sent

ALGORITHM 2: NotifyVertexNeighbors(sub-network)

```
for (each vertex v in the sub-network) do
   for (each neighbor ng of v) do
      send ng a message containing v's neighbors;
   end for
end for
```

Figure 3: Notifying neighbors of a vertex its all neighbors.

ALGORITHM 3: NotifyNewCommunities(sub-network)

```
for (each community c in lcs of the sub-network) do
   for (each newly joining vertex jv of c) do
      for (each neighbor ng of jv) do
         if (ng is not a member of c) then
            send ng a message containing community c;
         end if
      end for
   end for
end for
```

Figure 4: Notifying neighbors of newly joining members the community.

neighbor notification messages (and other messages) will be handled by the receiving vertexes.

In the subsequent evolving phase, a vertex will check each of its neighbors if it should join to the neighbor's communities. From the viewpoint of a community, neighbors of members (but not members yet) of a community should know the existence of the community to make them have a chance to join to. For the sake of such existence knowledge, the PSOCD sends neighbors of newly joining members of a community messages containing the community. The community notification algorithm is shown in Fig. 4. Especially, notifications to external neighbors make the community expand freely across sub-networks.

3.3. Evolving Phase

The evolving phase plays a key role in finding an optimal community structure for a network. The main question in evolving phase is whether a vertex should join to a community, i.e., under which condition a vertex will join to a community? To address this, we define a new type of connection strength of a vertex with a community, the connection score.

3.3.1. Connection Score

Tanmoy et. al. [36] claimed that the connection strength of a vertex with a community is determined by two factors: 1) the number of connections between the vertex and each other community, but not the total number of connections between the vertex and all other communities, and 2) the strength with which this vertex connects to the candidate community, but not only the number of connections between the vertex and the candidate community. The strength is measured as the clustering coefficient of the vertex's neighbors belonging to the candidate community. The larger the coefficient, the stronger the strength, and vice versa. Tanmoy et. al. proposed a connection strength, the vertex permanence, defining as follows:

$$Perm(v) = \left[\frac{I(v)}{E_{max}(v)} \times \frac{1}{D(v)}\right] - [1 - c_{in}(v)] \tag{1}$$

where $I(v)$ is the connection number between vertex v and a candidate joining community, $E_{max}(v)$ is the maximum number of connections between v and candidate communities, $D(v)$ indicates the degree of vertex v, and $c_{in}(v)$ represents the clustering coefficient of v's neighbors belonging to the candidate community. The first term in eq. 1 considers the first factor described previously, while the second term takes into account the second. The range of the permanence of a vertex is between $[-1, 1]$. The authors advocated that the averaged permanence of all community members could be used as an index for the quality of the community.

However, it is not straightforward to depict a connection strength as a negative number. Intuitively, a negative number indicates the extent to which a vertex is not belonging to a community. Moreover, the two factors are measured independently in the permanence computation, while they should be related to each other in some way to get subtle distinction between connection strengths. At this point we propose a new type of connection strength,

the connection score (CS), following the two mentioned factors:

$$CS(v) = \left[\frac{I(v)}{D(v)}\right]^{(1-c_{in}(v))} \tag{2}$$

where $I(v)$, $D(v)$ and $c_{in}(v)$ are the same as in eq. 1. The $D(v)$ could be replaced with $E_{max}(v)$ as well. In CS, a connection strength is first measured by the connection number, and then magnified by the corresponding clustering coefficient. Therefore, the CS may get a subtle distinction between connection strengths. The range of CS lies within $[0, 1]$ representing absolutely not belonging to (0) and definitely belonging to (1) a community, respectively. It is easy to extend CS to weighted networks, by replacing the degree with sum of corresponding edge weights and defining the clustering coefficient in a way taking edge weights into consideration.

3.3.2. Vertex Community Evolving

A vertex either joins to communities to which some of its neighbors belong or stays as a singleton community. Therefore, in each evolution round, the algorithm updates a vertex's communities as follows: first, get currently joining communities of all neighbors of the vertex as candidate communities, to which the vertex will try to join. Note that two neighbors may join into the same community, thus the duplicated ones should be removed from the candidates for efficiency. Second, compute the connection score for the vertex with each candidate community. Finally, add the vertex into the candidate communities with 'stronger' strength, and remove the vertex from originally joining communities which are not joined to in this iteration of evolution. The connection strength of which the ratio between it and the maximum connection score exceeds a given threshold is considered as stronger. Actually, the connection strength ratio threshold defines what a community is found by the algorithm: a set of vertexes more densely connected compared to other vertexes in neighborhood. The threshold value controls the size of found communities. A large threshold leads to small size communities, while a small threshold incurs large size communities. Moreover, it must be noticed that only if the connection number of a vertex with a community is greater than or equal to 3, could the connection strength be computed, because the clustering coefficient is meaningful only if this condition is hold. If the connection number equals 2, a vertex will join to such a candidate community only if its maximum connection is not more than 3. Otherwise a

vertex will not join to a candidate community. The vertex community evolving algorithm is described as in Fig. 5. A vertex who does not change its joining communities is considered as 'inactive', and does not need to execute the evolving algorithm. An inactive vertex becomes active again if it receives any message from another vertex.

The evolving algorithm is executed iteratively for each sub-network, and optimal communities will emerge gradually. Theoretically, the evolving will terminate if there is no vertex left that is changing its communities. However, such a graceful termination may not be achieved due to a few vertexes who, in a cyclic way, repeat the same actions within several sequential evolving generations, i.e. leaving from a community and later joining to it again, because of mutual influence of actions of different vertexes. We set a maximum evolving generations to ensure it will terminate eventually.

At the end, in collecting phase, the whole community structure of the analyzed network could be collected from these evolved sub-networks.

3.3.3. Keeping Consistence

To keep community state being consistent, the changes of a community, i.e. some vertexes joining and some leaving, should be notified to its replicas, if applicable during evolution. Given that the PSOCD runs in a distributed manner, one way of notifying these changes could be sending associate messages after all vertexes of a sub-network finishing their eovlutions. One community change notification algorithm is shown in Fig. 6. Notifying neighbors of newly joining vertexes is for computing clustering coefficient in connection score calculation.

3.3.4. Community Merging

It could happen that a community is contained in another one completely during evolution. These contained communities should be merged (deleted) to eliminate unnecessary vertex actions and thereby speed up the algorithm' execution. Our algorithm merges communities from a vertex viewpoint, i.e., it checks each community a vertex joining to if it is contained by another community the vertex joining to. The reason is that if a community is contained by another one, the common members of the two communities must join into both, and thus the merging could be found easily by searching within a common member's joining communities. This search strategy greatly reduces the effort needed to find a containing community in local community structure for a given community.

ALGORITHM 4: EvolveCommunities(sub-network)

```
for (each vertex v in the sub-network) do
    if (v is inactive) then
        continue;
    end if
    get joining communities of v's neighbors as candidates;
    compute connection score cs for v with each candidate;
    for (each candidate community cc) do
        if (connection number cn of v with cc ≥ 3) then
            if (((cs with cc) / cs_max) ≥ threshold) then
                v joins to cc;
            end if
        else if (((cn with cc) == 2) and (cn_max ≤ 3)) then
            v joins to cc;
        end if
    end for
    for (each community c that v joins to originally) do
        if (v does not join to c now) then
            v leaves from c;
        end if
    end for
    if (v does not change its joining communities) then
        set v as inactive;
    end if
end for
```

Figure 5: Evolving communities of vertexes in a sub-network.

1:
2: **ALGORITHM 5**: NotifyVertexActions(sub-network)
3:
4: **for** (each community c in lcs of the sub-network) **do**
5: **for** (each current member cm of c) **do**
6: send cm a message containing newly joining vertexes (and their neighbors) and leaving vertexes of c;
7: **end for**
8: **end for**
9:

Figure 6: Notifying vertex actions.

While deleting a community, the deletion could be notified to members of the community to make the deleting only occur once and thus get performance gain. By notifying other members after the first deleting on one member, no more deletion is needed any more. To reduce notification cost, if a merging occurs, only local members of the community need to be notified, but external members located in other sub-networks need not. It is because that if a deleted community exists in a sub-network, the containing community must exist too. Therefore, the same merging will occur there as well. However, there is a special case needed to be carefully dealt with to make sure the same merging occur in different sub-networks. The case is the situation that the contained and containing communities contain completely same members but with different IDs. It should be guaranteed that in such a situation the deleted community must be the same one. The PSOCD fulfills this requirement by always deleting the community with a small ID. The merging algorithm is shown in Fig. 7.

3.4. Post Process

Due to the possible ungraceful termination of the algorithm, there may exist some vertexes joining to wrong communities. As an extreme example, a vertex may join to a community, but has no connection with any member of the community. We finally apply a post-process procedure (shown in Fig. 8) to the collected community structure for rectification. The post algorithm repeatedly removes vertexes from wrongly joining communities. The leaving criteria include: 1) the connection number of a vertex to a community is 0; 2) the connection number is 1 but its maximum connection number is greater than or equal to 2; 3) the connection number is 2 but the maximum

1:
2: **ALGORITHM 6**: MergeCommunities(sub-network)
3:
4: **for** (each vertex v in the sub-network) **do**
5: get v's joining communities;
6: sort communities firstly by their sizes, then by community IDs;
7: **for** (each small community sc) **do**
8: **if** (sc is same as another community) **then**
9: delete the community with small ID from lcs;
10: notify local members of the deleted community the deletion;
11: **else if** (sc is contained by a big community bc) **then**
12: delete sc from lcs;
13: notify local members of sc the deletion;
14: **end if**
15: **end for**
16: **end for**
17:

Figure 7: Merging communities from a vertex viewpoint.

connection number is greater than or equal to 4; and 4) the connection number is greater than or equal to 3, while the connection score could be computed, but the ratio of the connection score to the maximum score is less than a given threshold, which is set the same as the joining threshold during PSOCD evolving. These leaving criteria are approximate the opposites of these joining rules in vertex community evolving algorithm (algorithm 4).

4. Implementation of PSOCD

A number of distributed graph processing systems have been developed to meet the challenge of rapidly processing growing graph data. Almost all these systems divide an input large scale graph into a number of small sub-networks and take the 'think like a vertex' programming model to achieve iterative graph computation. The Apache Giraph is such a system that adopts the BSP (Bulk Synchronous Parallel) computation model. In BSP, a computation consists of numbers of super-steps, in each of which each vertex of a sub-network does its own computation and sends its results to other related vertexes by messages through the system scheduler, no matter if the destination vertex is located in the same local sub-network or other

ALGORITHM 7: PostProcess(community structure)

repeat
  for (each vertex v in network) **do**
    compute connection number cn and connection score cs of v to each joining community;
    for (each community c v joins to) **do**
      if (c is singleton) **then**
        continue;
      end if
      if ($cn == 0$) **then**
        v leaves from c;
      else if ($cn == 1$ **and** $cn_{max} \geq 2$) **then**
        v leaves from c;
      else if ($cn == 2$ **and** $cn_{max} \geq 4$) **then**
        v leaves from c;
      else if ($cn \geq 3$ **and** cs / $cs_{max} <$ threshold) **then**
        v leaves from c;
      end if
    end for
    if (v joins to none community) **then**
      v forms a singleton community;
    end if
  end for
until (no vertex leaves from a community)

Figure 8: Post Process algorithm.

sub-networks. The system will guarantee to deliver these messages to their correct destination vertexes at the end of the current super-step, i.e., after all vertexes finishing their computation of the current super-step. Tian et. al. [4] improved the Giraph and proposed the Giraph++ which provides a 'think like a graph' programming model. In Giraph++, the sub-network structure is opened up to all local vertexes and thus vertex communication within a sub-network can bypass the heavy message passing or system scheduling support and messages can be sent to local destination vertexes directly. Therefore, the communication mechanism of Giraph++ can be viewed as semi-asynchronous, within local sub-network asynchronous and between subnetworks synchronous. This optimized local asynchronous communication will make the swarm intelligence system evolution be more efficient. We implemented our PSOCD on the Giraph++ platform.

4.1. Considerations

In a swarm intelligence system, each individual keeps its state and interacts with its neighbors and the system environment to make its own decisions independently. For the community detection problem the state of a vertex includes its neighbors, neighbors' neighbors, joining communities, external neighbors' communities, and more. Clearly, the communities saved by vertexes of a sub-network have a lot of replicas. To keep these replicas being consistent, messages must be sent to each vertex who holds a replica. Note that in Giraph++, a sub-graph programming model is supported, such that all local vertexes of a sub-network could share some variables among them. Therefore, in our implementation, communities used locally are held by a sub-network and identified by unique identifiers across the whole system. We call them the local community structure. A vertex then only saves IDs of communities to which itself joins or its external neighbors join. There are two advantages in following this approach: 1) the consumed memory is reduced; 2) the messages needed to keep community replicas being consistent are decreased greatly. The reason for that is if one vertex updates the state of a community, all local vertexes referencing this community will be aware of the changes immediately as well. Consequently, those messages related to community state update (community notification messages and vertex action notification messages) do not need to be sent to local vertexes. Moreover, those messages do not need to be sent to each of relevant external vertexes: select a representative vertex from message heading vertexes located in a same sub-network and send only one such a message to the selected vertex,

then the representative vertex is capable of being responsible for updating its local community replica. Therefore, such a message needs to be sent for the times of at most the number of sub-networks. In addition, when a vertex notifies an external neighbor of its joining communities, it only sends the IDs of these communities, which is much smaller in size than the sending of whole structures of these communities. The implementations of aforementioned notification algorithms could be improved by using the advantages of the community storage approach.

As the evolving phase progresses, some communities may no longer be referenced by any vertex. To remove these unnecessarily saved communities from the local community structure of a sub-network, at the beginning of each evolving round, the algorithm should examine all communities and delete those null referenced ones.

4.2. Alogrithm Framework

The framework of PSOCD on Giraph++ is shown in Fig. 9. The functions starting with 'Process' deal with messages received from last super-step for vertexes in a sub-network and thus update states of local communities and vertexes. The 'Notify' functions here are improved ones by taking into considerations of local shared community storage approach. The first two super-steps, step 0 and 1, achieve the initializing phase. Note that the two functions in super-step 1, 'NotifyNewCommunities' and 'NotifyVertexCommunities' (described in Fig. 11), spread information of the initialized communities across subnetworks and prepare for subsequent community evolving. The remained super-steps compute the community evolution. In the even super-steps, the algorithm processes received notification messages and updates states of local communities and vertexes and then evolves communities of each local vertex. At the end it sends messages of community evolution (vertex actions) to related subnetworks aiming to keep changed communities be in consistent. Similarly, in the odd super-steps, the algorithm first handles notification messages of community evolution and renew local communities, and then merges contained communities if possible. Finally, it diffuses current states of local vertex joining communities and prepares for the next round evolving. The corresponding flow diagram is shown in Fig. 10.

4.3. Computational Complexity Analysis

It is difficult to analyze the computational complexity of a parallel algorithm in distributed running. In this section, we focus on the algorithm

```
1:
2: ALGORITHM 8: Compute(sub-network)
3: ---------------------------------------------------------------
4: if (super-step == 0) then
5:    //initializing.
6:    NotifyVertexNeighbors(sub-network);
7: else if (super-step == 1) then
8:    //initializing.
9:    ProcessNeighborMessages(sub-network);
10:   InitializeCommunities(sub-network);
11:   NotifyNewCommunities(sub-network);
12:   NotifyVertexCommunities(sub-network);
13: else if ((super-step % 2) == 0) and (super-step ≤ MAXSTEP)) then
14:   //evolving, even step.
15:   ProcessNewCommunityMessages(sub-network);
16:   ProcessVertexCommunityMessages(sub-network);
17:   EvolveCommunities(sub-network);
18:   NotifyVertexActions(sub-network);
19: else if ((super-step % 2) == 1) and (super-step ≤ MAXSTEP)) then
20:   //evolving, odd step.
21:   ProcessActionMessages(sub-network);
22:   MergeCommunities(sub-network);
23:   NotifyNewCommunities(sub-network);
24:   NotifyVertexCommunities(sub-network);
25: else
26:   ProcessActionMessages(sub-network);
27:   set all local vertexes to be halt;
28: end if
29: ---------------------------------------------------------------
```

Figure 9: The framework of PSOCD on Giraph++.

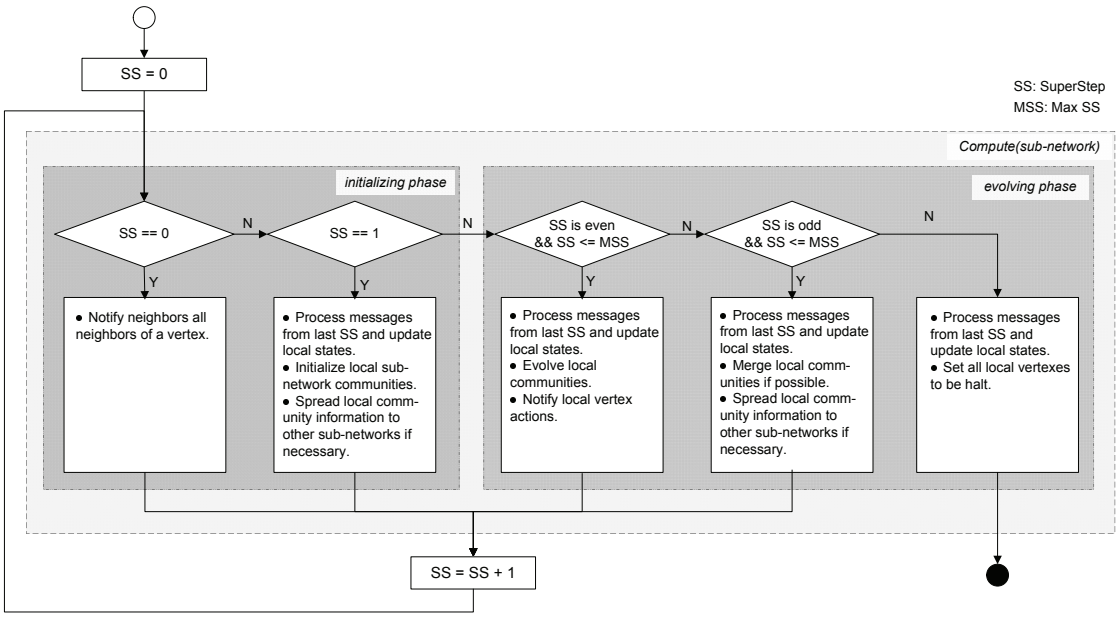

Figure 10: The flow diagram of PSOCD on Giraph++.

execution on one sub-network and analyze the computational complexity of the two major operations of PSOCD, i.e., the initializing phase and evolving phase. The used labels are listed in Table 1. We assume that all vertexes are divided into approximately equal sized sub-networks and that network communication time is not taken into consideration.

The initializing phase consists of two super-steps. In the first (super-step 0), each vertex notifies its neighbors all its neighbors. In the second (super-step 1), a sub-network processes firstly the received neighbor notification messages and then finds a 3-clique initial community for each local vertex. Finally, it spreads initialized local communities to other sub-networks if necessary. The computational complexity of notifying neighbor information is $O(\bar{N}_V \cdot \bar{N}_N)$, while that of processing neighbor notification messages is $O(\bar{N}_V \cdot \bar{N}_{EN})$. Local neighbor notifications can be done at the same time as notifying in super-step 0 due to the opening of sub-network structure in Giraph++ and none message is actually sent, thus no processing. A vertex is an external neighbor of its external neighbors, too. The worst computational complexity of finding initial communities for all local vertexes is $O(\bar{N}_V \cdot \bar{N}_{LN}^2)$. If a community contains members having neighbors located

1:
2: **ALGORITHM 9**: NotifyVertexCommunities(sub-network)
3:
4: **for** (each vertex v in the sub-network) **do**
5: **for** (each neighbor ng of v) **do**
6: **if** (ng is a local vertex) **then**
7: notify ng v's joining community IDs directly;
8: **else**
9: send ng a message containing v's joining community IDs;
10: **end if**
11: **end for**
12: **end for**
13:

Figure 11: Notifying joining communities of vertexes.

in other sub-networks, then it should be spread to these external neighbors. In consideration of the storage approach of communities in a sub-network, the community spreading could be achieved by the two functions shown in algorithm 8 (Fig. 9), NotifyNewCommunities and NotifyVertexCommunities (algorithm 9, in Fig. 11). The former sends a local community, if necessary, to the selected representative vertexes located in other sub-networks, and the later sends the joining community IDs of each local vertex to all its neighbors. The worst computational complexity of the community notification is $O(\bar{N}_C \cdot (N_P - 1))$, and the computational complexity of vertex community ID notification is the same as its neighbor notification, $O(\bar{N}_V \cdot \bar{N}_N)$. As a result, the worst computational complexity of the initializing phase is $O(2\bar{N}_V \cdot \bar{N}_N + \bar{N}_V \cdot \bar{N}_{EN} + \bar{N}_V \cdot \bar{N}_{LN}^2 + \bar{N}_C \cdot (N_P - 1))$. Usually, the $\bar{N}_N$, $\bar{N}_{LN}$ and $\bar{N}_{EN}$ are much smaller comparing with $\bar{N}_V$, and N_P is smaller than $\bar{N}_C$. In addition, within a sub-network having good community structure, finding a 3-clique with two local neighbors for most vertexes does not need $N_{vLN}(N_{vLN}-1)/2$ times searches, especially for vertexes with a large number of local neighbors. Therefore, the computational complexity of initializing could be simplied as $O(K_1 \cdot \bar{N}_V + K_2 \cdot \bar{N}_C)$ where K_1 and K_2 are two constants.

The evolving phase is executed a number of iterations and each iteration consists of two super-steps as well. In the first (even super-step), a sub-network firstly processes received notification messages of new communities and vertex community IDs from last super-step and updates states of local communities and vertexes, and then evolves community associations for each

local vertex. Finally, the sub-network sends updates of each local community (vertex joining and leaving) to selected representative vertexes located in other sub-networks if a replica of the community exists there for the purpose of keeping them being consistent. The worst computational complexity of processing new community messages is $O(\bar{N}_C \cdot (N_P - 1))$, while the complexity of processing vertex community ID messages is $O(\bar{N}_V \cdot \bar{N}_{EN})$. The worst complexity of vertex community evolving is $O(\bar{N}_V \cdot \bar{N}_N \cdot \bar{N}_{VC})$ because the algorithm checks all candidate communities for each vertex if it joins, and that of the vertex action notification is $O(\bar{N}_C \cdot (N_P - 1))$ as messages are only sent to selected representative vertexes.

In the second step (odd super-step), a sub-network processes firstly received vertex action notification messages and updates states of local communities, and then merges communities from a vertex viewpoint if possible. Finally, it prepares for the next round evolving by sending new community notification and vertex joining community ID notification messages, as in the end of initializing phase. The worst computational complexity of processing vertex action messages is $O(\bar{N}_C \cdot (N_P - 1))$, the same as their sending, and that of the merging is $O(\bar{N}_V \cdot \bar{N}_{VC}^2)$ for checking each community if it is contained in another one.

Therefore, the worst computational complexity of one evolving iteration is $O(\bar{N}_V \cdot (N_P - 1) + \bar{N}_V \cdot \bar{N}_{EN} + \bar{N}_V \cdot \bar{N}_N \cdot \bar{N}_{VC} + 3\bar{N}_C \cdot (N_P - 1) + \bar{N}_V \cdot \bar{N}_{VC}^2 + \bar{N}_V \cdot \bar{N}_N)$. It could be simplied as $O(K_3 \cdot \bar{N}_V + K_4 \cdot \bar{N}_C)$ where K_3 and K_4 are two constants due to the similar reasons explained in initializing phase analysis. It should also be noted that the $\bar{N}_C$ gradually becomes small as evolving forwarding.

When the overall cluster is considered, the complexity of the initializing or one evolving iteration depends on the worst one of the sub-networks processed by a number of worker machines, plus the worst network communication cost. Generally, it is a sensible assumption that the network bandwidth is sufficient and thus network communication will not be the bottleneck of the algorithm's execution.

5. Performance Evaluation

In this section the performance of our proposed algorithm will be evaluated and compared against two algorithms, the SLPA[14] and the OSLOM[34], that are reported having good performance for overlapping community detection[8] and of which the source codes are easily to get.

Table 1: Used labels in computational complexity analysis

Variable	Description
N_P	sub-network number
$\bar{N}_V$	average number of vertexes in a sub-network
$\bar{N}_C$	average number of local communities in a sub-network
$\bar{N}_N$	average number of neighbors of a vertex
$\bar{N}_{LN}$	average number of local neighbors of a vertex
N_{vLN}	the number of local neighbors of vertex v
$\bar{N}_{EN}$	average number of external neighbors of a vertex
$\bar{N}_{VC}$	average number of joined communities of a vertex

Table 2: Computational complexity of algorithms in PSOCD

Algorithm	Worst computational complexity
NotifyVertexNeighbors	$O(\bar{N}_V \cdot \bar{N}_N)$
ProessNeighborMessages	$O(\bar{N}_V \cdot \bar{N}_{EN})$
InitializeCommunities	$O(\bar{N}_V \cdot \bar{N}_{LN}^2)$
NotifyNewCommunities	$O(\bar{N}_C \cdot (N_P - 1))$
NotifyVertexCommunities	$O(\bar{N}_V \cdot \bar{N}_N)$
ProcessNewCommunityMessages	$O(\bar{N}_C \cdot (N_P - 1))$
ProcessVertexCommunityMessages	$O(\bar{N}_V \cdot \bar{N}_{EN})$
EvolveCommunities	$O(\bar{N}_V \cdot \bar{N}_N \cdot \bar{N}_{VC})$
NotifyVertexActions	$O(\bar{N}_C \cdot (N_P - 1))$
ProcessActionMessages	$O(\bar{N}_C \cdot (N_P - 1))$
MergeCommunities	$O(\bar{N}_V \cdot \bar{N}_{VC}^2)$

5.1. Experiment Setup

We implemented the PSOCD algorithm on the Giraph++ model and deployed it on a cluster with 20 virtual machines, one virtual machine as the master of the system and the other 19 as slavers (worker nodes). Each virtual machine has 4 core CPUs and 16G memory. The operating system used is Ubuntu 14.04, and the version of Hadoop deploying Giraph++ is 0.20.203.

The maximum computation super-steps in experiments is set as 30. Within the first half number of evolution iterations, the threshold of connection strength ratio of vertex joining is set as 0.8, in order to get a high quality core community structure, and then in the second half, the threshold is reduced to 0.5 or even less, for the purpose of getting more complete communities.

In the run of the SLPA algorithm, its parameter r takes all possible values and we picked out the best result as the compared one. The three algorithms are run 30 times on each test network.

Table 3: LFR parameter setting

Parameter	Description	Experiment Setting
N	number of vertexes	from 50,000 to 350,000, increased by 50,000.
k	average degree of vertexes	40
maxk	maximum degree	100
μ	mixing parameter	from 0.1 to 0.7,increased by 0.1.
t1	minus exponent for degree distribution	-2
t2	minus exponent for community size distribution	-1
minc	minimum community size	20
maxc	maximum community size	100
on	number of overlapping vertexes	from 10% to 80% of N, increased by 10%.
om	number of community memberships of overlapping vertexes	from 1 to 10,increased by 1.

5.2. Synthesized Networks

5.2.1. Network Data

First we use synthesized large complex networks as evaluation data set, since the exact real community structure could be used as the ground truth. The LFR model [37] is widely employed in performance evaluation of community detection algorithms and could generate a large scale network and its community structure. The model is characterized by parameters listed in Table 3. The mixing parameter $\mu = z_{out}/(z_{in} + z_{out})$ that gives the ratio between external degree of a node and its total degree . The z_{in} and z_{out} are the internal and external degree of a node with respect to its belonging community, respectively. As $\mu < 0.5$, the community structure of a network is well defined.

We generated four types of networks with different parameter combinations. In the first type, we changed the scale of networks by setting N from 50,000 to 350,000, increased by 50,000, and fixed other parameters. The mixing parameter μ is 0.3. The number of overlapping vertexes (on) is set as 10% of the total number of vertexes and the number of community memberships of overlapping vertexes (om) is set as 3. In the second type, we modified the μ from 0.1 to 0.7, increased by 0.1, and set N as 100,000, on as 10%, and om as 3. In the third type, we varied the on from 10% to 80% of total vertexes, increased by 10%, and set N as 100,000, μ as 0.3 and om as 3. Finally, in

the fourth type, we raised the *om* from 1 to 10, increased by 1, and set N as 100,000, μ as 0.3, and *on* as 10%. Other parameter-settings can be found in Table 3. There are 32 networks generated in total. Each network is divided to approximately equal-sized sub-networks, with size about from 20,000 to 30,000, using the Metis algorithm.

5.2.2. Community Structure Quality Measurement

To evaluate the quality of the detected community structure, we adopt the Overlapping Normalized Mutual Information (ONMI) to measure the similarity between the detected community structure and the real ground-truth one. Moreover, for overlapping community structure, the quality of detected overlapping vertexes should also be assessed. For this purpose, the recall, the precision, and the F-score metrics are employed.

- ONMI

The Normalized Mutual Information (NMI)[38] is a well-known entropy measure in information theory and is widely used for disjoint community structure evaluation. McDaid et. al. [39] developed a new version, which is defined as follows, for overlapping community structure evaluation:

$$\text{ONMI} = \frac{I(X:Y)}{max(H(X), H(Y))} \tag{3}$$

where X and Y are the detected community structure and the real one, $H(X)$ and $H(Y)$ are the entropies of the two community structures, respectively. The $I(X:Y)$ is the mutual information of X and Y, defined as:

$$I(X:Y) = \frac{1}{2}[H(X) - H(X|Y) + H(Y) - H(Y|X)] \tag{4}$$

where $H(X|Y)$ and $H(Y|X)$ are the conditional entropies of X and Y, respectively. More details could be found in [39].

The ONMI measures the quality of the whole community structure statistically. The larger the value, the more similar the two structures are. If the two compared structures are same, the ONMI equals 1. On the other hand, if they are totally different, the ONMI becomes 0.

It is worthy to mention that we applied the proposed post-process algorithm to all community structures detected by the three used algorithms for fairness. Moreover, we even applied the post-process to generated community

structures by LFR used as ground-truth. We found that in the generated structures, there are a number of wrong assignments where vertexes join to communities that they should not, and missing assignments where vertexes do not join to communities that they should, especially for networks with large μ, om, or on. However, the post-process could only correct the majority of wrong assignments, but has no effect on missing assignments. Fortunately, the missing assignments are much less than the wrong assignments in generated community structures. In addition, from the principle of PSOCD, it is a way to correct missing assignments, namely, missing assignments rarely exist in results by PSOCD.

- Quality of Overlapping Vertexes

The quality of detected overlapping vertexes should be measured from three aspects: 1) how many detected overlapping vertexes are true? 2) how many true overlapping vertexes are detected? and 3) are the numbers of community memberships of overlapping vertexes correct? We use the precision (denoted as P) to evaluate the first aspect and the recall (denoted as R) to assess the second. The third aspect is depicted statistically by distributions of community memberships of all overlappign vertexes.

The precision is defined as:

$$P = \frac{NumberOfCorrectlyDetectedOverlappingVertexes}{NumberOfTotalDetectedOverlappingVertexes}, \tag{5}$$

and the recall is:

$$R = \frac{NumberOfCorrectlyDetectedOverlappingVertexes}{NumberOfTrueOverlappingVertexes}. \tag{6}$$

Usually, the F-score (denoted as F), which is the harmonic mean of P and R, is used as an overall index of P and R. The F is defined as:

$$F = \frac{2R \cdot P}{R + P} \tag{7}$$

5.2.3. Results Analysis

For each used algorithm, we computed the average quality metrics and their corresponding 95% confidence intervals over the results of 30 runs for each test network. Fig.12 shows the ONMI results for test networks. As can

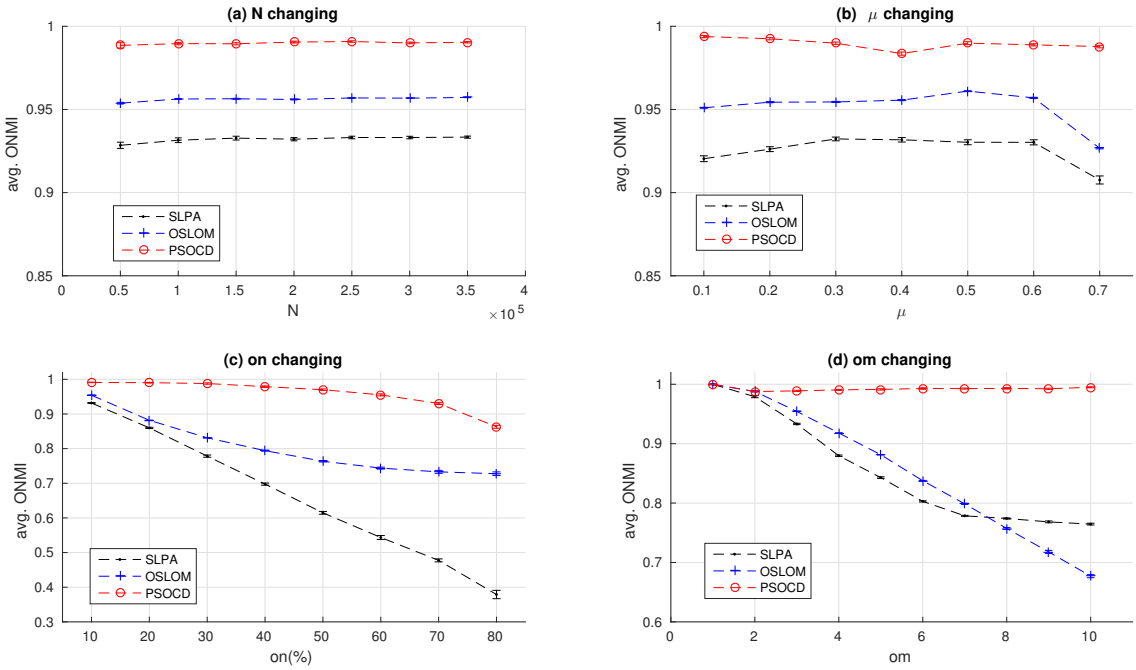

Figure 12: ONMIs of test synthesized networks.

be seen from the figure: 1) the ONMIs of PSOCD are the best for nearly all test networks, and those of OSLOM are the second best except for networks with very large *om*. 2) as N or μ increases, the ONMIs of all three algorithms are quite stable. 3) as *on* raises from 10% to 80% of total vertex number, the ONMIs of all three algorithms are decreasing greatly, except as the ratio reaching 70% and 80% for OSLOM where the ONMI is slightly increasing. However, the ONMIs of PSOCD are much better than those of the two others, particularly for large *on* networks, i.e. high overlapping density networks. 4) while the overlapping memberships *om* grows from 1 to 10, the ONMIs of SLPA and OSLOM are decreasing greatly, but those of PSOCD are much stable. It suggests that the PSOCD is more suitable for high overlapping diversity network (network with large *om*) analysis.

As for the quality of discovered overlapping vertexes, the precisions, recalls, and corresponding F-scores are shown in Fig. 13, 14 and 15, respectively. From Fig.13, it can be noticed that generally the precisions of SLPA and OSLOM are better than those of PSOCD. In fact, the precisions of SLPA are 100% for most test networks. However, the margins between the PSOCD

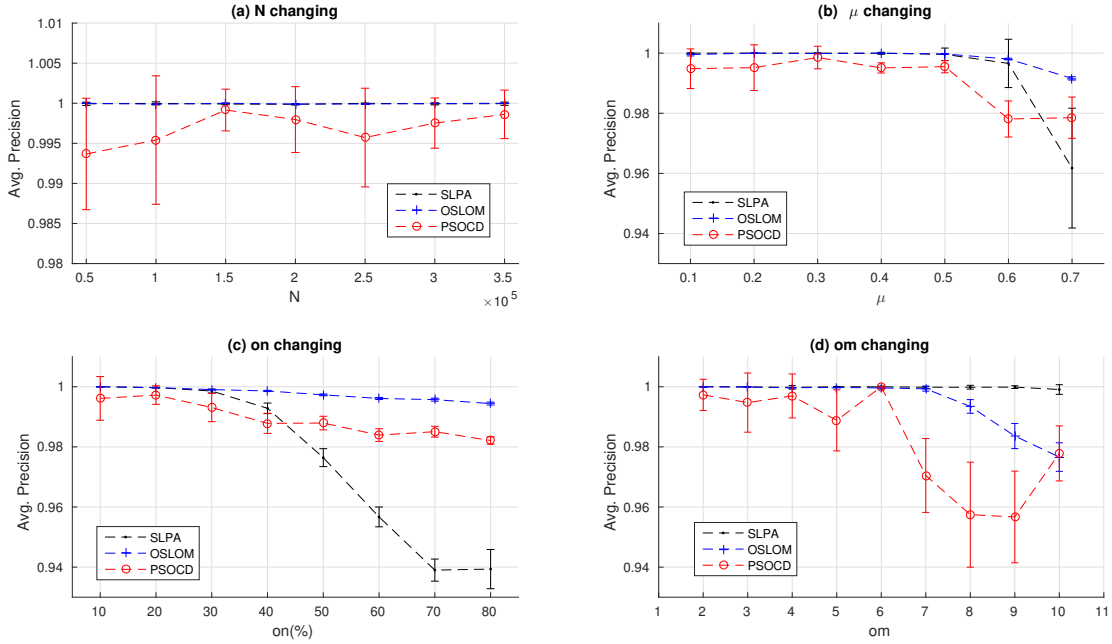

Figure 13: Precisions of test synthesized networks.

and the better one of the two others are very small, and the precisions of PSOCD are fairly good. Fig.14 demonstrates that for all test networks, the PSOCD could find out the overwhelming majority of true overlapping vertexes, and the recalls of PSOCD are much better than those of OSLOM and SLPA. Particularly, the recalls of SLPA could not be acceptable for networks with very large *on*. Fig.15 shows the F-scores for test networks, and it confirms that the performance of PSOCD is better than that of OSLOM and SLPA as a whole.

The distributions of detected community memberships of overlapping vertexes (excluding $om = 1$) from results of once run for networks with *om* varying are shown in Fig.16. The 'REAL' distributions are the results of the post-processed LFR generated community structures. It is clear that the PSOCD is capable of correctly identifying the majority of overlapping memberships. As a contrast, the SLPA and the OSLOM can only rightly recognize a small portion of overlapping memberships except for $om = 2$, and as *om* increases further, the detected overlapping memberships by them are dispersed in a wide range and thus the majority real overlapping memberships cannot be discerned.

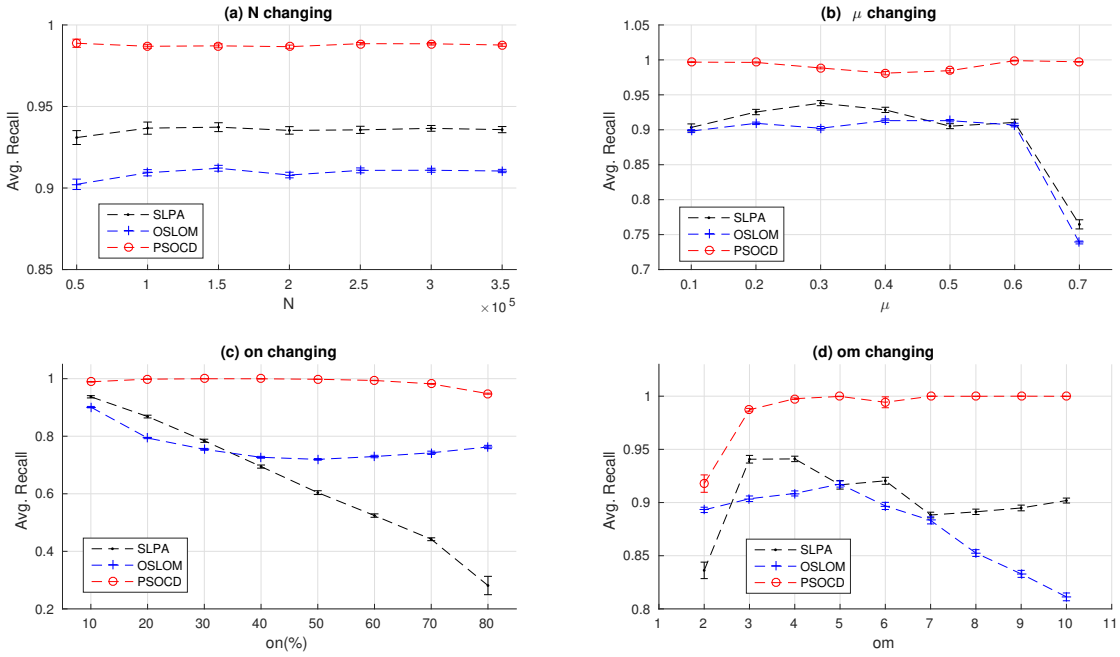

Figure 14: Recalls of test synthesized networks.

As a result, it is safe to conclude that the performance of the proposed PSOCD is better than that of the SLPA and the OSLOM, and the PSOCD is an appropriate choice for high overlapping density or high overlapping diversity network analysis.

5.3. Real Networks

5.3.1. Network Data

The advantage of using synthesized networks in evaluation is that the ground-truth community structures are known priorly. In result, the assessment of algorithm performance is easily conducted. However, the properties of real world networks could be more complicated than those of synthesized networks. We also tested our algorithm on three real, undirected and unweighted networks. They are *the Condense Matter Physics collaboration network* (CondMat), *the Computer science bibliography network* (DBLP) and *the Amazon co-purchased products network* (Amazon). All the three networks

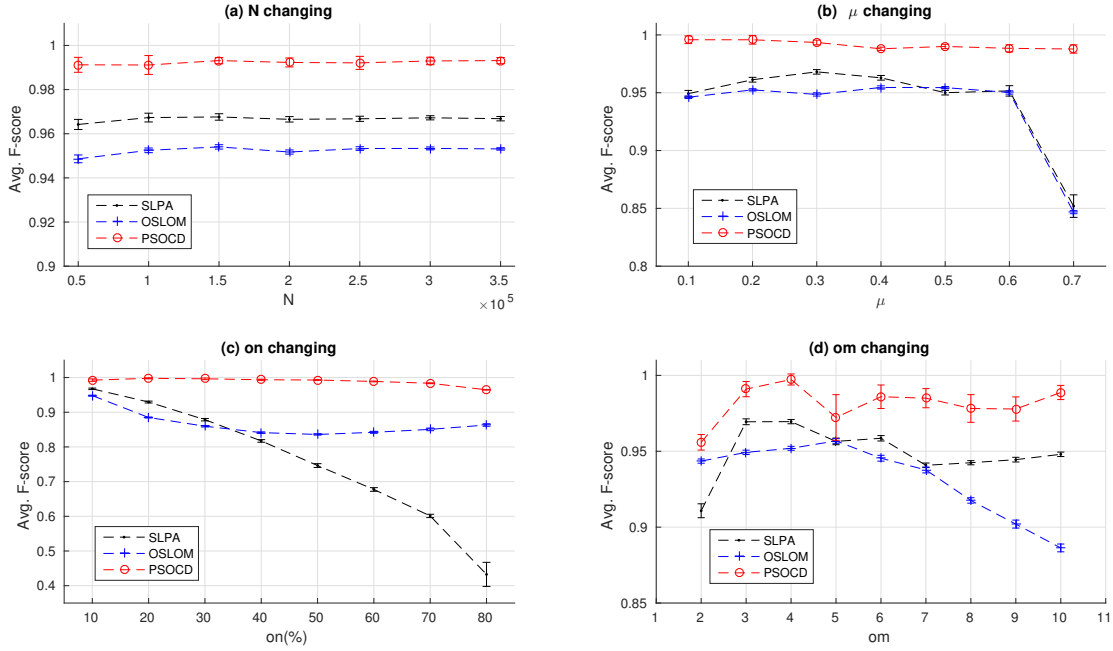

Figure 15: F-scores of test synthesized networks.

are downloaded from the Stanford Network Analysis Project (SNAP)[1]. The properties of the three real networks could be found in Table 4.

The CondMat network [40] is from the e-print Arxiv and covers scientific collaborations between authors of whom papers are submitted to Condense Matter category. A vertex in the network represents an author, and if author i co-authored a paper with author j, the network contains an undirected edge between i and j. The network covers papers in the period from January 1993 to April 2003 (124 months).

The DBLP (Digital Bibliography and Library Project) data [41] provides a comprehensive list of research papers in computer science. The SNAP constructs a co-authorship network where two authors are connected if they publish at least one paper together.

The Amazon network was collected by crawling Amazon website [42]. A vertex of the network is a product, and if product i is frequently co-purchased by customers with product j, the network then contains an undirected edge

[1]http://snap.stanford.edu/data/index.html

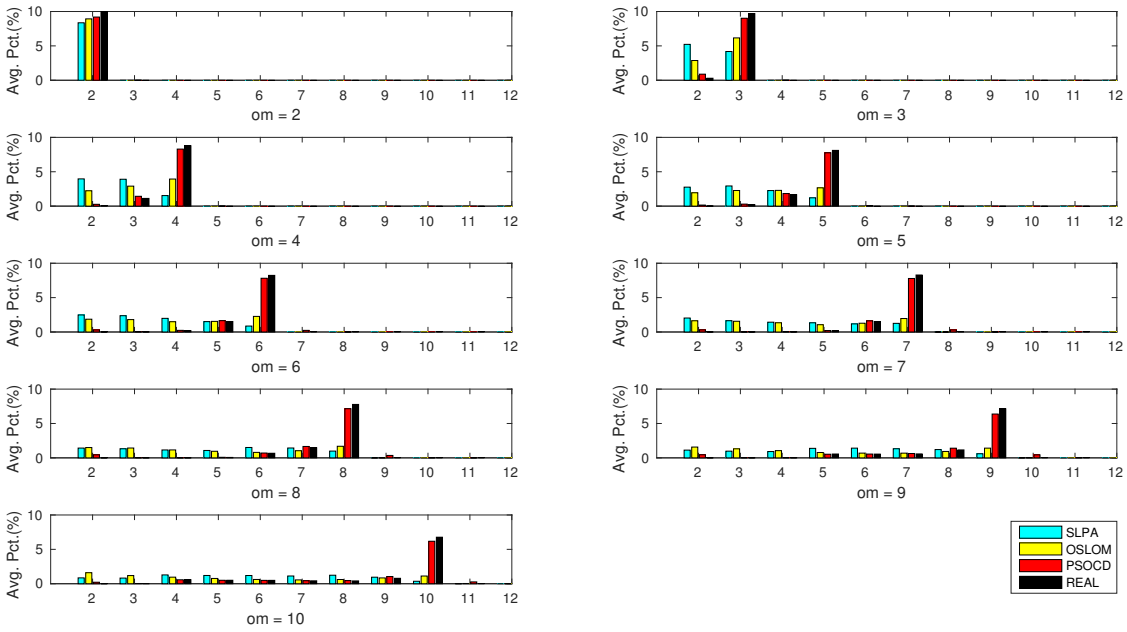

Figure 16: Distributions of overlapping memberships of synthesized networks.

Table 4: Properties of Real World Networks

Network	Vertex Number (V#)	Edge Number (E#)	Max Component V#	Max Component E#	Remained V#	Removed Ratio(%)
CondMat	23133	186878	21363	182572	19606	8.22
DBLP	317080	2099732	317080	2099732	271646	14.33
Amazon	334863	1851744	334863	1851744	305892	8.65

between vertex i and vertex j. It is reasonable to expect that the connections of the Amazon network is sparse, while those of the CondMat and DBLP network are dense, i.e. their topology structures are more complex.

We do not run community detection algorithms directly on these real world networks. Instead, first we find the maximum connected component of each network, for each such a component could be processed separately. The Amazon and the DBLP themselves are connected components. Then we trim these found maximum component networks by removing single connection vertexes iteratively until there is no such a vertex. The degree of a single connection vertex is one, thus if such a vertex joins to communities, it can only joins to communities to which its sole neighbor belongs. Therefore, it's not necessary to include these vertexes while processing a network. Besides, after removing these vertexes, the performance of analyzing could be

accelerated. The properties of trimmed maximum component networks are listed in Table 4, as well.

5.3.2. Quality of Community Structure

Generally, it is hard to assess the quality of a community structure of a real network, since a true community structure is difficult to define and is most usually unknown, especially for a large network. Though a 'ground-truth' community structure of the Amazon (and the DBLP) is provided by SNAP, it is a so-called functional community structure of which a community is a set of vertexes having a common function or role. For example, a 'true' community of the Amazon is a product category, and a 'true' community of the DBLP is a publication venue (a journal or a conference), namely, authors who publish papers in the same journal or conference belong to a community. However, a community discovered by almost all current algorithms, including PSOCD, is a so-called structural community which is defined on the connectivity pattern of a network. Whether a functional community structure could be effectively found by a structural community detection algorithm depends on if the functional communities exhibit a distinct connectivity pattern. In this paper, we focus on a structural community detection algorithm to extract high quality structural communities. We do not use the ONMI to evaluate detected structural community structures of real networks to avoid the impacts of ground-truth functional community structure constructions.

Another way to evaluate the quality of a community structure is mapping the structure to a score. The most famous one is the Newman's modularity, which is widely used to measure the quality of disjoint community structures. There are a number of metrics extended from the Newman's modularity for assessment of overlapping community structures. Chen et. al. [5] compared the most of extended modularities and other six metrics systematically. Moreover, they proposed a general extension modularity for overlapping community structure and extended the modularity density metric to enable its usage for overlapping community structure as well.

- General Extension Modularity for Overlapping Community Structure

After studying several versions of extended modularities, Chen et. al. [5] proposed a general extension modularity which is defined as follows:

$$Q_{ov} = \sum_{c \in C} \left[\frac{|E_c^{in}|}{|E|} - \left(\frac{2|E_c^{in}| + |E_c^{out}|}{2|E|} \right)^2 \right] \tag{8}$$

while c is a community from a community structure C, $|E_c^{in}| = \frac{1}{2}\sum_{i,j\in c} f(a_{i,c}, a_{j,c})A_{ij}$, $|E_c^{out}| = \sum_{i\in c}\sum_{c'\in C, c\neq c', j\in c'} A_{ij}$, and $|E| = \frac{1}{2}\sum_{ij} A_{ij}$. The $a_{i,c}$ ($a_{j,c}$) is the *belonging coefficient* of vertex i (j) to community c, and the $f(a_{i,c}, a_{j,c})$ is the *belonging function* which can be the product or average of $a_{i,c}$ and $a_{j,c}$. Based on the results of a number of experiments, the authors recommended to use the reciprocal of the community membership number as the belonging coefficient and the product of two belonging coefficients as the belonging function because of their effectiveness and simplicity. The A_{ij} is the ith row and jth column element of the adjacent matrix representing the processed network. If vertex i connects to vertex j, then $A_{ij} = 1$, otherwise $A_{ij} = 0$.

- Extension Modularity Density for Overlapping Community Structure

However, the Newman's modularity incurs two opposite yet coexisting problems, thus the extended ones: in some cases it tends to prefer small communities over large ones, while in others favor large communities over small ones. In literature, the later tendency is referred as the *resolution limit problem*. To address these two problems, the Modularity Density was proposed by integrating two additional components, the *split penalty* and the *community density*, into the modularity calculation. Chen et. al. [5] proposed an extension modularity density for overlapping community structure assessment as well. The extension modularity density is described as:

$$Q_{ds}^{ov} = \sum_{c\in C}\left[\frac{|E_c^{in}|}{|E|}d_c - \left(\frac{2|E_c^{in}| + |E_c^{out}|}{2|E|}d_c\right)^2 - \sum_{c'\in C, c'\neq c}\frac{|E_{c,c'}|}{2|E|}d_{c,c'}\right] \quad (9)$$

while $|E_{c,c'}| = \sum_{i\in c, j\in c', c\neq c'} f(a_{i,c}, a_{j,c'})A_{ij}$, $d_c = 2|E_c^{in}|/\sum_{i,j\in c, i\neq j} f(a_{i,c}, a_{j,c})$ and $d_{c,c'} = |Ec, c'|/\sum_{i\in c, j\in c'} f(a_{i,c}, a_{j,c'})$. The suggested belonging coefficient and belonging function were same as in the extension of modularity, i.e. the reciprocal of the community membership number and the product of two belonging coefficients, respectively.

Though the two mentioned metrics, especially the modularity density, give two reasonable gauges for the quality of an overlapping community structure, we found that they are not appropriate metrics for performance comparison between two overlapping community detection algorithms. The

reason behind is that the two metrics take overlapping membership number into consideration, and in the case if the overlapping community structures discovered by the two algorithms are quite different, for example, the overlapping memberships and overlapping vertexes detected by one algorithm are small and less, while those detected by the other one are high and mass, the conclusion deduced could be misleading. As an example, we generated a 100,000 vertex LRF synthesized network of which fifty percent vertexes are overlapping vertexes. The overlapping membership number om is set as 10, the μ is 0.3, and other parameters are same as listed in Table 3. The average ONMI (30 runs) of the community structures detected by PSOCD with respect to the post-processed ground-truth community structure is as high as 0.7672, while the average ONMI of SLPA is only 0.5064. There is no doubt that the quality of the community structures detected by PSOCD is much better than those by SLPA. However, the averaged extended modularity of the community strructures by PSOCD and SLPA are 0.1694 and 0.1865, respectively. As a result, a wrong conclusion that SLPA outperforms PSOCD could be deduced. The averaged extended modularity density of the results by PSOCD and SLPA are 0.06920 and 0.06148, respectively. Though a consistent conclusion as the ONMI values suggest could be inferred for this generated network, a false conclusion may be derived for some networks, just as the prior example in the extended modularity case, due to the same reason. Therefore, we suggest that the two metrics (and similar metrics) should not be used as indexes for performance comparison between different overlapping community detection algorithms.

- Community Structure Property Distribution

To reveal features of an overlapping community structure, Gergely et. al. [43] suggested to use four statistical distributions. They are: 1) the cumulative distribution of overlapping membership number; 2) the cumulative distribution of overlap size, that is the shared vertex number between two communities; 3) the cumulative distribution of community degree, that is the vertex degree of the corresponding community network formed as a vertex representing a community and an edge representing two communities are overlapping; and 4) the cumulative distribution of community size, i.e. the number of vertexes a community containing. The authors analyzed the k-clique community structures of three real networks. They found that the community degree cumulative distribution starts exponentially and

then crossovers to power-law, while the other three distributions obey power-law. These distributions uncover the modular structure of complex networks, however, they could not be used to rank community structures detected by different algorithms. We employ these distributions to describe features of discovered community structures in our experiments.

- Network Average Connection Strength

We define the average connection strength of a network based on the vertex connection score in equation 2 as a quality metric for overlapping community structure, that is:

$$CS_G = \frac{1}{|C|} \sum_{c \in C, |c| \geq 3} \frac{1}{|c|} \sum_{v \in c} CS(v) \tag{10}$$

while C denotes a community structure, $|C|$ represents the number of communities in C, c is a community belonging to C, and $|c|$ indicates the size of community c. Given that a community with only one or two members has no internal structure, we do not include such communities as computing the network average connection strength. We refer these communities with at least three members as a core community structure of a network. Generally, a community has more internal connections within its members, but less external connections with other communities. As for an overlapping community structure, the measurement of external connections in quality evaluation is meaningless, even harmful, because overlapping vertexes may have a large number of out connections. The CS_G presents a type of internal connection strength measurement and reflects the quality of a core community structure to some extent. The larger the value, the better the quality. It could be used as a metric for ranking different community structures.

5.3.3. Results Analysis

Table 5 lists some statistical properties of the detected community structures of the three real world networks by different algorithms. The SLPA$_{omod}$ and SLPA$_{omds}$ stand for the best results of SLPA algorithm selected from different r parameters according to the extended modularity and the extended modularity density, respectively. For an algorithm, the results from runs with different parameter settings are possibly not massively different because of same running principle, and thus it is reasonable to rank obtained results

Table 5: Properties (averages and 95% confidence intervals) of detected community structures of real world networks, averaged over 30 runs. $|\bar{C}|_{>2}$ is the averaged number of communities with size>2; $|\bar{C}|_{=2}$ is the averaged number of communities with size=2; $|\bar{C}|_{=1}$ is the averaged number of communities with size=1; OVR is the averaged overlapping vertex ratio; and CS_G is the network averaged connection strength.

Network	Algorithm	$\|C\|_{>2}$	$\|C\|_{=2}$	$\|C\|_{=1}$	OVR (%)	CS_G
CondMat	SLPA$_{omod}$	1375.9 ± 31.4	95.0 ± 24.7	138.0 ± 32.9	1.87 ± 2.31	0.8430 ± 0.0073
	SLPA$_{omds}$	1375.4 ± 32.3	100.9 ± 19.8	144.5 ± 21.4	0.64 ± 0.92	0.8415 ± 0.0078
	OSLOM	1529.8 ± 29.8	5.4 ± 5.1	354.9 ± 43.5	11.15 ± 0.49	0.8760 ± 0.0035
	PSOCD	1263.6 ± 21.2	26.7 ± 7.1	4728.8 ± 64.1	30.23 ± 0.76	**0.9208** ± 0.0031
DBLP	SLPA$_{omod}$	20137.7 ± 130.5	1632.9 ± 99.0	2060.0 ± 144.1	2.16 ± 0.69	0.8512 ± 0.0016
	SLPA$_{omds}$	20100.5 ± 126.6	1737.5 ± 70.3	2207.47 ± 107.8	0.52 ± 0.02	0.8503 ± 0.0016
	OSLOM	17384.3 ± 131.7	83.8 ± 16.7	5324.7 ± 121.1	7.47 ± 0.18	0.8709 ± 0.0012
	PSOCD	10406.6 ± 56.8	188.5 ± 17.6	114420.8 ± 324.0	17.60 ± 0.37	**0.9179** ± 0.0010
Amazon	SLPA$_{omod}$	21342.7 ± 120.7	1224 ± 100.0	2968.8 ± 152.9	1.45 ± 1.02	0.8155 ± 0.0020
	SLPA$_{omds}$	21317.5 ± 112.5	1292.5 ± 61.7	3064.3 ± 118.8	0.44 ± 0.04	0.8141 ± 0.0012
	OSLOM	17075.0 ± 84.3	5.3 ± 4.9	1316.2 ± 113.1	4.12 ± 0.12	0.8488 ± 0.0008
	PSOCD	7942.6 ± 28.8	15.7 ± 4.7	109297.9 ± 453.2	4.08 ± 0.22	**0.9016** ± 0.0006

according to the extended modularity or the extended modularity density and select the best one. All values in Table 5 are averaged over results of 30 runs. The associate 95% confidence intervals are given as well.

From the table, it could be seen that: 1) As for the number of communities, the PSOCD found the least core communities (community with size > 2) and the most one-size communities, while the two SLPA discovered the most two-size communities. The differences of detected community numbers between PSOCD and other algorithms are huge. Therefore, the detected community structures by different algorithms are quite different. We carefully checked the one-size communities and found that the sole member of such a community has at most one connection with other communities. In consequence, the existence of these vast one-size communities is reasonable. 2) From the aspect of overlapping vertexes, the variances of the results are great as well, and particularly those found by the two SLPA algorithms have large gaps. The SLPA$_{omds}$ found the least overlapping vertexes, while the PSOCD found the most ones except for the Amazon network. For the Amazon network, the results of OSLOM and PSOCD roughly agree to each other. 3) From the viewpoint of network average connection strength, the results of SLPA$_{omod}$ are slightly better than those of SLPA$_{omds}$, which can be attributed to the former found more overlapping vertexes. The results of OSLOM are better than those of the two SLPA algorithms, and the results of PSOCD are the best. Remind that we excluded the two-size and one-size communities as computing CS_G, therefore the gains of PSOCD are mainly

from the fact that it found a huge number of one-size communities. In other words, the PSOCD discovered better core community structures for these real networks.

We also present the four statistical distributions proposed in [43] to describe features of detected community structures by different algorithms. The results are also averaged over 30 runs. Fig. 17 depicts the cumulative distributions of overlapping memberships. As can be seen, the overlapping memberships found by PSOCD are quite different from those by OSLOM and SLPA. The PSOCD detects more overlapping vertexes and their overlapping memberships are distributed within a much wider range, which has a high upper bound, while the OSLOM and the SLPA can only find less overlapping vertexes and their associate overlapping memberships are much smaller. Though the PSOCD may overestimate at the end part of large overlapping memberships, just as shown in the synthesized network experiments, those overestimated take up only a small portion as shown in the figure. Specifically, the overlapping memberships detected by PSOCD for the Amazon network are similar to those by OSLOM, with the slight difference that for each large *om* the PSOCD finds more overlapping vertexes. It should also be noted that the trends of detected overlapping membership distributions of the CondMat by each algorithm are similar to those of the DBLP. However, the trends of the Amazon are much different from those of the two collaboration networks. This reflects the fact that the structure of the Amazon is quite different from the structures of the CondMat and the DBLP, just as what expected.

Fig. 18 displays the double log plots of the overlapping membership cumulative distributions by PSOCD, as they have more data points. From the figure, we can see that the cumulative overlapping membership distributions of the CondMat and the DBLP follow a two-phase power-law function. The power-law exponents are shown in the figure. The distribution of the Amazon network is not fit because there is only a small number of data points. In [44], Jebabli et. al. analyzed the functional community structure of the same DBLP network, and they reported that the distribution of cumulative overlapping memberships is power-law, too. Moreover, the range of detected overlapping memberships by PSOCD is comparable to the reported one in [44]. Thereby, we believe the quality of overlapping memberships discovered by our PSOCD for the three real networks is superior to those of other compared algorithms.

Fig. 19 depicts the log-log plots of community size cumulative distribu-

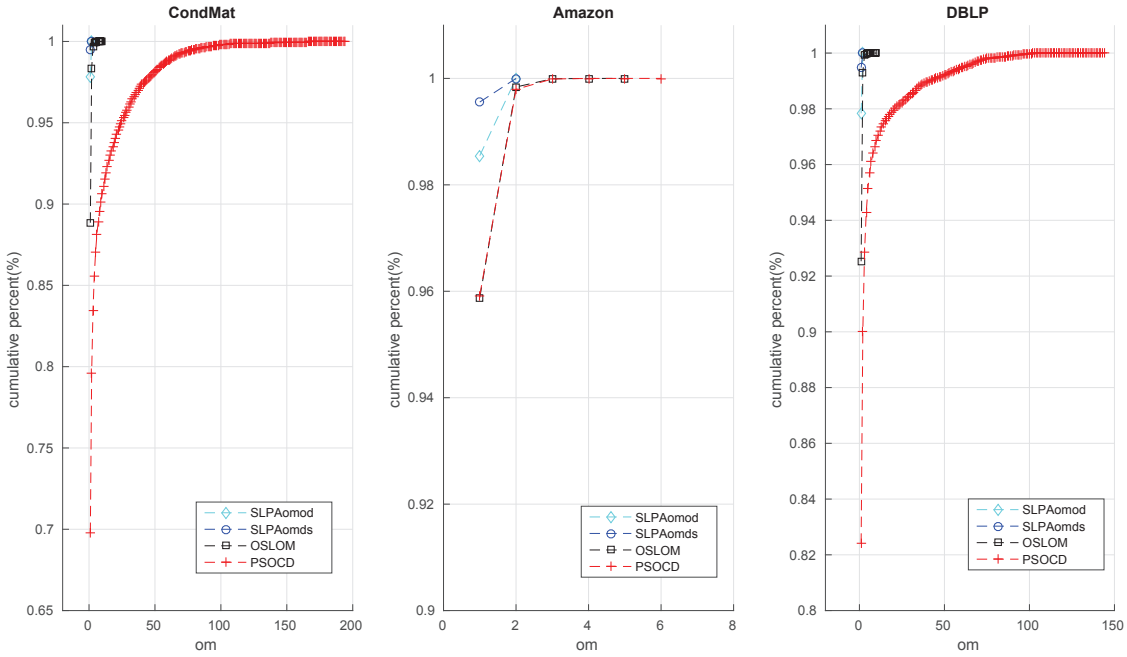

Figure 17: Overlapping membership cumulative distributions of real world networks.

tions of the three real networks. Since there are a huge number of one-size communities found by PSOCD, we omit the one-size community numbers to eliminate their effects on plot trends in cumulativeness depiction. It could be seen that the SLPA and the PSOCD could discover larger size communities comparing with the OSLOM, especially for the two collaboration networks. However, by carefully checking the values of large community sizes, we found that these values varied greatly, even in different runs of the same algorithm. Moreover, note that the end part of community size distributions, except the one of the CondMat network by PSOCD, are much flat, indicating that there are only a small number of large size communities. In fact, the numbers of the majority large size communities are just one. Therefore, we think the found large size communities are suspicious and need further processing, such as combining results of different runs or algorithms and then finding uncontroversial parts. As for the first part of community size cumulative distributions, it is clearly that they could be described by pow-law functions.Because the numbers of detected communities by different algorithms changes greatly, it is unfair to compare the frequency numbers of community sizes directly. We

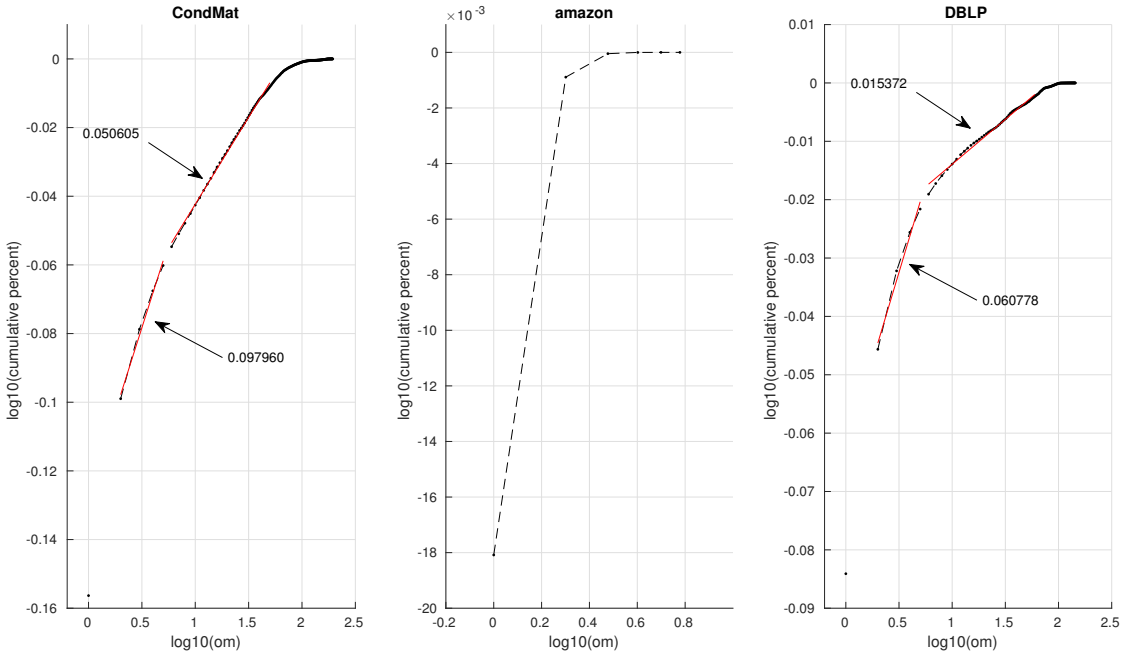

Figure 18: Double log of overlapping membership cumulative distributions of real world networks, detected by PSOCD.

focus only on the trends of cumulative distributions here.

The double log plots of community degree cumulative distributions of real networks are shown in Fig. 20. In the plots we omit the degree one as well, due to the similar reason as we do in community size cumulative distributions. It could be derived that the PSOCD could detect larger community degrees comparing with other algorithms. This suggests that the overlapping structures detected by PSOCD are more complex, namely, averagely a community is overlapping with more others. Once again, it can be seen that the first part of most distributions could be described by a pow-law function. We could also find a flat tail at most plots, and that indicates there are only a small number of large community degrees. For the distributions of the two collaboration networks by PSOCD, though they have increasing tails, their frequency numbers are comparably small as well. What is interesting is that the community degrees of the three networks discovered by the two SLPA algorithms are much different, though the numbers of detected communities by them are approximately equal. Generally, the SLPA$_{omod}$ could find larger

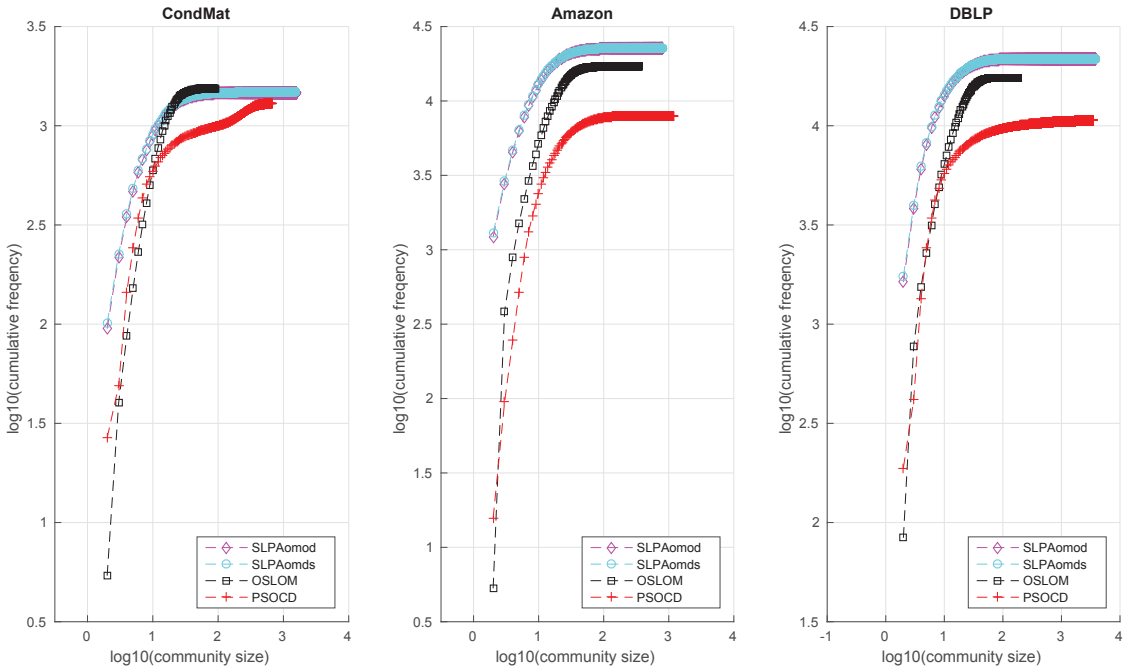

Figure 19: Double log of community size cumulative distributions of real world networks.

community degrees and more overlapped communities, i.e. more complex community structures.

Finally, Fig. 21 demonstrates overlap size cumulative distributions of the three real networks. At the first glance, it is clear that the PSOCD could find much larger overlap sizes, i.e. more complex overlapping structures. Again, the first part of most distributions could be fitted by a pow-law function. Similarly as community degree cumulative distributions, the overlap size cumulative distributions by the two SLPA algorithms are quite different. The plots suggest that the community structures found by SLPA$_{omod}$ are more complex than those by SLPA$_{mods}$.

Though there are several related researches, the difference lies in that we analyze the overall structural overlapping community structures of very large real networks, while they mainly focus on functional community structures. In [44], the authors analyzed the properties of functional community structure of the same DBLP network and its corresponding community network, and in [42], the same authors analyzed the properties of the community network of the Amazon functional community structures. Yang and Leskovec

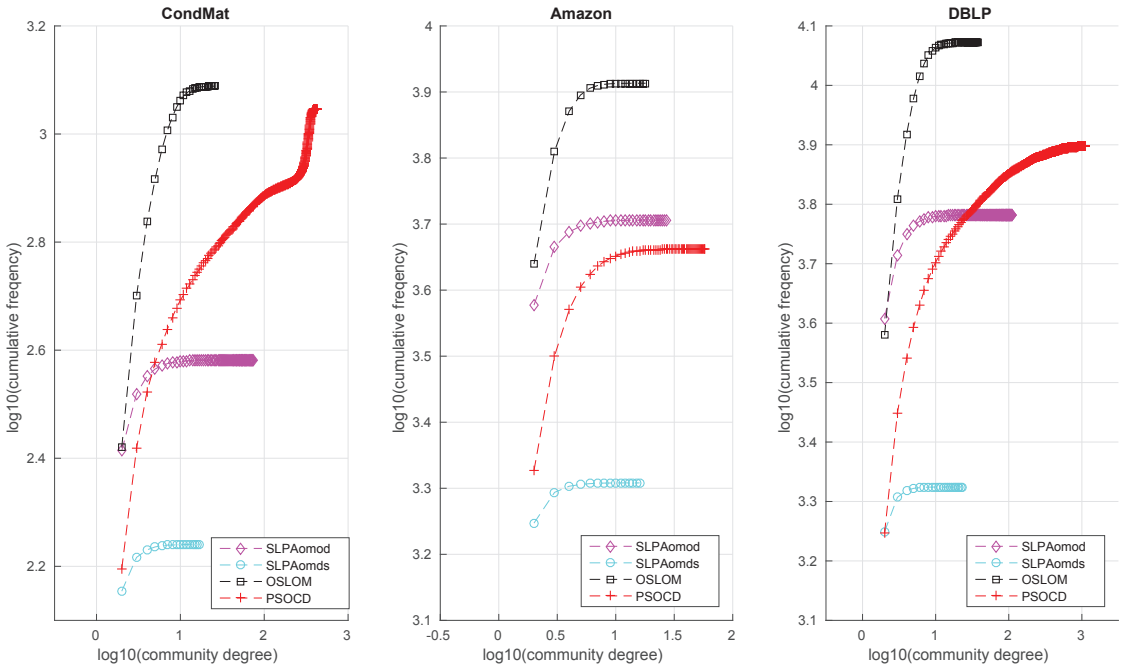

Figure 20: Double log of community degree cumulative distributions of real world networks.

[45] studied the functional community structures of a number of very large real networks. They proposed a seed based algorithm that finds all structural communities the seed vertex belongs to, and compared the detected communities with the functional ground-truth ones. Palla et. al. [43] explored the structural K-clique community structures of three real networks, but the max network size is only 30739. The K-clique overlapping community detection algorithm can hardly be used on very large networks due to its computational complexity.

The properties of structural community structures of the DBLP detected by PSOCD, such as the ranges of overlap memberships, community degrees and overlap sizes, are much closer to those of the functional community structures reported in [44]. In addition with the fact that the internal connection strengths by PSOCD are the best (shown in Table 5), we expect the quality of structural communities by PSOCD is superior.

To sum up, based 8on the experiment results of synthesized and real networks, we can safely state that our PSOCD outperforms the SLPA and

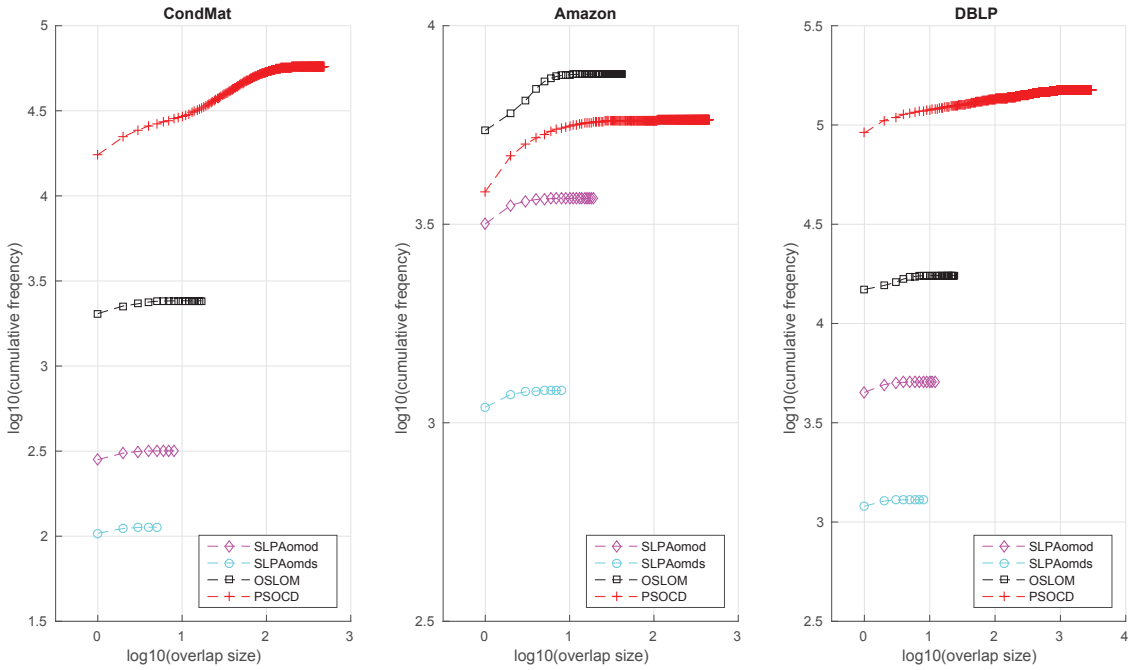

Figure 21: Double log of overlap size cumulative distributions of real world networks.

the OSLOM.

5.4. Discussion

The PSOCD is implemented in a distributed manner to be capable of processing large scale networks. One may wonder that the maximum network used in experiments has only 350,000 vertexes, though this size is much larger than the majority used in previous researches. However, if we increase vertex number further, say 400,000, the system will crash. The reason is that the current Giraph++ is developed on an old version Giraph, version 0.2, which has no message flow control mechanism and could crash if messages come too fast, but not the principle of PSOCD. The lack of message flow control has been fixed in new version Giraph, which could run on new version Hadoop to gain performance further. We believe the supported network size would be increased greatly if the PSOCD is implemented on a Giraph++ migrated to a new version Giraph.

Usually, a real network evolves, namely some vertexes and edges are added, and (or) some deleted. As a consequence, the corresponding com-

munity structure evolves, too. The classic community evolution includes emergence of a new one, death of an old one, expansion and contraction. The basic idea of PSOCD inherently has a good support for such a dynamic network analysis. The initializing phase could be executed properly on added vertexes and their direct neighbors to find newly created communities. The joining and leaving actions will lead to expansion and contraction of communities, respectively. If all members leave from a community, then this community is dead. Therefore, by proper implementation of the algorithm on a platform that supports network evolution, it could be used to analyze evolving community structures of dynamic networks.

Another main property of community structure is hierarchy, i.e., there exists a community structure in the community network recursively. Since the size of a community network is usually much smaller than the size of the associate original network, it is easy to construct a hierarchy structure if lowest level communities are given. What the PSOCD found is just a fine-grained and lowest level community structure. As a result, a hierarchy structure could be constructed easily by running PSOCD or other algorithms having good performance but with difficulty processing large scale networks recursively.

6. Conclusion

In this paper, we proposed a parallel self-organizing overlapping community detection algorithm named PSOCD that is based on the idea of swarm intelligence. The algorithm first divides an analyzed large scale network into a number of small size sub-networks and then treats each sub-network as a swarm intelligence subsystem. Each vertex of a sub-network, acting as an individual, can make its own decisions to join to or leave from communities mainly according to a predefined rule: if the ratio of its connection score to a candidate community to its maximum connection score exceeds a designated threshold, it joins to the candidate if it is not a member yet, otherwise it leaves from the candidate if it is a member already. The algorithm is implemented using the Giraph++ platform in a distributed environment, and the evolution of each sub-network is processed by a separate computation unit. Since vertexes of a sub-network may connect to vertexes of other sub-networks, it is important to keep joining community views of these vertexes and their external neighbors be consistent. The algorithm achieves this by passing messages of new communities and vertex actions among sub-networks

if necessary. By having all vertexes iteratively making their own joining or leaving decisions over a number of generations, an optimal community structure of the whole large network will be emerging gradually.

We tested our algorithm on a number of variety large scale synthesized networks and real world networks. The results of synthesized network analysis indicate that the PSOCD outperforms two state-of-art algorithms, the SLPA and the OSLOM, as finding overlapping community structures, especially on networks with high overlapping density and (or) high overlapping diversity. The results of real world network analysis show that the properties of the found structural communities by PSOCD are closer to those of the associate functional communities reported in previous literatures. In summary, we believe that our PSOCD is a superior approach for overlapping community structure analysis.

Acknowledgment

The authors would like to thank Amazon Web Services for providing resources for us to conduct experiments.

Funding: This work was supported by the Science and Technology Innovation Project of Shaanxi Province, China [grant number 2016KTZDGY04-01]; the Natural Science Foundation Research Project of Shaanxi Province, China [grant number 2016JM6048]; the Special Research Program of Education Departmentof Shaanxi Province, China [grant number 17JK0711]; the Special Funds for Construction of Key Disciplines in Universities in Shaanxi.

Reference

[1] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, Defining and identifying communities in networks, Proceeding of the National Academy of Sciences of the United States of American 101 (9) (2004) 2658–2663.

[2] G. Palla, I. Derenyi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (7043) (2005) 814–818.

[3] M. Coscia, F. Giannotti, D. Pedreschi, A classification for community discovery methods in complex networks, Statistical Analysis and Data Mining 4 (5) (2011) 512–546.

[4] Y. Tian, A. Balmin, S. A. Corsten, S. Tatikonda, J. McPherson, From 'think like a vertex' to 'think like a graph', in: Proceedings of the VLDB Endowment, Vol. 7, 2013, pp. 193–204.

[5] M. Chen, K. Kuzmin, B. K. Szymanski, Extension of modularity density for overlapping community structure, in: IEEE/ACM 4th Social Network Analysis and Applications (SNAA) Workshop at ASONAM, 2014, pp. 856–863.

[6] S. Fortunato, Community detection in graphs, Physics Reports 486 (3) (2010) 75–174.

[7] A. Amelio, C. Pizzuti, Overlapping community discovery methods: A survey, in: G. dc ., E.-U. A (Eds.), Lecture Notes in Social Networks, Springer, Vienna, 2014, Ch. Social Networks: Analysis and Case Studies, pp. 105–125.

[8] J. Xie, S. Kelley, B. K. Szymanski, Overlapping community detection in networks: The state-of-the-art and comparative study, ACM Computing Surveys 45 (4) (2013) 43.

[9] M. Planti, M. Crampes, Survey on social community detection, in: N. Ramzan, R. van Zwol, J.-S. Lee, K. Clver, X.-S. Hua (Eds.), Computer Communications and Networks, Springer, London, 2013, Ch. Social Media Retrieval, pp. 65–85.

[10] A. Lancichinetti, S. Fortunato, Community detection algorithms: a comparative analysis, Physical review E 80 (5) (2009) 056117.

[11] S. Gregory, Finding overlapping communities in networks by label propagation, New Journal of Physics 12 (10) (2010) 103018.

[12] Q. Dai, M. Guo, Y. Liu, L. Chen, Mlpa: Detecting overlapping communities by multi-label propagation approach, in: Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops., 2013, pp. 681–688.

[13] Z. Wu, Y. Lin, S. Gregory, H. Wan, , S. Tian, Balanced multi-label propagation for overlapping community detection in social networks, Journal of Computer Science and Technology 27 (3) (2012) 468–479.

[14] J. Xie, B. K. Szymanski, X. Liu, Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process, in: Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops., 2011, pp. 344–349.

[15] Z. Liang, F. Y. Jianping Li, P. Athina, Detecting community structure using label propagation with consensus weight in complex network, Chinese Physics B 23 (9) (2014) 098902.

[16] H. Sun, J. Huang, Y. Tian, Q. Song, H. Liu, Detecting overlapping communities in networks via dominant label propagation, Chinese Physics B 24 (1) (2015) 018703.

[17] K. Konstantin, S. S. Yousaf, S. B. K., Parallel overlapping community detection with slpa, in: 2013 International Conference on Social Computing, 2013, pp. 204–212.

[18] C. L. Staudt, H. Meyerhenke, Engineering parallel algorithms for community detection in massive networks, IEEE Transactions on Parallel and Distributed Systems 27 (1) (2016) 171–184.

[19] W. Chen, Z. Liu, X. Sun, Y. Wang, A game-theoretic framework to identify overlapping communities in social networks, Data Mining and Knowledge Discovery 21 (2) (2010) 224–240.

[20] H. Alvari, S. Hashemi, A. Hamzeh, Discovering overlapping communities in social networks: A novel game-theoretic approach, AI Communications 26 (2) (2013) 161–177.

[21] H. Alvari, S. Hashemi, A. Hamzeh, Detecting overlapping communities in social networks by game theory and structural equivalence concept, in: Proceedings of the third international conference on artificial intelligence and computational intelligence, 2011, pp. 620–630.

[22] A. Hajibagheri, H. Alvari, A. Hamzeh, S. Hashemi, Social network community detection using the shapley value, in: 2012 16th CSI International Symposium on Artificial Intelligence and Signal Processing, 2012, pp. 222–228.

[23] L. Zhou, K. L, P. Yang, L. Wang, B. Kong, An approach for overlapping and hierarchical community detection in social networks based on coalition formation game theory, Expert Systems with Applications 42 (24) (2015) 9634–9646.

[24] C. Pizzuti, Overlapped community detection in complex networks, in: Proc of the 11th Annual conference on Genetic and Evolutionary computation, 2009, pp. 859–866.

[25] X. Zhou, Y. Liu, J. Zhang, T. Liu, D. Zhang, An ant colony based algorithm for overlapping community detection in complex networks, Physica A: Statistical Mechanics and its Applications 427 (2015) 289–301.

[26] B. S. Rees, K. B. Gallagher, Overlapping community detection using a community optimized graph swarm, Social Network Analysis and Mining 2 (4) (2012) 405–417.

[27] B. S. Rees, K. B. Gallagher, Detecting overlapping communities in complex networks using swarm intelligence for multi-threaded label propagation, in: M. R., E. A., G. M. (Eds.), Studies in Computational Intelligence, Springer, Berlin Heidelberg, 2013, Ch. Complex Networks, pp. 111–119.

[28] J. Baumes, M. K. Goldberg, M. S. Krishnamoorthy, N. Preston, Finding communities by clustering a graph into overlapping subgraphs, in: Proceedings of the IADIS International Conference on Applied Computing, 2005, pp. 97–104.

[29] J. Baumes, M. Goldberg, M. Magdon-Ismail, Efficient identification of overlapping communities, in: Proceedings of the 2005 IEEE International Conference on Intelligence and Security Informatics, 2005, pp. 27–36.

[30] A. Lancichinetti, S. Fortunato, J. Kertesz, Detecting the overlapping and hierarchical community structure of complex networks, New Journal of Physics 11 (2009) 033015.

[31] Y. Li, Z. Zhu, A fast method of detecting overlapping community in network based on lfm, Journal of Software 10 (7) (2015) 825–834.

[32] F. Wei, W. Qian, W. Chen, A. Zhou, Detecting overlapping community structures in networks, World Wide Web 12 (2) (2009) 235–261.

[33] M. Aaron, H. N. J, Detecting highly overlapping communities with model-based overlapping seed expansion, in: Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining, 2010, pp. 112–119.

[34] A. Lancichinetti, F. Radicchi, J. J. Ramasco, S. Fortunato, Finding statistically significant communities in networks, PLoS ONE 6 (4) (2011) e18961.

[35] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM Journal on Scientific Computing 20 (1) (1998) 359–392.

[36] T. Chakraborty, S. Sriram, G. Niloy, On the permanence of vertices in network communities, in: Proc of the 20th ACM SIGKDD international conference on Knowledge discovery and data miningt, 2014, pp. 1396–1405.

[37] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Physical Review E 78 (4) (2008) 046110.

[38] L. Danon, J. Duch, A. Diaz-Guilera, A. Arenas, Comparing community structure identification, Journal of Statistical Mechanics: Theory and Experiment 2005 (2005) 09008.

[39] A. F. McDaid, D. Greene, N. Hurley, Normalized mutual information to evaluate overlapping community finding algorithms (Aug 2013).
URL `https://arxiv.org/abs/1110.2515v2`

[40] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: Densification and shrinking diameters, ACM Transactions on Knowledge Discovery from Data (ACM TKDD) 1 (1).

[41] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, in: 2012 IEEE 12th International Conference on Data Mining, 2012, pp. 745–754.

[42] M. Jebabli, H. Cherifi, C. Cherifi, A. Hamouda, Overlapping community detection versus ground-truth in amazon co-purchasing network, in: 2015 11th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS), 2015, pp. 328–336.

[43] P. Gergely, D. Imre, F. Illes, V. Tamas, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (7043) (2005) 814–818.

[44] M. Jebabli, H. Cherifi, C. Cherifi, A. Hamouda, Overlapping community structure in co-authorship networks: a case study, in: IEEE 7th International Conference on u- and e-Service, Science and Technology, 2014, pp. 26–29.

[45] Y. Jaewon, L. Jure, Defining and evaluating network communities based on ground-truth, Knowledge and Information Systems 42 (1) (2015) 181–213.