网络出版时间：2018-03-27 21:30:09
网络出版地址：http://kns.cnki.net/kcms/detail/51.1277.U.20180327.2130.008.html

复杂网络的重叠社区发现并行算法

滕飞[1,2]， 戴荣杰[1]， 任晓春[2]

（1.西南交通大学信息科学与技术学院，四川成都 610031；2.轨道交通工程信息化国家重点实验室（中铁一院），陕西西安 710043）

摘要：随着网络规模的快速增长，传统社区发现算法难以处理大规模网络数据和满足复杂网络的可扩展分析需求。本文提出一种适用于大规模复杂网络的重叠社区发现算法 PHLink。该算法根据复杂网络的无标度特性将节点建立连边的原因进行分析和归类，用以识别网络中具有重叠性的社区结构，并采用 MapReduce 计算框架对网络进行分割和冗余存储，减弱了图计算的耦合性，解决了社区发现算法的分布式计算问题。通过真实网络测试， PHLink 算法可以大幅度降低了边计算的复杂度，对于无标度特性明显的复杂网络提取 0.1%的枢纽节点即可节省 90%以上的计算量，较传统算法具有较高的稳定性和准确性，并且在 Hadoop 平台有良好的加速性和伸缩性，可以处理千万级连边规模的大规模复杂网络。

关键词： 复杂网络；重叠社区；社区发现；无标度；并行算法；Hadoop

中图分类号： V221.3 **文献标识码：** A

大规模网络的无标度特性更强，如 youtube、skitter 网络，极少数的点拥有绝大多数的连边，仅选择度最大的 0.1%的节点即可节省 94%以上的计算量。

A Parallel Algorithm for Overlapping Community Detection in Complex Networks

TENG Fei[1,2] ，*DAI Rongjie*[1] ，*REN Xiaochun*[2]

（1. School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031,China；2.State Key Laboratory of Rail Transit Engineering Informatization (FSDI), Xi’an 710043, China）

Abstract: With the increasing size of complex networks, traditional detection methods are difficult to scale to large networks. This paper proposes a parallel hierarchical link (PHLink) algorithm to discover overlapping communities. By studying power-law degree distribution of complex networks, we distinguished the motivations why two nodes intend to establish a connection， so that the complexity for community detection is reduced. PHLink also implemented distributed computing for large-scale networks, through graph segmentation and redundantly storage. Experiments validate that PHLink could accurately discover network communities as well as the overlaps between communities. For scale-free networks, removing 0.1% hub nodes reduces 90% of the whole computation time. Meanwhile, PHLink on Hadoop could scale to very large complex networks with millions of edges.

Key words: complex network; overlapping community; community detection; scale-free; parallel algorithm; Hadoop

收稿日期：2014-00-00

基金项目：国家自然科学基金资助项目（61573292），四川省软科学研究计划资助项目（2016ZR0034）

作者简介：滕飞（1984-），女，副教授，研究方向为数据挖掘与云计算， E-mail：fteng@swjtu.edu.cn

通信作者：任晓春（1962-），男，教授级高级工程师，研究方向为云计算与轨道工程信息化， E-mail：renxcne@163.com

引用格式：滕飞，戴荣杰，任晓春. 复杂网络的重叠社区发现并行算法［J］. 西南交通大学学报，2018,40（9）：1000-1005

TENG Fei, DAI Rongjie, REN Xiaochun. A Parallel Algorithm for Overlapping Community Detection in Complex Networks [J].Journal of Southwest Jiaotong University ,2018,40(9)：1000-1005

随着信息技术的迅速发展，人类社会已经迈入了复杂网络时代，各种超大规模网络不断涌现，例如智能电网、电话网络、社交网络、引文网络等。大规模复杂网络中的一些社会化特征在全局层面往往具有稳定的统计规律，可用来理解人类社交关系的结构和行为。社区发现根据网络的拓扑结构探索网络关系的群组特征，识别出网络中有意义的、自然的、相对稳态的社区结构，对网络信息的搜索与挖掘、舆论控制、信息推荐以及网络演化预测具有重要价值。

早期的社区发现算法主要研究的是非重叠社区结构。随着人们对网络性质认识的加深和新型网络不断出现，非重叠社区成员属性的过于单一，无法体现复杂网络形成的动机和社区结构的丰富内涵。重叠社区发现问题最早由 Palla 等提出，允许一个节点同时属于多个社区[1]。Palla 等提出派系过滤 CPM 算法，通过建立重叠矩阵寻找连通部分形成社区划分[2]。派系过滤算法需要以完全子图为基本单元来发现重叠，这对于很多真实网络尤其是稀疏网络而言，限制条件过于严格，只能发现少量的重叠社团。此后，非重叠社区算法经过调整也应用到重叠社区发现，例如局部优化法[3][4]、标签传播法[5]、概率模型算法[6]、模糊聚类算法[7]、，这些方法往往需要预先给定参数，算法的普适性和鲁棒性受到一定限制。Ahn 等在《Nature》杂志上首次提出将边作为社区划分的研究对象[8]，开创了重叠社区发现的一条新的道路，基于点的非重叠社区算法经过调整可应用到重叠社区发现[9]。复杂网络中的一条边通常对应某一种类型的特定交互，以边为对象使得划分结果更能真实地反映节点在网络中的角色或功能，一条边只归属于一个社区，从而允许一个节点归属于多个重叠社区。总体来说，基于边的社区发现算法与同类型的基于点的算法相比，无论是时间还是空间复杂度都高出很多，主要是由于网络中边的数量要远远多于点的数量，因此复杂度是设计重叠社区发现算法时需要考虑的重要因素。

并行化是降低运算时间的有效途径，目前可用的并行计算框架不断丰富，如 MPI、MapReduce、Spark 和专门用于图计算的 GraphX、Pregel 等。结合并行计算框架的算法开发，可以缓解大规模复杂网络的计算问题。一些学者通过并行化实现非重叠社区发现算法的快速迭代，提高了运行时间方面的性能[10][11]。社区发现并行算法带来效率的提升主要依赖于计算框架的部署规模，其最大难点在于复杂网络数据本身固有的连通性和图计算表现出强耦合性。因此，需要进行有效的图分割，尽可能降低分布式计算的各子图之间的耦合度。本文选择 MapReduce 作为并行计算的开发框架，提出了一种适用于大规模复杂网络的重叠社区发现算法 PHLink (Parallel Hierarchical Link)。

PHLink 算法创新性的提出将节点按照复杂网络的无标度特性进行分层，根据建立连边的不同动机来探索节点的多重属性和社区归属。这种思想也同样适用于其他基于边的社区发现算法，用于降低边计算的复杂度，具有一定的通用性。其次，PHLink 算法在 Hadoop 平台上将复杂网络进行分割和冗余存储，减弱了图计算的强耦合性，使子图得以独立的并行处理，解决了社区发现算法的分布式计算问题。大量真实网络测试表明 PHLink 算法综合性能良好，划分社区质量较高，在并行环境下具有良好的加速性和伸缩性，可以处理千万级连边规模的大规模复杂网络。

1 边社区发现算法及其改进

1.1 边社区发现算法及复杂度分析

传统的社区发现算法是将网络划分为若干个不重叠的点社区，每个节点具有唯一属性且仅能隶属于唯一社区。然而事实上每个节点可以包含多重属性，例如出于不同动机与他人建立连接，亲戚或同事。边社区能更为精确地刻画属性的类别，在应用中更加具有实际意义。

图论提供了一种用抽象的点和边表示各种实际网络的统一方法，是目前研究复杂网络的一种共同语言，因此本文用图的形式对复杂网络的结构进行描述，刻画边社区。

定义1（复杂网络）.复杂网络可抽象为一个由点集V和边集E组成的无向无权图$G(V,\ E)$，其中$v \in V$表示个人或物品等节点信息，$e \in E$表示节点之间的连接关系，点集和边集的规模分别为n和m。

定义2（边相似度）[8].具有一个公共节点v_k的连边对e_{ik}和e_{jk}之间的边相似度是节点v_i和v_j所拥有的共同邻居的相对数量，记为

$$S(e_{ik},\ e_{jk}) = \frac{|n_+(i) \cap n_+(j)|}{|n_+(i) \cup n_+(j)|} \qquad (1)$$

其中$n_+(i)$为节点v_i及其所有邻居节点的集合。

边社区划分可以采用Link单连接凝聚聚类的方法对边集进行聚类[8]，该算法具有直观、易理解的优点，然而其复杂度较高不适用于大规模的复杂网络。Link算法中最重要的步骤为计算连边之间的相

似度得到相似度矩阵，该矩阵代表了任意两条边之间的亲疏关系，复杂度为$O(m^2)$。考虑到连边相似度在计算时仅需考虑有共同节点的两个连边，因此相似度矩阵的复杂度可以降至$O(nK^2)$，其中K是网络中节点度的最大值。

1.2 改进思路

在线社交类的复杂网络的一阶度分布已经被证明具有无标度特性，节点度为k的概率正比于$k^{-\gamma}$，参数γ被称为幂律系数。有研究显示社交网络的幂律系数范围在 1.5<γ<2.5[12]。以 youtube 网络为例[13]，双对数坐标系下概率分布函数呈线性关系，幂律系数γ约为 1.7。这意味着，youtube 网络中仅有极少数的点拥有比较大的度，绝大多数的点只拥有少量连接。比如度超过 200 节点仅占全部节点的 0.2%，这些节点拥有的边占总边数的 21%，而相似度计算量占比却高达 95%。因此本文将利用幂律分布特性来减缓边计算内存存储的压力，降低算法复杂度。

一阶度分布刻画的是网络中不同度的节点各自所占的比例，但具有相同度分布的两个网络可能具有非常不同的性质或行为。二阶度分布显示了度的相关性。Erzsebet等经实际测量研究表明社交网络的层级结构使得集团内部连接紧密，但节点平均度较小，集团间连接稀疏，但负责连接的枢纽节点度较大[14]。枢纽节点之间的连接描述了核心层的连接情况，其社区密度将远远高于网络的划分密度，也即富人俱乐部连通性现象。基于以上结论，本文按照一阶度分布把网络节点分为枢纽节点普通节点，分层进行相似连边聚类。由于枢纽节点往往跨领域连接到其他社区的节点，去除了枢纽节点和其归属边后，由普通节点构成社团的结构将比之前更加清晰，有利于提取社区信息。

2 并行层级连接算法 PHLink

本文基于 Hadoop 平台设计并实现了并行层级连接算法 PHLink（Parallel Hierarchical Link），可以解决复杂网络由于数据量大而导致的计算时间过长，内存不足等问题。PHLink 算法的示意图如图 1 所示，首先对原始网络 *G* 进行预处理，分成三个子图 *G1*、*G2* 和 *G3*，其中 *G1* 包含了普通节点之间的连边，*G2* 包含了普通节点与枢纽节点的连边，*G3* 包含了枢纽节点之间的连边。PHLink 分别对 *G1* 和 *G3* 进行相似度计算，采用凝聚聚类的方法合并相似的边，形成边社区。接下来将 *G2* 中的连边归属到由 *G1* 形成的边社区。最后将边社区转换成点社区，并对点社区进行清洗，去掉节点数少于三个的点社区。至此，PHLink 算法得到了非重叠的边社区划分与重叠的点社区划分结果。

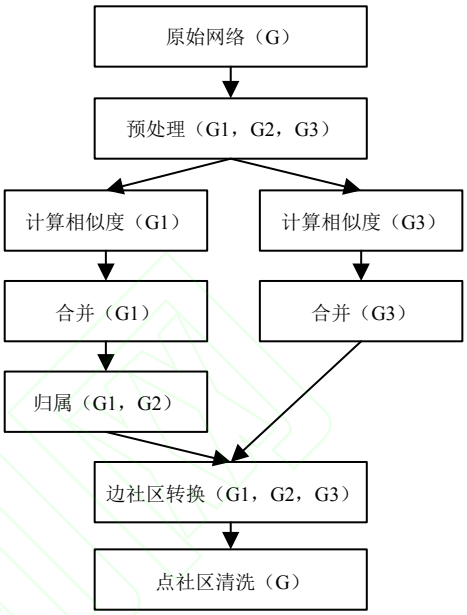

图 1 PHLink 算法示意图

Fig.1 Overview diagram for PHLink

在Hadoop平台上实现PHLink算法的难点在于计算相似度、合并以及归属三个步骤，可以用三个MapReduce作业加以实现，需要重点完成Mapper和Reducer自定义类的设计。接下来将对这三个作业的并行实现算法做详细阐述。

2.1 相似度计算

预处理阶段按照节点度将原始网络G分割为三个子网络$G1$、$G2$和$G3$，并以点邻接表的格式存储在HDFS。

算法 1：相似度计算

Mapper: map()

输入：<*k*， *v* > *k* 为 *node*， v 为 *node* 所有的邻居点 *adj*

θ: 相似度值;

输出：<*k*， *v* > *k* 为 *link*，*v* 为 *link* 邻居边 *neiglink*

1.从 HDFS 读入原始网络 *G*

2. FOR each *v1* in *adj*

3. FOR each *v2* in *adj*

4. *intersection=v1.getAdj()∩v2.getAdj();*

5. *union=v1.getAdj()∪v2.getAdj();*

6. *sim=intersection.size()/union.size();*

7. *link1= node+v1;*

8. *link2= node +v2;*

9. IF *(sim > θ)*
10. *write(key:link1， value:link2);*
11. *write(key:link2， value:link1);*
12. ENDIF
13. ENDFOR
14. ENDFOR

Reducer: reduce()

输入：*<k， v > k* 为 *link*，*v* 为 *link* 邻居边 *neiglink*

输出：*<k， v > k* 为 *link*，*v* 为 *link* 所有的邻居边 *linkAdj*

15. *itr = neiglink.iterator();* //迭代器
16. WHILE *itr.hasNext()* DO
17. *linkAdj = linkAdj ∪itr.next();*
18. ENDWHILE
19. *write(key:link， value: linkAdj);*

相似度计算作业中，map 函数的输入键值对为点邻接表形式，value 值是与 key 节点相邻的所有节点的集合。输出键值对为边邻接表形式，value 值仅包括相似度大于参数θ的邻居边。map 函数按照公式（1）计算包含有共同点 node 的两条相邻边的相似度（4-8 行），如果相似度大于θ，将两条边分别作为键和值输出两次（9-12 行）。reduce 函数对具有相同 key 的边进行规约操作（15-18 行），输出键值对为边邻接表形式。

相似度计算的并行算法时间复杂度为$O(sK_s{}^2)$，其中s为 map 分片中键值对的个数，也即节点数，K_s为该分片中所有节点的度最大值。由于复杂网络的无标度特性，$K_s \ll K$。相似度计算需要从 HDFS 中将原始网络 G 的邻接表读入内存，空间复杂度为$O(m)$，m 为原始网络 G 的边数。

2.2 合并

得到边邻接表之后，可以将每条边及其邻居边初始化为边社区。合并作业的目的是将具有单边连接关系的小社区合并为大社区。本文利用并查集将小社区进行连通，解决了分布式计算中网络耦合的问题。考虑到合并具有层次性，本文采用多次合并的策略平衡计算中的时空复杂度，由 map 任务完成局部边集的合并，reduce 任务对 map 合并的结果进行汇总合并，有且仅有一个 reduce 任务。map 和 reduce 具有相同的功能，两者采用了相同的算法。

map 函数通过扫描边和其所在的社区编号找到社区之间的连通关系。输入键值对为初始边社区，key 为社区编号，value 为社区对应的边集。输出键值对为合并后的边社区，key 为社区编号，value 为社区对应的边集。*linkMap* 存储边的社区编号，如果一条边已经存在 *linkMap* 中，则把已存储的编号和当前的编号组成一对组合 *pair*，标记两个社区是连通的（4-6 行），更新边的社区编号（7 行）。否则，将边与其所属的社区编号组成新键值对存储到 *linkMap*（8-9 行）。遍历初始化社区中的每一条边可以获得所有社区与社区之间的连通关系，存放在 *pairSet*（3-11 行）。cleanup 方法通常在执行完毕 map 后，进行相关变量或资源的收尾工作，仅且执行一次。合并算法的 cleanup 利用并查集将社区进行连通（12-14 行）。*clusterMap* 存储所有的边社区，每条边通过查询并查集找到所归属的社区（17-19 行），如果社区不存在，则新建社区并加入 *clusterMap*（21-23 行）。由此，可以完成将具有单边联系的小社区合并为大社区的效果。

算法 2：合并

Mapper: map()

输入：*<k， v > k* 为社区编号 *id*， *v* 为边社区 *inCluster*

输出：*<k， v > k* 为社区编号 *id*，*v* 为边社区 *outCluster*

1.初始化 *linkMap;*//存放所有边的 map 映射

2.初始化 *pairSet;*//存放社区之间连通关系的 set 集合

3. FOR each *link* in *inCluster*
4. IF (*linkMap.constainsKey(link)*)//边已存在
5. *preId=linkMap.get(link);*
6. *pairSet.add(preId，id)*;//当前 id 与前一个 id 组成 pair
7. *linkMap.put(link，id);*
8. ELSE
9. *linkMap.put(link，id);* //新边加入 linkMap
10. END IF
11.END FOR

Mapper: cleanup()

12. FOR each *pair* in *pariSet*
13. *unionFound.union(pair.v1， pair.v2)*//利用并查集对 pair 进行连通
14. ENDFOR
15.初始化 *clusterMap;* //存放所有社区的 map 映射
16. FOR each *link* in *linkMap*
17. *clusterId = unionFound.find(linkMap.get(link))* //查找 link 应归属的社区
18. IF (*clusterMap.containedKey(clusterId)*) //已存在社区
19. *clusterMap.get(clusterId).add(link);*//加入边
20. ELSEIF
21. *outCluster = null*; //增加新社区
22. *outCluster.add(link);*
23. *clusterMap.put(clusterId， outCluster)*
24. ENDIF
25. ENDFOR

26. *write(key:clusterId， value: outCluster);*

Reducer:reduce()

27. 同 Mapper:*map*()

Reducer:cleanup()

28. 同 Mapper:*cleanup*()

map 函数的时间复杂度为$O(sm_s)$，s为 map 分片中键值对的个数，即输入社区数，m_s为 map 分片中不重复的边数，也即社区的最大规模。cleanup 函数中通过查询并查集，为每条边找到对应的社区编号，时间复杂度为$O(m_s \log s)$。*linkMap* 和 *clusterMap* 存储的是 map 分片中每条边的社区编号，空间复杂度为$O(m_s)$。

合并作业的极端情况是每个 map 只处理一个社区，不进行合并，合并工作全部由唯一的 reduce 完成。此时，待合并的社区数为m_1，社区的最大规模为邻居边的个数$2K_1$，K_1为子图 *G1* 节点度的最大值。故合并作业的并行算法时间复杂度为$O(m_1K_1)$，空间复杂度为$O(m_1)$。

2.3 归属

已知 *G1* 网络的边社区的划分结果 *outCluster*，归属作业把 *G2* 网络的边加入到已知边社区中。map 函数的输入键 key 为枢纽节点 *stone*，值 value 为与 *stone* 相连的所有的普通节点。输出键值对为边社区，key 为社区编号，value 为社区对应的边集。

map 函数从 HDFS 读入边社区划分 *outCluster*，将键值对中包含的每条边 stone-node 归属到可以使 ΔD增加最多的社区（4-16 行）。reduce 函数对具有相同 key 的社区进行规约操作（18-22 行），输出键值对的 key 为社区编号，value 值为边社区。

算法 3：归属

Mapper:map()

输入：<*k*， *v* > *k* 为*枢纽节点 stone*， *v* 为 stone 的邻居点 *adj*

输出：<*k*， *v* > *k* 为 *id*，*v* 为边社区 *cluster*

1.从 HDFS 读取 *outCluster;*

2.*max*=-1;

3.*id*=-1;

4.FOR each *node* in *adj*

5. *link = stone+node;*

6. FOR each *cluster* in *outCluster*//查找增量最大的社区

7. IF (*node in cluster*)

8. *calculate* Δ*D;* //计算增量

9. IF (Δ*D* > *max*)

10. *max*=Δ*D*;

11. *id=clusterId;*

12. END IF

13. END IF

14. END FOR

15.END FOR

16.*clusterMap.get(id).add(link);* //加入增量最大的社区

17.*write(key:id， value: clusterMap.get(id));*

Reducer: reduce()

输入：<*k*， *v* > *k* 为 *id*，*v* 为 *inCluster* 边社区

输出：<*k*， *v* > *k* 为 *id*，*v* 为 *outCluster* 边社区

18. *itr = inCluster.iterator();* //迭代器

19. WHILE *itr.hasNext()* DO

20. *outCluster = outCluster*∪*itr.next();*

21. ENDWHILE

22. *write(key:id， value: outCluster);*

归属并行算法时间复杂度为$O(sq_1)$，其中s是map分片中键值对的个数，q_1是$G1$子图划分的社区数。算法从HDFS中将$G1$子图的社区划分读入内存，空间复杂度为$O(m_1)$，m_1为$G1$子图的边数。

综合相似度计算、合并、归属三个作业的复杂度分析，考虑到 map 分片的大小是人为可控的，子图 *G3* 的规模远远小于子图 *G1*，PHLink 算法的时间复杂度为$O(m_1K_1)$，空间复杂度$O(m)$，其中m_1和m分别为子图 *G1* 和原始网络 *G* 的边集规模，K_1为子图 *G1* 节点度的最大值。可见，PHLink 算法具有较好的扩展性，可以通过增加工作节点，降低运行时间。

3 实验分析

本实验使用的 Hadoop 集群环境由 12 台 Dell E5506 服务器组成，每台服务器拥有 4 核 CPU、12G 内存、500G 硬盘、安装了 hadoop-2.4.1，并以千兆以太网相连。本文中是实验数据来自 SNAP 实验室真实网络数据[13]，其中带星号的三个网络标注有 ground-truth 高质量社区。

表1基准测试数据集

Tab. 1 Benchmark datasets

网络	节点数	边数	描述
football	115	616	足球联盟
jazz	198	2742	爵士合作
facebook	5000	8194	社交网络
dblp*	317080	1049866	引文网络
amazon*	334863	925872	商品网络
youtube*	1134890	2987624	视频网络
skitter	3997962	11095298	网络拓扑

3.1 评价指标

假设真实社区为C^*，由社区发现算法识别出的社区为$\hat{C}$，社区匹配定义为

$$g(i) = arg\,max_j \frac{|C_i \cap \widehat{C_j}|}{|\widehat{C_j}|} \tag{2}$$

准确率定义为

$$P(C^*,\ \hat{C}) = \frac{1}{C^*}\sum_{C_i \in C^*} \frac{|C_i \cap \widehat{C_{g(i)}}|}{|\widehat{C_{g(i)}}|} \tag{3}$$

召回率定义为

$$R(C^*,\ \hat{C}) = \frac{1}{C^*}\sum_{C_i \in C^*} \frac{|C_i \cap \widehat{C_{g(i)}}|}{|C_i|} \tag{4}$$

F1 定义为

$$F1 = \frac{2P(C^*,\ \hat{C})R(C^*,\ \hat{C})}{P(C^*,\ \hat{C})+R(C^*,\ \hat{C})} \tag{5}$$

Omega[6]定义为任意两个节点所归属的社区数量的准确性。

$$\Omega = \frac{1}{n^2}\sum_{u,\ v \in V} 1\{C^*{}_{uv} = \hat{C}_{uv}\} \tag{6}$$

扩展的规范化互信息(ENMI ， Extended Normalized Mutual Information)[6]度量了基准网络的社区集合和挖掘算法发现的社区集合的一致性程度，定义参见[3]。

社区覆盖率[8]定义为属于非平凡社区（3 个或以上节点的社区）的节点所占的比例。

社区重叠率[8]定义为每个节点所属的非平凡社区的平均值。

3.2 枢纽节点比例阈值对计算效率的影响

PHLink 算法计算效率受到枢纽节点比例阈值的影响。比例阈值越高，计算效率越高，综合指标相应降低。比例阈值需根据实际网络的特性和规模选取。

表2比例阈值对计算量的影响

Tab. 2 Threshold of computation proportion

	1%	0.5%	0.1%
facebook	73%	48%	22%
dblp	56%	37%	21%
amazon	46%	35%	20%
youtube	99%	98%	94%
skitter	99%	98%	96%

如表2所见，大规模网络的无标度特性更强，如youtube、skitter网络，极少数的点拥有绝大多数的连边，仅选择度最大的0.1%的节点即可节省94%以上的计算量。随着比例阈值进一步提高，计算量随之减少，但降低幅度变缓。dblp和amazon网络的枢纽节点对计算量的影响较弱，是由于网络本身特性决定的。比如在dblp中，单一学者的论文数量有限，一般最多能到达百篇量级， dblp中节点的最大度为347，因此无标度特性比youtube较弱。

3.3 运行时间及网络规模

本文选取了重叠社区发现的经典算法 CPM[2]、Bigclam[6]、Link[8]与 PHLink 进行比较。实验中 youtube 和 skitter 网络规模较大，采用 12 个节点并行处理，其余网络仅用单个节点进行计算。Bigclam 算法需要提前设置社区数作为输入参数，其中 dblp 和 amazon 按照网络的真实社区数取值，其他数据集由于社区数未知，取值 30。CPM 算法设置完全图参数为 4，PHlink 设置枢纽节点比例阈值为 0.1%。

表3运行时间比较（单位：秒）

Tab. 3 Comparison of computation time（second）

网络	PHLink	Link	CPM	Bigclam
football	0.21	0.24	0.73	0.69 (30)
jazz	16.9	25.5	NA	56.3 (30)
facebook	3.62	3.92	NA	135 (30)
dblp	291	395	NA	5358 (13.5k)
amazon	75.5	85.8	NA	4442 (75k)
youtube	346	NA	NA	NA
skitter	6242	NA	NA	NA

PHLink 无论从运算时间还是网络规模都明显好于其他几种算法。值得注意的 PHLink 处理 jazz 网络的时间要高于 facebook 网络，主要原因是 PHLink 算法的复杂度不仅与网络规模有关，也受到网络的稠密程度（平均度）的影响。

对于 youtube 和 skitter 两个网络，本文通过改变集群规模，考察 PHLink 算法的并行能力。

表4 PHLink算法加速性能（单位：秒）

Tab. 4 Speedup for PHLink（second）

网络	4 节点	8 节点	12 节点
youtube	603	444	346
skitter	17984	9483	6242

由表 4 可知，PHLink 算法有着良好的并行加速性能，而且随着网络规模的增大，加速比接近于 1。

3.4 社区质量评价

评价指标采用 3.1 节中定义的 F1、Omega、ENMI、重叠率和覆盖率，将 5 个指标的取值分别进行归一化，使得每个指标最大值为 1，最小值为 0。五项指标的归一化值加和用于评价算法的综合性能，

其最大取值为 5，最小取值为 0。

PHLink 和 Link 算法采用单连接凝聚聚类的方法对边集进行聚类，将生成大量平凡社区，因此用来测度社区划分准确性的 F1、Omega、ENMI 三个指标将以 5000 个高质量社区作为基准，重叠率和覆盖率计算的是全网中属于非平凡社区的节点。如图 2 所示，PHLink 和 Link 的综合性能相差无几，其中，F1、Omega 两项指标优于 Bigclam 算法。ENMI 在 amazon 数据集下逊于 Bigclam 算法，这与网络数据集的特性有关。amazon 与 dblp 相比，平均社区规模小，节点分布广，社区结构较为松散。PHLink 和 Link 算法的重叠率略高于 Bigclam，但覆盖率低于 Bigclam，主要原因是 Bigclam 算法本身可以避免生成平凡社区。而实际网络 dblp 中，度为 1 的节点比例高达 14%，存在大量的平凡社区。

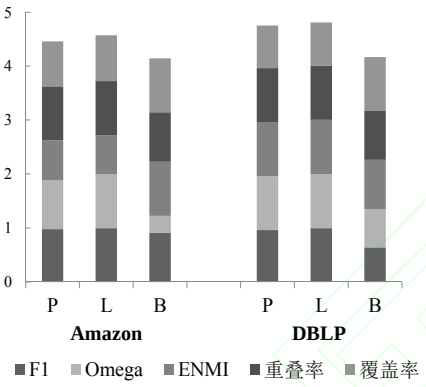

图 2 综合性能比较

Fig.2 Comparison of composite performance

4 结论

本文重点阐述了大规模复杂网络重叠社区发现并行算法 PHLink 的工作原理，基于MapReduce并行计算框架，解决了大规模复杂网络中社区的识别问题。根据复杂网络的无标度特性将节点分为枢纽层和普通层，对不同节点建立连边的原因进行分析和归类，以边作为社区划分的研究对象，用以识别网络中具有重叠性的社区结构。由于节点分层处理，大大降低了计算量，并在此基础上实现并行化，缓解了对内存限制，使子图得以独立的并行处理，解决了传统社区发现算法无法处理的大规模复杂网络社区划分问题。实验在真实大规模复杂网络上进行，与多种经典的重叠社区发现算法进行对比，验证了本文PHLink算法对大规模复杂网络社区识别的时效性，可以处理千万级连边规模的大规模复杂网络。

致谢:中铁第一勘察设计院集团有限公司轨道交通工程信息化国家重点实验室开放课题(SKLK16-04)。

参考文献:

[1] GERGELY P, ALBERT L SZL B, TAM S V. Uncovering the overlapping community structure of complex networks in nature and society[J]. Nature, 2005, 435 (7116): 814-818.

[2] ADAMCSEK B, PALLA G, FARKAS I, I, et al. CFinder: locating cliques and overlapping modules in biological networks[J]. Bioinformatics, 2006, 22 (8): 1021-1023.

[3] LANCICHINETTI A F, SANTO; KERTESZ, JANOS Detecting the overlapping and hierarchical community structure in complex networks[J]. New Journal of Physicals, 2009, 11 (3): 1-19.

[4] 李首庆, 徐洋. 基于自适应聚概率矩阵的 JPDA 算法研究[J]. 西南交通大学学报, 2017, 52 (2): 340-347.
LI Shouqing, XU Yang. Joint probabilistic data association algorithm based on adaptive cluster probability matrix[J]. Journal of Southwest Jiaotong University, 2017, 52(2):340-347.

[5] 刘世超, 朱福喜, 甘琳. 基于标签传播概率的重叠社区发现算法[J]. 计算机学报, 2016, 39 (4): 717-729.
LIU Shichao, ZHU Fuxi, GAN Lin. A label-propagation probability based algorithm for overlappling community detection[J]. Chinese Journal of Computers , 2016, 39 (4): 717-729.

[6] YANG J, LESKOVEC J. Overlapping Communities Explain Core–Periphery Organization of Networks[J]. Proceedings of the IEEE, 2014, 102 (12): 1892-1902.

[7] WANG X, LIU G, LI J, et al. Locating Structural Centers: A Density-Based Clustering Method for Community Detection[J]. Plos One, 2017, 12 (1): e0169355.

[8] YONG-YEOL A, BAGROW J P, SUNE L. Link communities reveal multiscale complexity in networks[J]. Nature, 2010, 466 (7307): 761-764.

[9] SUN H, LIU J, HUANG J, et al. LinkLPA: A Link‐Based Label Propagation Algorithm for Overlapping Community Detection in Networks[J]. Computational Intelligence, 2017, 33 (2): 308-331.

[10] 乔少杰, 郭俊, 韩楠, et al. 大规模复杂网络社区并行发现算法[J]. 计算机学报, 2017, 40 (3): 687-700.
QIAO Shaojie, GUO Jun, HAN Nan et al. Parallel algorithm for discovering communities in large-scale complex networks[J]. Chinese Journal of Computers, 2017, 40 (3): 687-700.

[11] BU Z, WU Z, CAO J, et al. Local Community Mining on Distributed and Dynamic Networks From a Multiagent Perspective[J]. IEEE Transactions on Cybernetics, 2017, 46 (4): 986-999.

[12] CLAUSET A, SHALIZI C R, NEWMAN M E J. Power-Law Distributions in Empirical Data[J]. Siam Review, 2012, 51 (4): 661-703.

[13] YANG J, LESKOVEC J. Defining and Evaluating Network Communities Based on Ground-Truth[J]. Knowledge & Information Systems, 2012, 42 (1): 745-754.

[14] CLAUSET A, NEWMAN M E J, MOORE C. Finding community structure in very large networks. [J]. Physical Review E, 2005, 70 (6 Pt 2): 66-111.