大规模复杂网络下重叠社区的识别

王诗懿,董一鸿,李志超,陈华辉,钱江波
(宁波大学信息科学与工程学院,浙江宁波 315211)

摘 要: 随着网络规模的不断扩大,经典的复杂网络重叠社识别算法已不能高效处理现有的大规模网络图数据.本文在 GraphLab 并行计算模型上提出了基于重要节点扩展的重叠社区识别算法 DOCVN (Detecting the Overlapping Community algorithm based on Vital Node Expanding in GraphLab).算法选取网络中 PageRank 值大的节点作为重要节点,计算其他节点归属于重要节点的节点归属度,并以重要节点为中心形成核心社区及扩展社区,最后根据重要节点间的连接紧密度合并核心社区及扩展社区,并计算出每个节点在所属社区里的节点重要度,实现了大规模网络的重叠社区识别.实验表明该算法与 PD (Propinquity Dynamics)等现有并行算法相比更能有效地识别大规模网络的重叠社区结构.

关键词: 大规模复杂网络; GraphLab; 重叠社区识别; 社会网络; 核心社区
中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112 (2015)08-1575-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2015.08.016

The Identification of Overlapping Communities in Large-Scale Complex Networks

WANG Shi-yi, DONG Yi-hong, LI Zhi-chao, CHEN Hua-hui, QIAN Jiang-bo
(*Department of Information Science and Engineering, Ningbo University, Ningbo, Zhejiang* 315211, *China*)

Abstract: With the unceasing expanding of network scale, many classic detection algorithms of overlapping communities cannot work efficiently in large-scale complex network. Detecting the overlapping community algorithm based on vital node expanding in parallel framework GraphLab (DOCVN) is introduced to identify the overlapping communities. In this algorithm, nodes with high PageRank value are regarded as vital nodes, and then the affiliation degree of other nodes to these vital nodes are computed. After that, kernel communities and expanding communities are identified respectively. Finally, the kernel communities and expanding communities are combined into some overlapping communities by judging whether they connect tightly. And the importance weight of each node in its community is also computed. Experimental results show that the algorithm is more effective than the existing parallel algorithms like PD (Propinquity Dynamics) to identify large-scale overlapping communities.

Key words: large-scale complex network; GraphLab; overlapping community identification; social network; kernel community

1 引言

随着网络规模的扩大,网络中节点数量不断增加,很多交友网站如腾讯、新浪微博等用户人数已达到 10 亿,对于这样大规模的网络,经典的社区识别算法如 Newman 快速算法[1]、改进 Newman 贪婪算法[2]、BOCLP 算法(Balanced Overlapping Community detecting algorithm by Label Propagation)[3]、LMF (Local Maxima of Fitness function)算法[4]等已不能有效地识别社区结构.基于对大规模图数据研究的需要,出现了专门处理大图的计算系统,如基于 BSP(Bulk Synchronous Parallel)模型的 Pregel 和 Giraph 并行计算平台,卡梅隆大学的 GraphLab 并行计算模型等.基于这些大图数据处理平台,出现了并行的社区识别算法,如 Zhang Y 等基于 BSP 模型上提出的 PD (Propinquity Dynamics)算法[5],从玉相等人提出的基于 MapReduce 的标签传播算法[6],J Shi 等人在大规模社会网络下基于 MapReduce 提出的 DEPOLD(DElayed Processing of Large Degree nodes)社区识别算法[7];A Clauset 等人提出了基于 Newman 贪婪算法并行改进的识别算法[8];Riedy 等人[9]提出的基于 Cary-XMT 系统和拥有多核处理器的服务器并行计算模块性最大化值实现社区识别的算法,F.Niu 等人基于 lock-free 机制提出了名为 Hog-

收稿日期:2014-09-25;修回日期:2015-02-25;责任编辑:覃怀银
基金项目:国家自然科学基金(No.61472194);浙江省自然科学基金(No.LY13F020040);宁波市自然科学基金(No.2013A610063,No.2014A610023)

Wild 随机梯度下降算法[10].

上述并行算法,基于 BSP 模型的算法运行时,受限于 BSP 的同步消息传递机制,算法必须完成一个超步中所有计算才能进入下一个超步,增大了运行时间;基于 Hadoop 平台的 MapReduce 在处理大规模图数据时面临无法高效进行频繁数据交换的问题,而基于内存共享并行计算的思想对大规模稠密网络图的社区识别效果并不好.为了克服这些缺陷,本文提出了 GraphLab 云计算平台下基于重要节点扩展的重叠社区识别算法 DOCVN.GraphLab 模型能异步地实现顶点处理,节省较多等待时间.

2 DOCVN 算法描述及算法分析

GraphLab 是近年来兴起的基于内存共享机制的分布式机器学习框架,支持图数据的异步迭代计算,解决了 MapReduce 不适应需要频繁数据交换的迭代机器学习算法,用户只需要考虑算法的实现逻辑,无需关心数据的图模型表示、集群节点之间的通信、一致性和容错性等细节问题.本文提出的 DOCVN 算法在该并行模型上实现.算法实现分为 3 步:(1)重要节点的选择;(2)节点归属度的计算;(3)基于重要节点的社区合并.

2.1 DOCVN 算法描述

2.1.1 重要节点选取

重要节点即网络图中用来扩展识别社区的节点.文中选用能考虑全局拓扑结构特性且易并行的 PageRank 算法[11]作为重要节点的选择策略.改进形式如下:

$$\mathrm{PR}_n(A)=0.15+0.85\times\left(\sum_{i=1}^{m}\frac{\mathrm{PR}_{n-1}(T_i)}{C(T_i)}\right) \quad (1)$$

其中 $\mathrm{PR}_{n-1}(T_i)$表示与节点 A 相连节点 T_i在第 $n-1$ 次迭代中的 PageRank 值,$C(T_i)$是节点 T_i的度.

本文通过节点 PageRank 值的大小判断节点在整个网络图中的重要度,PageRank 值高的节点重要性强,为了限制重要节点的选取数量,文中设置一个阈值 β,当节点的 PageRank 值大于 β 时,记该节点为重要节点,阈值 β 的设置成为重要节点选择的关键.为了社区识别的准确性,提出一个限制参数 α,可根据 α 的变化确定最符合条件的 β 值,β 可表示为:

$$\beta=\alpha\times\mathrm{PR}(A)_{\max},0<\alpha<1 \quad (2)$$

伪代码描述如下:

算法 1 重要节点选择

输入:载入网络图 $G(V,E)$到 GraphLab 模型

输出:$U=\{u_1,u_2,\cdots,u_m\}$//重要节点集合

```
1. Gather: Gather(contest, vertex, eage) //进入 GraphLab 模型的 GAS 三步更新处理阶段
        return PR_{n-1}(T_i)/C(T_i)
2. Apply: Apply(vertex, total) //更新顶点 T_i中的 PageRank 值
        preval = PR_{n-1}(A); newval = total * 0.85 + 0.15;
        if (|preval-newval | < 1E-3)
        perfom_scatter-ture; n.push_back(newval)
3. Scatter; Sactter(context, vertex, eage) //判断是否进入下次更新
   if (perfom_scatter) T = T∪T_i;// T 为下次迭代更新的节点集合
   else return null
步骤 4、5 判定节点是否为重要节点
4. sort(m.begin(), m.end(), form large to small);
        PR(A)_max = max(PR_n(T_m));
5. if (T_i.PageRank > β) U = U∪T(i);
6. fprintf {u_1, u_2, ⋯, u_m}
```

2.1.2 节点归属度计算

定义 1 k 长度可达和 k 长度路径:图 $G(V,E)$中以 v_1为起始节点,经 $v_2,v_3\cdots,v_{k+1}$无重复节点到达 v_{k+1},称节点 v_1到 v_{k+1}节点 k 长度可达,v_1到 v_{k+1}的 k 长度路径 L 记为 $v_1\xrightarrow{k}v_{k+1}$.

定义 2 节点分配度:记节点 u 为网络图 $G(V,E)$中的重要节点,以 u 为起始节点的任一 k 长度路径 L,经过无重复节点 $v_1,v_2,\cdots,v_{k-1}$,k 长度可达终止节点 v_k,则重要节点 u 对节点 v_k的节点分配度为:

$$\xi(u\xrightarrow{k}v_k)=\frac{1}{d(u)d(v_1)d(v_2)\cdots d(v_{k-1})} \quad (3)$$

定义 3 节点的 k 长度归属值:网络图 $G=(V,E)$中路径 L 为 u 到 v 的 k 长度路径,其中 u 为重要节点,n_i为 u 到 v 的 k 长度路径数量,则节点 v 归属到重要节点 u 的 k 长度归属值用 $p(v\xrightarrow{k}u)$表示:

$$p(v\xrightarrow{k}u)=\sum_{j=1}^{n_i}\xi_j(u\xrightarrow{k}v) \quad (4)$$

其中,$\xi_j(u\xrightarrow{k}v)$指 v 到 u 的第 j 条路径上重要节点 u 对节点 v 的节点分配度.

定义 4 节点归属度:网络图 $G(V,E)$中 u 为重要节点,v 为任意节点,v 到 u 的最大路径长度为 $\max l$,则 v 归属于 u 的节点归属度为该节点所有 k 长度归属值之和,用 $P(v\rightarrow u)$表示:

$$P(v\rightarrow u)=\sum_{k=1}^{\max l}p(v\xrightarrow{k}u) \quad (5)$$

节点归属度值越大表明节点与该重要节点的关系越近;归属度值越小说明节点与该重要节点的关系越远.文中设置一个节点归属度阈值 λ,当节点归属度值小于阈值 λ 时,表明该节点相对于这个重要节点的作用非常小,将其过滤.节点归属度计算阶段更新函数的伪代码描述如下:

算法 2　节点归属度计算

输入:图 G' // 经过算法 1 更新后的图 G

输出:$\{v_{id},\{<u_{id},P(v_{id}\rightarrow u_{id})>\}$// 每个顶点所归属的重要节点及归属度值信息

1. Gather://计算节点到重要节点的每条路径节点分配度 ξ 和节点 k 长度归属值 P

   int i,k;

   While($k<\max l$) //$\max l$ 为两点之间的最大路径长度

   do{for($i=1;i<=k;i++$)

   $$\xi(u\xrightarrow{i}v_k)=\frac{1}{d(u)d(v_1)d(v_2)\cdots d(v_{k-1})};$$

   $$p(v_k\xrightarrow{i}u)=p(v_k\xrightarrow{i}u)+\xi(u\xrightarrow{i}v_k)$$

   $k++;$}

2. Apply:// Apply 阶段更新计算节点 v_k到重要节点 u 的节点归属度 P

   $$P_j(v_k\rightarrow u)=P_j(v_k\rightarrow u)+\sum_{k=1}^{\max l}p(v_k\xrightarrow{k}u)$$

   if ($|P_j-P_{j-1}|<$1E-3) perfom_scatter = true

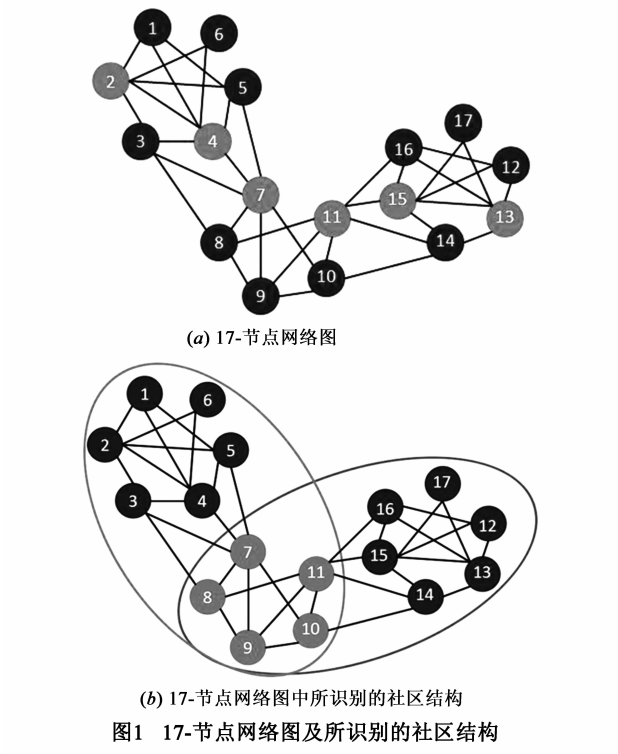
3. Scatter:// 判断是否进入下一次迭代计算

   if (perfom_scatter) $T=T\cup v_k$;// T 为下次迭代更新的节点集合

   else return null

4. if($P(v_k\rightarrow u)>\lambda$ fprintf$\{v_k,\{<u,P(v_k\rightarrow u)>\}\}$

图 1(a)为一个简单的 17-节点网络图,设式(2)中的 α 值为 0.75,利用式(1)可计算出该网络图的重要节点(灰色标记).根据算法 2 可计算出节点 v 到相应重要节点的节点归属度值,表 1 为图 1(a)中 17 个节点所输出的节点归属度信息($\lambda=0.1$).



(a) 17-节点网络图

(b) 17-节点网络图中所识别的社区结构

图1　17-节点网络图及所识别的社区结构

表 1　17-节点网络图节点归属度信息($\lambda=0.1$)

节点 id v_i	节点 v_i相应的重要节点归属度信息 $<u_j,P(v_i\rightarrow u_j)>$			
v_1	$<v_2,0.341536>$	$<v_4,0.303928>$	$<v_7,0.143157>$	
v_2	$<v_2,1>$	$<v_4,0.441298>$	$<v_7,0.220848>$	
v_3	$<v_2,0.31237>$	$<v_4,0.297765>$	$<v_7,0.293981>$	
v_4	$<v_4,1>$	$<v_2,0.529557>$	$<v_7,0.325911>$	
v_5	$<v_2,0.384042>$	$<v_4,0.353928>$	$<v_7,0.253971>$	
v_6	$<v_2,0.271441>$	$<v_4,0.238108>$		
v_7	$<v_7,1>$	$<v_4,0.325911>$	$<v_2,0.265017>$	$<v_{11},0.20893>$
v_8	$<v_7,0.299436>$	$<v_{11},0.250651>$	$<v_4,0.143007>$	$<v_2,0.131983>$
v_9	$<v_7,0.287797>$	$<v_{11},0.287797>$	$<v_4,0.105558>$	
v_{10}	$<v_{11},0.299436>$	$<v_7,0.250651>$	$<v_{15},0.143007>$	$<v_{13},0.131983>$
v_{11}	$<v_{11},1>$	$<v_{15},0.325911>$	$<v_{13},0.265017>$	$<v_7,0.20893>$
v_{12}	$<v_{13},0.341536>$	$<v_{15},0.303928>$	$<v_{11},0.143157>$	
v_{13}	$<v_{13},1>$	$<v_{15},0.441298>$	$<v_{11},0.220848>$	
v_{14}	$<v_{13},0.31237>$	$<v_{15},0.297765>$	$<v_{11},0.293981>$	
v_{15}	$<v_{15},1>$	$<v_{13},0.529557>$	$<v_{11},0.325911>$	
v_{16}	$<v_{13},0.384042>$	$<v_{15},0.353928>$	$<v_{11},0.253971>$	
v_{17}	$<v_{13},0.271441>$	$<v_{15},0.238108>$		

节点归属度计算结束后,根据节点到相应重要节点的归属度值可实现重要节点的扩展识别,从而以重要节点为核心形成相应的核心社区与扩展社区.

定义 5　核心社区:设 u 为网络图 $G(V,E)$中的一

个重要节点,图 G 中相对于重要节点 u 的归属度值最大的所有节点集合组成了以重要节点 u 为核心的核心社区.图 G 中每个重要节点的核心社区不相交.

定义 6 扩展社区:设 u 为网络图 $G(V,E)$中的一个重要节点,除去重要节点 u 核心社区内的顶点,所有与节点 u 存在归属关系的节点集合组成了重要节点 u 的次核心层,称为重要节点 u 的扩展社区.

表 2 显示了图 1(a)中 17-节点网络图的所有重要节点形成的核心社区以及扩展社区集合.

表 2 17-节点网络图所有重要节点形成的核心社区和扩展社区

重要节点 id v_j	核心社区	扩展社区
v_2	$\{v_1,v_2,v_3,v_5,v_6\}$	$\{v_4,v_7,v_8\}$
v_4	$\{v_4\}$	$\{v_1,v_2,v_3,v_5,v_6,v_7,v_8,v_9\}$
v_7	$\{v_7,v_8,v_9\}$	$\{v_1,v_2,v_3,v_4,v_5,v_{10},v_{11}\}$
v_{11}	$\{v_{10},v_{11}\}$	$\{v_7,v_8,v_9,v_{12},v_{13},v_{14},v_{15},v_{16}\}$
v_{13}	$\{v_{12},v_{13},v_{14},v_{16},v_{17}\}$	$\{v_{10},v_{11},v_{15}\}$
v_{15}	$\{v_{15}\}$	$\{v_{11},v_{12},v_{13},v_{14},v_{16},v_{17},v_{10}\}$

2.1.3 社区的合并策略

基于重要节点形成核心社区和扩展社区后,需要对社区进一步合并处理.本文采用的社区合并策略是将关系紧密的重要节点所形成的核心社区及扩展社区进行合并,从而减少冗余.

定义 7 重要节点比率:每个社区包含一个或多个重要节点,社区内的节点对这些重要节点都存在一个相应的节点归属度值,重要节点比率 $\eta(u_j\in C_i)$就是社区内的某一重要节点在该社区内的全部重要节点中所占的比重,其计算公式表示如下:

$$\eta(u_j\in C_i)=\frac{\sum P(v_m\rightarrow u_j)}{\sum_j\sum P(v_m\rightarrow u_j)} \tag{6}$$

其中,v_m是社区 C_i中与重要节点 u_j存在归属度值的节点,且 $v_m\neq u_j$.

定义 8 节点重要度:若节点 v_x属于社区 C_i,节点 v_x在社区 C_i里的重要程度 $Q(v_x\in C_i)$称节点重要度:

$$Q(v_x\in C_i)=\sum(P(v_x\rightarrow u_j)\times\eta(u_j\in C_i)) \tag{7}$$

其中 $P(v_x\rightarrow u_j)$为 C_i里的节点 v_x到重要节点 u_j的节点归属度.

社区合并阶段的伪代码描述如下:

算法 3 社区合并

输入: $U,A,C=\varnothing$// U 为重要节点关系数据集,A 为重要节点的核心社区及扩展社区集合

输出: $C=\{C_i,\{[k,Q(k\rightarrow C_i)]\}\}$ // 输出合并后社区,每个社区内的节点表示为[节点 k,k 在 Ci 的节点重要度]

1. While ($U\neq\varnothing$)
   do {if($u_j\in U_{ui}$&&max($P(u_i\rightarrow u_x)$) = $P(u_i\rightarrow u_j)$)
   $C_{(ui,uj)}=A_{ui}\cup A_{uj}$; $U=U-\{u_i\}$} //第一次合并后的社区 C
2. itvt = C.begin();
   While(itvt! = C.end() && i! = u&& i! = w &&j! = u &&j! = w) do
   {if ($i=u\parallel i=w\parallel j=u\parallel j=w$) $C_{(x,y)}=C_{(i,j)}\cup C_{(u,w)}$//$x$,$y$ 为 i,j,u,w 中不相等的两个值
   itvt + +;} //i,j,u,w 为社区 $C_{(i,j)}$,$C_{(u,w)}$的下标.
3. for($i=1;i<n;i++$)
   $C_i=C_{(x,y)}$; $Q(k\rightarrow C_i)=Q(k\rightarrow C_i)+\sum P(k\rightarrow u_j)\times\eta(u_j\in C_i)$;
4. fprintf $C=\{C_i,\{[k,Q(k\rightarrow C_i)]\}\}$

表 3 17-节点网络图中 C_1社区里节点的节点重要度

节点 id	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
节点重要度 Q	0.234	0.472	0.272	0.525	0.282	0.144	0.487	0.201	0.155	0.116	0.206

表 4 17-节点网络图中 C_2社区里节点的节点重要度

节点 id	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}
节点重要度 Q	0.206	0.116	0.155	0.201	0.487	0.224	0.472	0.272	0.525	0.282	0.144

图 1(b)是 17-节点网络图经社区合并处理后所得的社区结构,由图 1(b)可知,该网络图有两个社区结构,其中节点 v_7,v_8,v_9,v_{10},v_{11}为两社区的重叠节点.表 3 和表 4 给出了 17-节点网络图两社区中节点的节点重要度.

社区里相应节点的重要度值直接反映了节点在社区里的重要性,对于重叠节点,在它相应的社区子集中都有相应的节点重要度值 Q.由表 3 和表 4 可以看出重叠节点 v_7,v_8在社区 C_1里的重要度较高,重叠节点 v_{10},v_{11}在社区 C_2里的重要度较高,而节点 v_9在两个社区里重要度相等,还可以看出重要节点在社区中的重要度明显高于一般节点.

2.2 DOCVN 算法分析

2.2.1 节点归属度计算的优化分析

节点归属度的计算需要得到两个节点间的所有路径,这是一个 NP 难问题,特别对于一个拥有百万节点的大规模网络图而言将是一个灾难.这里提出了近似计算的策略进行优化.

式(5)得到了节点 v 到重要节点 u 的 k 长度归属值

$p(v \xrightarrow{k} u)$,记为 p_k,从 1 长度归属值到 max$l$ 长度归属值得到数列:$p_1, p_2, \cdots, p_k, \cdots, p_{\max l}$.令 $S_k = p_1 + p_2 + \cdots + p_{k-1} + p_k$,$(1 \leqslant k \leqslant \max l)$,那么式(5)的节点 v 到 u 的归属度 $P(v \rightarrow u) = S_{\max l}$.令 $\psi_k = \frac{p_{k+1}}{p_k}$,表示数列 p_k的后项与前项之比;令 $\varepsilon_k = \frac{p_{k+1}}{S_k}$表示数列后项与前 k 项之和的比值,其中,$1 \leqslant k \leqslant \max l - 1$.

图 2(a)显示了 GraphLab 的 4 集群并行模型上合成网络图随路径长度 k 变化时 ψ_k和ε_k的变化,$k=7$ 时,$\psi_7 = 0.07004$;$k>7$ 时,ψ_k的值继续减小.也就是说,当 $k \geqslant 7$ 时,数列 p_k的后项与前项之比小于 1/10.ε_k的值随 k 值的增加总体呈指数下降,k 值越大下降越快.当 $k=7$ 时,$\varepsilon_7 = 2.6526 \times 10^{-5}$.

图 2(b)显示的是节点归属度计算时平均运行时间随 k 的变化,其运行时间随 k 的变化急剧增大.对于 10,000 节点的网络图而言,$k=7$ 时,运行时间为 10.85s,$k=8$ 时,运行时间达到 75.96s,其路径长度仅加 1,而运行时间却增大了 7 倍.

实验表明,当 $k \geqslant 7$ 时,$\psi_k < 1/10$,$\varepsilon_k < 3 \times 10^{-5}$,衰减速度很快,因此,如果取 $k=7$,以 S_7作为节点归属度的近似值,无需计算出所有路径的归属值,可以大大加快运行速度,而误差则在 0.003% 的范围内.误差估算如下:

令 $S_{>7} = p_8 + p_9 + \cdots + p_{\max l}$,则 $S_{\max l} = (p_1 + p_2 + \cdots + p_7) + (p_8 + \cdots + p_{\max l}) = S_7 + S_{>7}$

$\because \psi_k = \frac{p_{k+1}}{p_k}$,由实验得到 $0.1 > \psi_7 > \psi_8 > \psi_9 > \psi_{10}$ ……

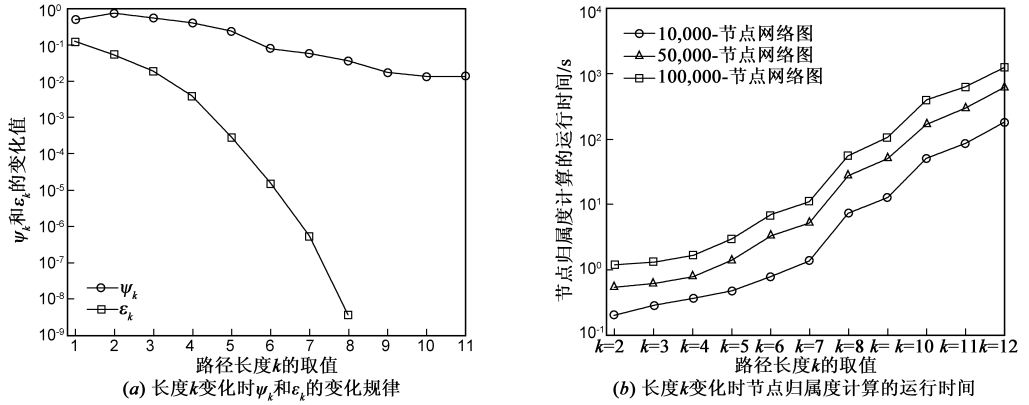
$\therefore p_9 = \psi_8 p_8 < 0.1 p_8$, $p_{10} = \psi_9 p_9 < 0.01 p_8$, $p_{11} = \psi_{10} p_{10} < 0.001 p_8$, ……

$\therefore S_{>7} = p_8 + p_9 + p_{10} + \cdots\cdots < (1 + 0.1 + 0.01 + 0.001 + \cdots) \times p_8 = 1.1111\cdots \times p_8$

$\because \frac{p_8}{S_7} = \varepsilon_7 = 2.6526 \times 10^{-5}$

$\therefore$ 误差 $\bar{\omega} = \frac{S_{>7}}{S_{\max l}} = \frac{S_{>7}}{S_7 + S_{>7}} < \frac{S_{>7}}{S_7} < 1.1111\cdots \times \frac{p_8}{S_7} \approx 2.9473 \times 10^{-5} < 3 \times 10^{-5}$

结合实验中 ψ_k值,ε_k值和运行时间随 k 值变化的规律以及 S_7与 $S_{\max l}$的误差值,取 $k=7$ 即式(5)中最大路径长度 max$l$ 为 7 时,节点归属度计算的误差控制在 0.003%,运行时间也相对较少,大大提高算法整体的效率,优化节点归属度的计算.



(a) 长度k变化时ψ_k和ε_k的变化规律　　(b) 长度k变化时节点归属度计算的运行时间

图2　节点归属度计算的优化分析图

2.2.2　算法时间复杂度分析

(1)重要节点选择,其时间复杂度与算法迭代次数 i 及节点的度有关,表示为 $O(i * |V| + i * |E|)$,即 $O(i * |E|)$;

(2)节点的归属度计算,其时间复杂度与归属度值存储表的长度 l(l 受阈值 λ 控制,一般小于 7,即 $l<7$)及节点所有邻接点的存储表迭代更新次数 i 有关,表示为 $O(i * |E| * l + i * |V| * (l + \log(l))$,即 $O(i * |E| * l)$.因此算法在 GraphLab 上的时间复杂度为 $O(i * |E|)$,该时间复杂度为线性的,易于并行;

(3)对于社区合并,其时间复杂度由重要节点个数 N_v(远小于顶点总数 $|V|$)以及第一次合并后所形成社区的社区个数 M_T决定,其中 M_T的值不大于 $N_v/2$,则该阶段时间复杂度为 $O(|N_v|^2/2)$.

综上,DOCVN 算法总的时间复杂度为 $O(i * |E| + |N_v|^2/2)$.

3　实验分析和验证

3.1　数据集

实验采用两类数据集,一类是人工合成网络图,由 LFR(Lancichinetti Fortunato Radicchi)基准程序[12]随机生成网络图,该基准程序合成网络图的同时生成与之相对应的社区结构,可用来检验社区识别算法的准确性;另一类是真实网络图.从 Stanford Network Analysis Project

(SNAP)[13]上得到.表 5 和表 6 是均为实验所用数据集信息.

表 5 人工网络图(混合参数 μ 和重叠节点数 o_n 均是 LFR 算法中的参数,可随意设定)

数据集	节点数($\|V\|$)	μ	o_n	边数$\|E\|$	节点平均度($2*\|E\|/\|V\|$)
synthNet-10w	100,000	0.4	10000	1,984,900	39.70
synthNet-50w	500,000	0.4	10000	9,831,289	39.33
synthNet-100w	1,000,000	0.4	10000	19,641,382	39.28

表 6 真实网络图

数据集	节点数($\|V\|$)	边数$\|E\|$	节点平均度	数据集描述
web – BerkStan	685,320	7,600,595	22.18	Berkeley 和 Stanford 的网络图
web-NotreDame	325,729	1,497,134	9.19	Notre Dame 的网络图
web-Stanford	281,903	2,312,497	16.41	Stanford.edu 的网络图
soc-LiveJournall	4,847,571	68,993,773	28.47	LiveJournal 在线社交网络
soc-Pokec	1,632,803	30,622,564	37.51	Pokec 在线社交网络
Wiki-Talk	2,394,385	5,021,410	4.19	Wikipedia talk (交际) 网络

实验基于 GraphLab 计算模型实现,16 台机器作为实验集群,每台机器的配置相同:Intel(R) Core(TM) i5 – 3470 4 核 CPU(3.2GHz),内存 8GB,集群中设置 1 台为 Master 机,其余为 Mirror 机.通过实验发现,取 $\alpha=0.75$, $\lambda=0.1$ 时社区识别的效率最高.根据前面的优化分析,节点归属度的计算取路径长度 $\max l=7$.

实验以 BOCLP 算法[3]和 PD 算法[5]以及 DEPOLD 算法[7]三种对重叠社区识别性能较好的算法作为对比算法.BOCLP 算法是 C++ 语言实现的单机算法;PD 算法是基于 Giraph 并行图处理模型实现,DEPOLD 算法是基于 MapReduce 的并行算法,在 Hadoop 模型下运行,集群个数均设定为 16,在下面的实验中,三个对比算法的参数均选取它们识别效率最高时的值.

3.2 社区识别质量分析

3.2.1 人工合成网络图

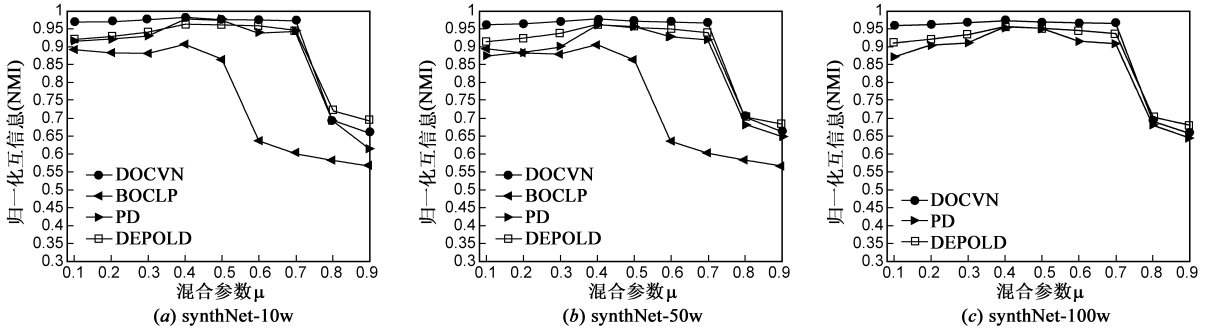
定义 9 归一化互信息 NMI[14]:NMI(Normalized Mutual Information)往往用来比较两个实验结果变量变化时相互之间的关系,是数据挖掘的聚类算法以及社区结构识别算法中常用的验证算法效率的比照参数.

$$\mathrm{NMI}=\frac{I(X,Y)}{\sqrt{H(X)H(Y)}} \tag{8}$$

其中,$I(X,Y)$是 X 和 Y 之间的互信息,$H(X)$,$H(Y)$分别是 X,Y 的熵.NMI 值用来反映社区识别算法所识别的社区结构与真正社区结构间的接近程度,如果 NMI($0\leqslant \mathrm{NMI}\leqslant 1$)的值越大,则说明社区结构越符合实际情况.

作为社区识别质量的评判标准,NMI 通常与 LFR 基准程序里混合参数 μ 相结合,根据 μ 变化时 NMI 值的变化来评定社区质量的好坏.

图 3 显示了不同识别算法在不同规模网络图上随混合参数 μ 值变化时对应 NMI 值的变化规律.图 3(a)和图 3(b)是四种算法在网络图 synthNet-10w 和 synthNet-50w 中 NMI 值的变化曲线.当 $\mu=0.4$ 时,DOCVN 算法的 NMI 值最大,几乎接近 0.98,与 BOCLP,DEPOLD 以及 PD 算法的最大 NMI 值相比,DOCVN 算法的社区识别效率相对较好.且参数 μ 在 0 ~ 0.8 之间变化时,DOCVN 算法的 NMI 值几乎没有波动,而另三种算法的波动相对明显,由此可以说 DOCVN 算法有很好的稳定性,图 3(c)是 PD,DOCVN 以及 DEPOLD 三种算法在 synthNet-100w 合成图中的 NMI 值比较,可以看出 DOCVN 算法的 NMI 值基本稳定在 0.97 ~ 0.981 之间,对大规模网络图的识别性能优于 PD 及 DEPOLD 算法.该实验表明了 DOCVN 算法在大规模网络上的高效性和稳定性.



(*a*) synthNet-10w (*b*) synthNet-50w (*c*) synthNet-100w

图3 三种算法的NMI值随混合参数变化的变化图

3.2.2 真实网络图

与人工合成网络图不同,真实的网络图没有原始的社区结构信息作比对,引入社区质量参数—聚类系数对真实网络图识别效率进行验证.

定义 10　聚集系数(Clustering Coefficient,CC)[15]:往往用于描述网络的聚集度,是一个非常重要的参数,网络的社区结构和搜索性能等问题均与其有密切关系.

$$c(i)=\frac{2n}{k(k-1)} \tag{9}$$

其中,k 为节点 i 的邻居节点数,n 为节点 i 的邻居节点之间实际的连接边数.所识别社区结构质量的好坏可用社区结构内所求的节点平均聚集系数的大小评判,平均聚集系数较高说明所识别的社区质量较好.文献[15]中指出若一个社区内所有节点的平均社区 CC 值大于或等于这些节点的平均全网 CC 值的 3 倍,就说明所识别出的社区结构有意义,也证明了所用社区识别算法的识别质量很高.

图 4 显示了 DOCVN 算法在表 6 的 6 个真实数据集上部分社区内节点的平均社区 CC 和平均全网 CC 的结果对比.图中节点在社区内平均社区 CC 的值明显高于节点对全网络图的平均全网社区 CC,由于 6 个真实数据集网络图结构的稀疏稠密不同,所得的聚集系数值的大小有所不同,但总的来说这些社区内节点的平均社区 CC 值基本上都高出节点平均全网 CC 值的 3 倍,即从节点聚集系数的角度证明了 DOCVN 算法的社区质量很高,在大规模真实数据集上是可行的,有很好的社区识别效率.

3.3　集群大小对算法运行时间和社区质量的影响

图 5 是 6 个真实数据集随集群个数变化时算法运行时间及所识别社区内部分节点的平均社区值的实验变化结果.由图 5(a)和图 5(b)可以看出,DOCVN 算法的运行时间的随着运行集群个数的增大而减少,而所识别社区内节点的平均社区 CC 值不变,说明算法随着集群数量的增大不影响算法的识别质量,图 5 的这个实验结果符合 GraphLab 计算模型的设计理念,即通过并行计算处理网络图数据来降低运行时间但又不影响算法的效率.由图 5 还可以看出,6 个不同数据集所减少的运行时间与集群个数的增加并不直接成反比,因为在集群个数增加的同时也增加了部署节点到集群的时间,增加额外开销,一定程度上影响了算法的运行时间.因此,集群的数量并不是越多越好,当网络图中节点规模小于百万时,Graphlab 的集群数量设置在 16 以内即可高效运行.

图 6 是集群个数为 16 时各算法运行时间的比较.DOCVN 算法识别社区的运行时间明显快于 PD 和 DEPOLD 算法.因为 DEPOLD 算法基于 Hadoop 平台的 MapReduce 框架,在 MapReduce 下处理数据时要进行频繁的数据交换,大大增加了算法的运行时间;PD 算法在同步消息传递机制的 BSP 模型上运行,只有超步中所有的节点都处理完毕才会进入下一迭代,增加了运行成本,而 DOCVN 算法在 GraphLab 并行计算模型上异步迭代运行,每个顶点独立计算不需同步等待即可进入下一次迭代,且算法还对节点归属度的计算进行了优化处理,能很好地控制运行时间.

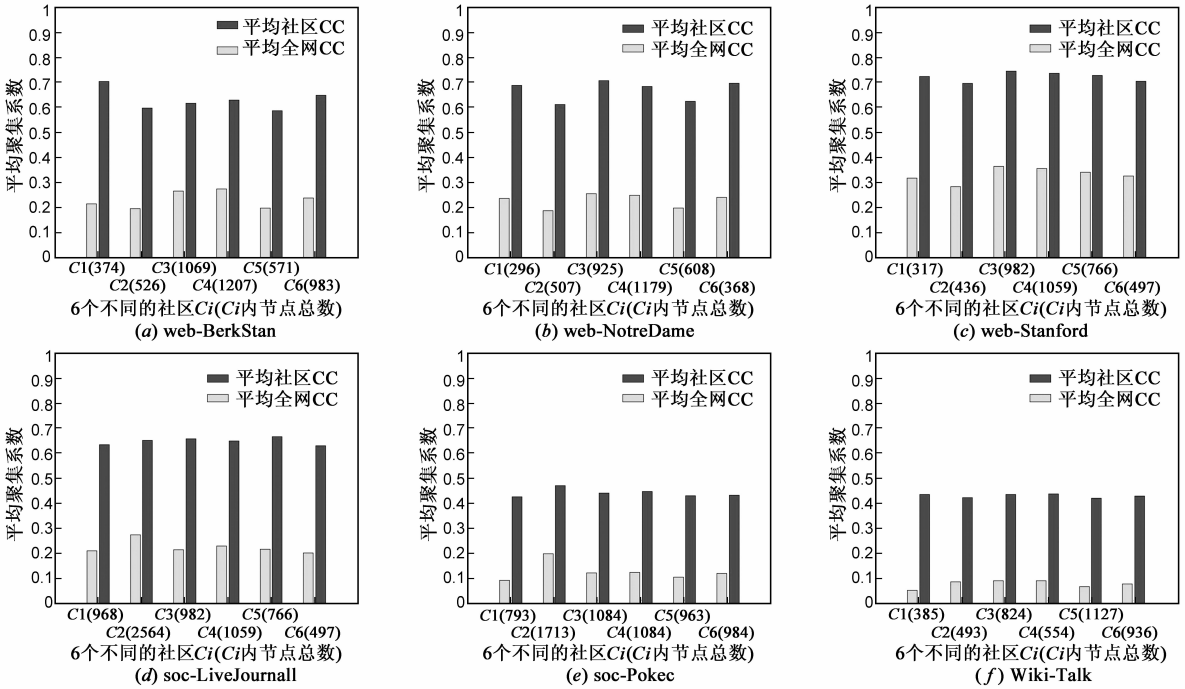
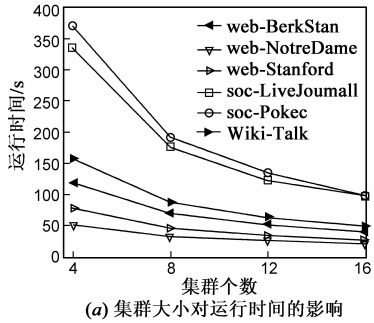
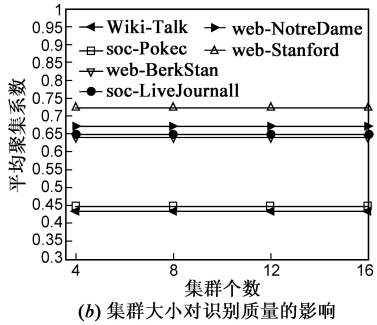


图4　DOCVN算法在不同真实网络图中的社区质量分析图

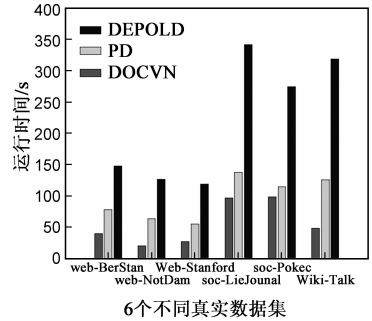


(*a*) 集群大小对运行时间的影响



(*b*) 集群大小对识别质量的影响

图5 集群大小对运行时间和社区识别质量的影响



6个不同真实数据集

图6 集群数相同时不同算法运行时间比较

4 结论

社区识别在复杂网络分析中扮演着重要角色,本文提出的 DOCVN 算法充分利用了 Graphlab 异步并行化和分布式计算框架的特点,解决了单机算法不能处理的网络规模问题,实现了对大规模复杂网络的重叠社区识别.实验证明,当网络节点规模在百万以上时,本文算法在不降低所识别社区准确性的前提下表现了出更好的运行效率,并且通过实验可发现该算法识别效率优于基于 MapReduce 框架和基于 BSP 模型的社区识别算法.

参考文献

[1] Newman M E J, Girvan M. Finding and evaluating community structure in networks[J]. Physical Review E, 2004, 69: 026113.

[2] Clauset A. Finding local community structure in networks[J]. Physical Review E, 2005, 72: 026132.

[3] 王庚.基于标签传播的稳定重叠社区挖掘算法研究[D].山东:山东建筑大学,2013.
Wang G. Research to Stable Detecting Overlapping Communities by Label Propagation on Social Networks[D]. Shandong: Shandong Jianzhu University, 2013. (in Chinese)

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

[5] Zhang Y, Wang J, Wang Y, et al. Parallel community detection on large networks with Propinquity dynamics[A]. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining[C]. Pairs: ACM, 2009. 997 – 1006.

[6] 从玉相.基于 MapReduce 的社区挖掘算法[D].上海:上海交通大学,2013.
Y X Cong. Community Detection Based on MapReduce[D]. Shanghai: Shanghai Jiao Tong University, 2013. (in Chinese)

[7] J Shi, W Xue, W Wang, Y Zhang, et al. Scalable community detection in massive social networks using MapReduce[J]. IBM J RES & DEV, 2013, 57(3): 3 – 14.

[8] A Clauset, Newman M E J, et al. Finding community structure in very large networks[J]. Physical Review E, 2004, 70: 066111.

[9] J Riedy, D A Bader, H Meyerhenke. Scalable multi-threaded community detection in social networks[A]. Proceedings of IEEE International Conference on Tools with Artificial Intelligence[C]. Seoul, Korea: IEEE, 2012. 1619 – 1628.

[10] F Niu, B Recht, S J Wright, et al. Hogwild !: A Lock-free Appoach to Parallelizing Stochastic Gradient Descent[DB/OL]. Available: http://books.nips.cc/papers/files/nips24/NIPS2011_0485.pdf, 2011.

[11] Bryan K, Leise T. The $ 25 000 000 000 eigenvector: the linear algebra behind Google[J]. SIAM Rev, 2006, 48(3): 569 – 581.

[12] Lancichinetti A, Fortunato S. Limits of modularity maximization in community detection[J]. Physical Review E, 2011, 84: 066122.

[13] Jure L. Stanford large network dataset collection[OL]. http://snap.stanford.edu/data/index.html, 2013.

[14] L Danon, A Diaz-Guilera, J Duch, A Arenas. Comparing community structure identification[J]. Journal of Statistical Mechanics: Theory and Experiment, 2005, 09: p09008.

[15] Newman M E J. The structure and function of complex networks[J]. SIAM Review, 2003, 45(2): 247 – 256.

作者简介

王诗懿 女,1988 年生于河南周口.现为宁波大学信息科学与工程学院硕士研究生.主要研究方向为大数据、数据挖掘.
E-mail: zwb_wsy@126.com

董一鸿(通信作者) 男,博士,1969 年生于浙江宁波.现为宁波大学教授、硕士生导师.主要研究方向为大数据、数据挖掘和人工智能. E-mail: dongyihong@nbu.edu.cn