大规模复杂网络社区并行发现算法

乔少杰[1] 郭 俊[2] 韩 楠[3] 张小松[4] 元昌安[5] 唐常杰[6]

[1](成都信息工程大学信息安全工程学院 成都 610225)

[2](西南交通大学信息科学与技术学院 成都 610031)

[3](成都信息工程大学管理学院 成都 610103)

[4](电子科技大学大数据研究中心 成都 611731)

[5](广西师范学院科学计算与智能信息处理广西高校重点实验室 南宁 530023)

[6](四川大学计算机学院 成都 610065)

摘 要 随着网络规模的不断扩大,传统社区发现算法已无法有效和高效地处理大规模网络数据.基于 Spark 分布式图计算模型,提出大规模复杂网络社区并行发现算法 DBCS(Discovering Big Community on Spark).算法利用基于模块度的聚类思想,首先计算出节点对之间的模块度增量,然后迭代查找出所有模块度增量最大的节点对,对所有节点对进行合并操作,并更新节点对之间的模块度增量,进而实现大规模复杂网络社区识别.大量真实复杂网络与仿真网络数据集上的实验结果表明:DBCS 算法能有效地解决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,百万级以上节点处理时间约为 4min,是 Hadoop 平台下并行发现算法运行时间的 1/20,社区识别准确率比传统社区发现算法提高了 7.4%.

关键词 复杂网络;社区发现;图计算;并行计算;模块度;社交网络

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2017.00687

Parallel Algorithm for Discovering Communities in Large-Scale Complex Networks

QIAO Shao-Jie[1] GUO Jun[2] HAN Nan[3] ZHANG Xiao-Song[4]

YUAN Chang-An[5] TANG Chang-Jie[6]

[1](*College of Information Security Engineering, Chengdu University of Information Technology, Chengdu* 610225)

[2](*School of Information Science and Technology, Southwest Jiaotong University, Chengdu* 610031)

[3](*School of Management, Chengdu University of Information Technology, Chengdu* 610103)

[4](*Big Data Research Center, University of Electronic Science and Technology of China, Chengdu* 611731)

[5](*Science Computing and Intelligent Information Processing of Guangxi Higher Education Key Laboratory, Guangxi Teachers Education University, Nanning* 530023)

[6](*College of Computer Science, Sichuan University, Chengdu* 610065)

Abstract As the size of networks grows larger, traditional community discovery algorithms cannot effectively and efficiently process the large-scale network data. Based on the Spark distributed graph computing model, this study proposes a parallel algorithm for discovering communities in large-scale complex networks, called DBCS (Discovering Big Community on Spark). The proposed approach employs the basic idea of clustering method beyond modularity, which first calculates the increment of the modularity between the node pairs, and then iteratively finds the

收稿日期:2015-05-24;在线出版日期:2015-11-06.本课题得到国家自然科学基金(61100045,61165013)、高等学校博士学科点专项科研基金(20110184120008)、教育部人文社会科学研究规划基金(15YJAZH058)、教育部人文社会科学研究青年基金(14YJCZH046)、四川省教育厅资助科研项目(14ZB0458)和科学计算与智能信息处理广西高校重点实验室开放课题(GXSCIIP201407)资助.**乔少杰**,男,1981 年生,博士,教授,中国计算机学会(CCF)高级会员,主要研究方向为大数据、社交网络、移动对象数据库.E-mail:sjqiao@cuit.edu.cn.**郭 俊**,男,1991 年生,硕士,主要研究方向为复杂网络社区发现.**韩 楠**(通信作者),女,1984 年生,博士,讲师,主要研究方向为复杂网络.E-mail:hannan@cuit.edu.cn.**张小松**,男,1968 年生,博士,教授,主要研究领域为复杂网络.**元昌安**,男,1964 年生,博士,教授,主要研究领域为数据挖掘.**唐常杰**,男,1946 年生,硕士,教授,博士生导师,主要研究领域为数据库.

maximum modularity increment among all the node pairs. Lastly, it merges the node pairs, and updates the modularity increment of the remaining nodes, in order to identify the communities in large-scale complex networks. Extensive experiments are conducted on several real and synthetic network datasets and the results demonstrate that DBCS can effectively deal with the problem of partitioning the large-scale networks that does not make sense for traditional algorithms. In particular, it only takes about four minutes to handle more than one million nodes for community discovery. In addition, the time cost is reduced to 1/20 of the parallel algorithm based on Hadoop. The accuracy is improved by 7.4% when compared to traditional community discovery algorithms.

Keywords complex networks; community discovery; graph computing; parallel computing; modularity; social networks

1 引 言

现实生活中网络无处不在,如人与人交往形成的人际关系网、论文引用形成的关系网、客户购买商品形成的交易网等,这些网络因其节点关系都具有很高的复杂性,被称为"复杂网络"[1].复杂网络通常呈现出社区的结构特性,所谓社区结构,通常指由一组内部连接紧密,与外部连接稀疏的节点集合构成的结构[2].社区结构作为复杂网络重要的特性,对认识节点内部关联、信息传播、兴趣点推荐等具有重要的研究价值.

大数据时代网络用户数量呈爆炸性增长,网络节点数量不断增加,诸多社交网络,如腾讯、新浪微博等用户数已超过 10 亿,Facebook 每月活跃人数超过 13 亿.社会网络规模的增长不仅体现为节点数量增多,还表现为节点之间关系复杂,节点数目及节点之间关联关系变化频繁.挖掘大规模复杂网络中的社区结构,对成员之间关联进行分析,可以发现用户的行为规律,进而为精准营销、广告投放、舆论控制等提供辅助决策支持.因此,大规模复杂网络社区发现在网络结构的理论研究和网络分析实际应用中具有重要意义.

然而,对于大规模的复杂网络,节点数目动辄上百万,节点之间的关联错综复杂,传统社区识别算法已不能有效地挖掘社区结构.因此提出新型高效、智能、准确的大规模社区识别算法成为亟需解决的问题.为了提高大规模复杂网络社区发现的准确性和效率,本文主要贡献包括:(1)结合图论、网络性质和模块度思想,提出新的节点模块度更新方法和社区合并方法,应用于社区并行发现算法中,极大地提高了社区识别的时间性能;(2)在 Spark 云计算平台下,构建模块度增量矩阵,查找具有最大模块度增量的节点对,应用新的节点合并方法,并以模块度增量是否全为负值为依据,判定是否完成社区的发现;(3)分别在大规模真实和仿真复杂网络数据上进行实验,通过与串行和并行社区发现算法进行对比实验,验证本文所提算法的性能优势.

2 相关工作

Newman[2-3]提出模块度最优算法以来,基于模块度的社区挖掘算法一直占据网络社区挖掘方法的主流,并衍生出了许多不同的改进算法[4],如经典的快速 Newman 社区划分方法[3]、CNM 算法[5].近年来,国内的研究者如潘磊等人[6]基于网络中的局部信息实现了边社区挖掘算法.冷作福[7]利用贪婪优化技术提出了新型网络社区挖掘算法,Zhang 等人[8]对快速 Newman 算法进一步改进,重新设计了模块度的求解方法,改进快速 Newman 算法的更新策略,较好地提高了社区识别的效率.黄伟平[9]提出了用户吸引力的概念,用于准确的社区发现.熊正理[10]提出基于用户紧密度的社区发现算法,将用户紧密度同层次聚类算法结合,实现了社区结构的挖掘.国外研究者,如 Blondel 等人[11]改进了模块度增量求解方法,通过新的计算公式,迭代合并社区,取得了良好效果.Parsa 等人[12]基于单变量边缘分布算法,使用概率向量模型,将进化算法与社区发现相结合,实现了社区发现.Oliveira 等人[13]利用改进的 Kuramoto 耦合振子同步模型,从动力学因素分析网络,实现了复杂网络中的社区发现.

近年来,研究人员相继提出了基于大数据平台

的社区并行识别算法. Clauset[14] 提出了基于 CNM 算法的社区并行识别方法，基本思想是并行计算使社区模块性最大化的值，通过减小通信开销实现对大规模网络的社区识别. 但当网络规模不断增大，数据量攀升到一定规模，算法无法运行. 李金鹏[15] 提出的基于 Hadoop 平台的链接社区识别方法，解决了当网络规模不断扩大，链接社区方法无法存储和处理大矩阵的问题. 但算法执行效率较低，当网络节点规模达到 1.5 万个时，处理时间达到 5000 s 以上. Riedy 等人[16] 利用具有多核处理器的服务器并行计算模块性最大化值，进而识别社区. 所提方法具有强烈的硬件依赖性，不具有通用性，可移植性较差. Moon 等人[17] 通过计算边介数改进了 Girvan-Newman 算法，使用近似优化技术加速了社区发现过程. Staudt 和 Meyerhenke[18] 基于内存共享思想，实现了并行的标签传播算法，并对算法做出了一定改进，实现了海量网络上的社区发现. Chen 等人[19] 将稀疏相似矩阵、Lanczos/Arnoldi 矩阵分解法应用于各种谱聚类算法，并在 MPI 和 MapReduce 上实现了并行化. Papadimitriou 和 Sun[20] 实现了 MapReduce 上的协同聚类算法，用以处理 PB 级挖掘任务.

Spark 是一种新型可扩展的分布式数据分析平台[21]，已经被应用于大规模数据的挖掘和知识发现. 应用 Spark 进行并行挖掘基于如下优点：(1) Spark 提供了一组丰富的操作处理数据，并且可以将数据全部缓存在内存当中，极大地提高了 Spark 的运算效率；(2) Spark 支持任何变量的广播，使每个节点均保存一份变量的值，克服了 Hadoop 变量传递只支持简单数据类型变量的限制.

为了克服传统社区发现方法的不足，本文提出了一种新的基于 Spark 的大规模社区并行发现算法，有效地解决了大规模复杂网络的社区识别问题，相比传统的社区识别算法和同类并行算法，在正确性和效率上得到了较大的提升. Spark 的优势在于其为一个弹性分布式系统，灵活度极高，当前的大多数社区发现方法均可实现在该平台上，如谱聚类算法、边介数法、标签传播法等. 但是，主流社区发现算法，如谱聚类算法进行大规模网络数据社区发现面临如下问题：(1) 对于大规模复杂网络，大矩阵存储对于传统算法，如谱聚类算法，变得十分困难，且复杂网络为稀疏网络，矩阵中大部分数据为空. Spark 可以过滤这些无用信息，存储大矩阵；(2) 对于传统社区划分算法，如何分解较大规模的相似度矩阵生成特征值和特征向量也变得困难；(3) k-means 等算法需要不断迭代，如何快速搜寻各个节点的相似度也成为一个障碍，而 Spark 存储数据没有顺序要求.

3　大规模复杂网络社区发现算法

本文提出的大规模复杂网络社区并行发现算法根据最大模块度增量选择节点和社区进行合并，实现了大规模网络中社区的并行发现，极大地提高了时间效率.

3.1　基本概念

本节对算法中使用的主要概念进行形式化定义，如下所示.

定义 1(复杂网络).　$CN=(V,E)$，$V=\{\sum v_i \mid i=1,2,n\}$ 表示网络中的节点集合，$E=\{\sum e_i \mid i=1,2,m\}$ 表示网络中边集合. 复杂网络不仅表现为节点和边的数量巨大，而且表现为节点的多样性，可以是任意的实体；边的多样性，各个边代表不同的关系. 其中，$v=(o,pr)$ 表示节点，o 表示网络中的实体，pr 表示实体的属性；$e=(p,q,r)$ 表示网络中一条边，p 和 q 表示两个不同实体，r 表示实体间的关系.

定义 2(大社区).　社区是网络中节点的集合，社区内部节点连接紧密，社区之间节点连接较为稀疏[3]. 随着网络规模增大，社区也不断增大(Velocity)，社区内节点数目不断增多(Volume)，社区内部和社区之间的关系也越来越复杂，大社区不断吞并小社区，小社区不断消亡，社区变得更加广阔(Variety)，此类社区结构称为大社区.

定义 3(节点度)[8].　已知一个节点数为 n，边数为 m 的网络 $G(V,E)$ 中，V 表示节点集合，E 表示边集合，如果节点 u 与节点 v 中间有边相连，则 $e_{uv}=1$，否则 $e_{uv}=0$. 节点 u 的度 k_u 表示为

$$k_u=\sum_{v\in V} e_{uv} \tag{1}$$

表示与节点 u 相连的边的数目.

如图 1 所示，左右两边表示两个不同的社区，节点 3 的度 $k_3=5$.

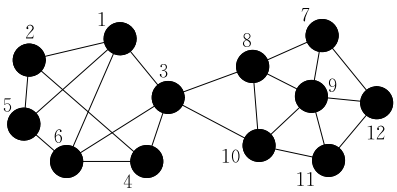


图 1　简单网络社区结构

定义 4(模块度)[2-3]. 网络的模块度 Q 定义为

$$Q=\sum_{i=1}^{k}(e_{ii}-a_i^2) \tag{2}$$

$$Q=\frac{1}{2n}\sum_{c\in C}\sum_{u,v\in V}\delta_{cu}\delta_{cv}\left(\boldsymbol{A}_{uv}-\frac{k_u k_v}{2n}\right) \tag{3}$$

式(2)和(3)等价,式(2)中,e_{ii} 表示在一个社区 i 内部,连接其中两个顶点的边数占网络总边数的比例;a_i 表示与社团 i 中节点相连,但另一个节点不属于社团 i 的边数占网络总边数的比例;k 表示社区数目. 在式(3)中,$\boldsymbol{A}$ 表示网络的邻接矩阵;k_u 和 k_v 分别表示节点 u,v 的度;C 表示所有社团构成的集合;n 表示网络的总边数. 如果节点 u 在社区 c 中,则 $\delta_{cu}=1$,否则 $\delta_{cu}=0$.

文献[2]中研究指出:当一个网络的模块度 Q 值达到最大时,表明社区划分达到了最好的效果. 但是,直接判断 Q 是否达到最大值相当困难. 因此,本文采用 Newman 提出的模块度增量 $\Delta\boldsymbol{Q}$ 的概念,其意义是:假如合并社区 i 和社区 j,造成的模块度 Q 增大或减小的幅度,其定义如下[3]:

$$\Delta\boldsymbol{Q}=2(e_{ij}-a_i\times a_j) \tag{4}$$

$$a_i=\frac{k_i}{2m} \tag{5}$$

e_{ij} 表示社区 i 与社区 j 之间的连接边占总边数的比例的一半,其中 $i\neq j$. 当 $\Delta\boldsymbol{Q}>0$ 时,Q 值递增;当 $\Delta\boldsymbol{Q}<0$ 时,Q 值达到最大,社区划分工作结束.

定义 5(社区发现准确率). 社区发现准确率定义为正确识别社区中节点的个数与网络节点总数的比率,用 DA(Detection Accuracy)定义如下:

$$DA=\frac{\sum_{i=1}^{k}\max\{C_i\cap C'_j\,|\,C'_j\in C'\}}{n},\ j=1,2,\cdots,l \tag{6}$$

其中:C 表示原始的精确的社区集合,$C=\{C_1,C_2,\cdots,C_k\}$;C' 表示利用社区发现算法识别出的社区集合,$C'=\{C'_1,C'_2,\cdots,C'_l\}$;$\max\{C_i\cap C'_j\,|\,C'_j\in C'\}$ 表示取所有结果社区集与第 i 个精确社区 C_i 公共节点的数目的最大值;n 表示网络节点数. DA 值越大反映社区识别算法发现社区质量越好.

社区发现准确率是对查全率和查准率[22]两个信息检索重要指标的结合,可信度较高,因此本文利用定义 5 给出的社区识别准确率评估不同算法的社区识别准确性.

3.2 算法思想

DBCS 算法基本思想是:首先,根据模块度增量概念初始化网络的图结构;然后,对网络进行搜索,借助图论理论,查找最大模块度增量,选取多个社区进行合并;最后根据网络的结构特性,更新整个网络,进行下一轮的迭代发现. 算法中应用的定理、性质和推论如下所示.

定理 1. 鸽子洞原理. 又称抽屉原理,将 $n+1$ 个元素放入 n 个集合,其中必定有两个元素处于同一个集合中.

定理 2. 在任何无向图中,必定存在两个或两个以上的节点的度完全相同.

证明. 已知 G 是具有 n 个节点的简单图 $n\geq 2$,且没有孤立节点.

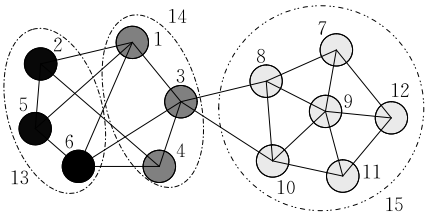
因为网络无孤立节点;

所以每个节点的度 $1\leq k\leq n-1$;

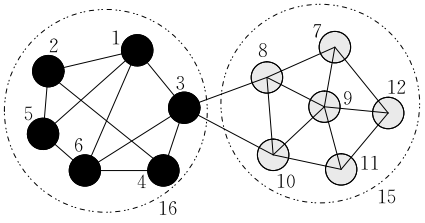
所以根据定理 1,n 个节点在 $n-1$ 个整数中选择一个数字,因此至少有两个节点选取同一个整数值作为其节点的度.

根据定理 2,在复杂网络中,必定存在大量结构相似的节点,因此,网络中也会存在大量具有相同模块度增量的节点对,将这些节点对同时合并,即为算法中新的社区合并方法,相比于传统基于模块度的社区发现算法每一轮只合并两个社区,提高了算法处理效率.

性质 1. 在一个复杂网络中,如果节点数和边数不变,社区合并形成的新社区,其社区内节点之间的边数是两个被合并社区内边数以及这两个被合并社区之间的边数的总和,新社区与其他社区之间的连接边数等于被合并社区与这个社区之间的连接边数之和.



(a) 社区合并前



(b) 社区合并后

图 2 合并社区 13 和 14 后的结果

例 1. 以图 2(a)所示的 12 个节点的简单网络为例,已知 13 和 14 表示两个不同社区,合并社区 13 和社区 14 形成新社区 16,合并之后结果如图 2

(b)所示.其中,E_i表示社区 i 的内部边集合,E_{i-j}表示社区 i 与社区 j 之间的连接边集合.边集合变化如下:

合并前:

$E_{13}=\{e_{2-5},e_{5-6}\}$, $E_{14}=\{e_{1-3},e_{3-4}\}$, $E_{14-15}=\{e_{3-8},e_{3-10}\}$, $E_{13-14}=\{e_{1-2},e_{1-5},e_{1-6},e_{2-4},e_{3-6},e_{4-6}\}$.

合并后:

$E_{16}=E_{13}\cup E_{14}\cup E_{13-14}=\{e_{1-2},e_{1-3},e_{1-5},e_{1-6},e_{2-4},e_{2-5},e_{3-4},e_{3-6},e_{4-6},e_{5-6}\}$; $E_{15-16}=E_{13-15}\cup E_{14-15}=\{e_{3-8},e_{3-10}\}$.

根据性质 1,可以得出如下推论.

推论 1.　对于一个网络来说,在节点数和边数保持不变的情况下,多个已知社区合并形成的新社区,与其他社区之间的模块度增量为:如果其他社区同多个已知社区相连,则为其模块度之和;如果不相连,则减去对应的节点度占总边数的比例的乘积,更新如式(7)所示:

$$\Delta\boldsymbol{Q}[z][k]=\begin{cases}\Delta\boldsymbol{Q}[z][k]+\Delta\boldsymbol{Q}[i][k]; & \langle i,k\rangle\in E,\ i\in C_z\\ \Delta\boldsymbol{Q}[z][k]-2\times a[i]a[k]; & \langle i,k\rangle\notin E,\ i\in C_z\end{cases}\tag{7}$$

$$a_z=\sum_{i\in C_z}a_i\tag{8}$$

其中:C_z表示合并之后的新社区,序号为 z,E 是边集合;k 是不属于 C_z的旧社区;i 是合并到社区 C_z中的旧社区,$\langle i,k\rangle$表示从社区 i 到社区 k 的边集合,a_i表示旧社区的度比例.

推论 1 即为算法中使用的新的模块度更新方法,而传统的更新方法如 fast newman 算法是重新计算每个社区之间的模块度增量.

例 2.　如图 1 所示的网络结构,初始时,认为每个节点代表一个社区,利用式(4)计算出各个社区之间模块度增量 $\Delta\boldsymbol{Q}$,构成如表 1(a)所示的矩阵,其中第 1 行和第 1 列均表示社区序号.因为仅关注需要进行合并的社区,社区内部的变化无需考虑,于是,对角线处可以直接初始化为 0.从矩阵的值来看,此时可以合并的社区有〈2,4〉、〈2,5〉、〈7,12〉、〈11,12〉,$\Delta\boldsymbol{Q}$ 值均为 0.036,根据定理 2,可以对这 4 组社区进行合并.以合并社区 2 和社区 4 形成社区 13 为例,合并后结果如表 1(b)所示.

由表 1(a)、(b)可以看出,社区 13 与其他社区之间的模块度增量为社区 2 和社区 4 与对应社区模块度增量的和,图中加粗字体标识出模块度增量和

表 1　社区合并 $\Delta\boldsymbol{Q}$ 矩阵的求取

(a) 图 1 网络合并前的 $\Delta\boldsymbol{Q}$ 矩阵

	1	**2**	3	**4**	5	6	7	8	9	10	11	12
1	0.000	**0.033**	0.025	**−0.012**	0.033	0.029	−0.012	−0.017	−0.021	−0.017	−0.012	−0.012
2	**0.033**	**0.000**	**−0.015**	**0.036**	**0.036**	**−0.012**	**−0.009**	**−0.012**	**−0.015**	**−0.012**	**−0.009**	**−0.009**
3	0.025	**−0.015**	0.000	**0.030**	−0.015	0.025	−0.015	0.025	−0.026	0.025	−0.015	−0.015
4	**−0.012**	**0.036**	**0.030**	**0.000**	**−0.009**	**0.033**	**−0.009**	**−0.012**	**−0.015**	**−0.012**	**−0.009**	**−0.009**
5	0.033	**0.036**	−0.015	**−0.009**	0.000	0.033	−0.009	−0.012	−0.015	−0.012	−0.009	−0.009
6	0.029	**−0.012**	0.025	**0.033**	0.033	0.000	−0.012	−0.017	−0.021	−0.017	−0.012	−0.012
7	−0.012	**−0.009**	−0.015	**−0.009**	−0.009	−0.012	0.000	0.033	0.030	−0.012	−0.009	0.036
8	−0.017	**−0.012**	0.025	**−0.012**	−0.012	−0.017	0.033	0.000	0.025	0.029	−0.012	−0.012
9	−0.021	**−0.015**	−0.026	**−0.015**	−0.015	−0.021	0.030	0.025	0.000	0.025	0.030	0.030
10	−0.017	**−0.012**	0.025	**−0.012**	−0.012	−0.017	−0.012	0.029	0.025	0.000	0.033	−0.012
11	−0.012	**−0.009**	−0.015	**−0.009**	−0.009	−0.012	−0.009	−0.012	0.030	0.033	0.000	0.036
12	−0.012	**−0.009**	−0.015	**−0.009**	−0.009	−0.012	0.036	−0.012	0.030	−0.012	0.036	0.000

(b) 图 1 网络合并社区 2,4 后的 $\Delta\boldsymbol{Q}$ 矩阵

	1	3	5	6	7	8	9	10	11	12	**13**
1	0.000	0.025	0.033	0.029	−0.012	−0.017	−0.021	−0.017	−0.012	−0.012	**0.021**
3	0.025	0.000	−0.015	0.025	−0.015	0.025	−0.026	0.025	−0.015	−0.015	**0.015**
5	0.033	−0.015	0.000	0.033	−0.009	−0.012	−0.015	−0.012	−0.009	−0.009	**0.027**
6	0.029	0.025	0.033	0.000	−0.012	−0.017	−0.021	−0.017	−0.012	−0.012	**0.021**
7	−0.012	−0.015	−0.009	−0.012	0.000	0.033	0.030	−0.012	−0.009	0.036	**−0.019**
8	−0.017	0.025	−0.012	−0.017	0.033	0.000	0.025	0.029	−0.012	−0.012	**−0.025**
9	−0.021	−0.026	−0.015	−0.021	0.030	0.025	0.000	0.025	0.030	0.030	**−0.031**
10	−0.017	0.025	−0.012	−0.017	−0.012	0.029	0.025	0.000	0.033	−0.012	**−0.025**
11	−0.012	−0.015	−0.009	−0.012	−0.009	−0.012	0.030	0.033	0.000	0.036	**−0.019**
12	−0.012	−0.015	−0.009	−0.012	0.036	−0.012	0.030	−0.012	0.036	0.000	**−0.019**
13	**0.021**	**0.015**	**0.027**	**0.021**	**−0.019**	**−0.025**	**−0.031**	**−0.025**	**−0.019**	**−0.019**	**0.000**

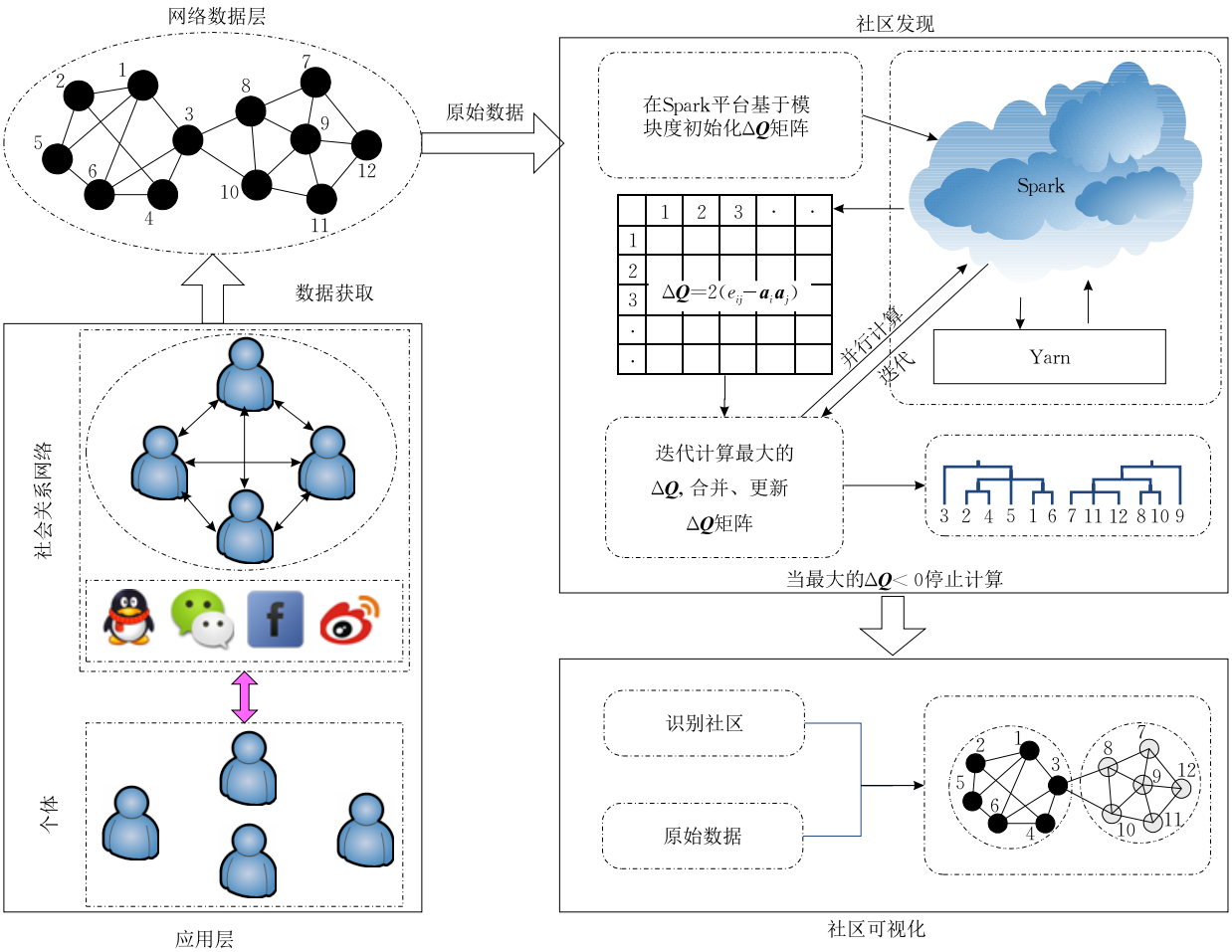


图 3 DBCS 算法工作原理

的求取过程. 以表 1(b)边〈1,13〉(第 1 行第 13)的模块度增量为 0.021 为例,其为表 1(a)中的社区 2 和社区 4 对应的边〈1,2〉的模块度增量 0.033 和〈1,4〉的模块度增量−0.012 之和.

基于上述分析,本文提出了 DBCS 算法,实现了社区发现在 Spark 平台上的并行算法,工作原理如图 3 所示,具体步骤包括:首先,对从复杂网络中获取的大规模数据进行预处理,以满足算法下一步处理的需要;然后,根据模块度增量公式(4),初始化整个网络,将每个节点视为一个社区,计算每个社区之间的模块度增量;再者,不断迭代计算发现新社区,根据定理 2 找出所有具有最大模块度增量的节点对进行合并,并利用式(7)、(8)更新所有社区之间的模块度增量,当最大模块度增量为负时,社区识别过程结束;最后,结合最终的社区发现结果,将网络社区结构可视化. 由于 GraphX 提供对边和节点属性的支持,DBCS 将顶点集 V 存储为(vId,cId)形式,其中 vId 表示节点序号,cId 表示社区序号;将边集 E 存储为(s,d,ΔQ),其中 s 为边的源节点,d 为边的目的节点,ΔQ 为这条边对应的模块度增量. 同时,Spark 提供了丰富的数据操作方法,DBCS 算法调用 Spark 平台的 mapEdges,mapVertices 等方法,并结合上述算法思想,完成并行社区发现工作.

3.3 社区并行发现算法

社区并行发现算法步骤为:(1) 数据预处理;(2) 参数初始化;(3) 构建模块度增量矩阵;(4) 查找最大模块度增量;(5) 社区合并与模块度更新;(6) 生成社区划分结果. 数据预处理利用 Spark 平台的 GraphX 将文本数据加载到系统中,GraphX 会自动地识别出顶点,并将顶点集合和边集合缓存于内存中,供下一步处理. 算法中其他关键步骤内容描述如下.

3.3.1 参数初始化

算法 1. DBCS 参数初始化.

输入:经过预处理的网络数据 D

输出:含 ΔQ 的图 G

1. $G \leftarrow graphLoader(D)$;
2. $n \leftarrow getVertice(G)$;
3. $m \leftarrow getEdge(G)$;

4. 将边数目 m 广播到集群上所有计算机节点；
5. FOR (each node i in D)
6. $k_i \leftarrow$getDegree(G)；
7. $a_i \leftarrow k_i/2m$；
8. END FOR
9. FOR (each e in E)
10. $\Delta \boldsymbol{Q}_{ij} \leftarrow 2(e_{ij}-a_i \times a_j)$；
11. END FOR

初始化阶段负责计算出算法必需的参数，主要包含网络节点数 n 和边数 m，计算出各个节点的度并求出向量 $\boldsymbol{a}$，并求出每个节点对之间的模块度增量 $\Delta \boldsymbol{Q}$，具体过程如算法 1 所示，主要步骤：

(1) 利用 GraphX 提供的接口加载复杂网络数据(第 1 行)，同时计算网络的节点数目 n(第 2 行)，边数 m(第 3 行)，将 m 广播到所有节点中(第 4 行)；

(2) 求出每个节点的出度(第 5～6 行)，利用式(5)计算向量 $\boldsymbol{a}$(第 7 行)；

(3) 利用式(4)求出每个节点对之间的模块度增量(第 9～11 行)，构成一个新的包含节点度增量的图 G.

3.3.2　查找最大模块度增量

完成模块度增量计算工作后，进入迭代社区发现的过程，根据定理 2，同时寻找出多个具有最大模块度增量的社区对. 在社区合并前，需将查找出的节点进行处理，为它们分配新的社区序号，如果发现这些社区对之间具有关联性，需要修改社区序号，给出它们合并之后的新社区标号.

例 3.　已知如图 1 所示的网络，根据表 1(a)的 $\Delta \boldsymbol{Q}$ 矩阵可知，当前可以合并社区〈2，4〉，〈2，5〉，〈7，12〉，〈11，12〉，显然，社区 2，4，5 和社区 7，11，12 应分别同时合并为社区 13 和社区 14.

算法 2.　查找最大模块度增量和需合并的社区.

输入：含 $\Delta \boldsymbol{Q}$ 的图 $G(E,V)$

输出：最大模块度增量 MQ，需合并的社区集合 $C=\{C_1,C_2,\cdots,C_l\}$

1. $MQ \leftarrow searchMaxDeltaQ(G)$；
2. 将 MQ 广播到所有集群集群节点；
3. $T \leftarrow E \times V$；
4. FOR (each triplet t in T)
5. IF ($getAttr(t)==MQ$)
6. $MC \leftarrow insert(t)$；
7. END IF
8. END FOR
9. FOR (each triplet t in MC)
10. $(i,j) \leftarrow getAttr(t)$；
11. IF (i or $j \in C$) THEN
12. $k \leftarrow$ 从 C 中获取社区 i 或社区 j 的新编号；
13. $C_k \leftarrow insert(i,j)$；
14. ELSE
15. $n \leftarrow n+1$；
16. $C_n \leftarrow insert(i,j)$；
17. END IF
18. END FOR

算法 2 的主要操作包括：

(1) 比较图 G 中各个边 e 的 $\Delta \boldsymbol{Q}$ 值，根据定理 2 找出所有最大值 MQ(第 1 行)，并将最大模块增量 MQ 广播到集群所有节点中(第 2 行)；

(2) 求得边集合 E 和顶点集合 V 的笛卡尔积 T，其结构为 $t=(s,sc,d,dc,\Delta \boldsymbol{Q})$，$s$ 为源节点序号，d 为目的节点编号，sc 和 dc 分别是源节点和目的节点的社区序号(第 3 行)；

(3) 查找集合 T 中 $\Delta \boldsymbol{Q}$ 等于 MQ 的元素子集合 MC(第 4～8 行)；

(4) 将需要合并的社区进行整理，首先获得边 t 中源节点属性 sc 和目的节点属性 dc，这两个值分别代表当前要合并的社区(第 10 行)，如果 i 或 j 已经属于 C 中某个新社区，则获取这个新社区将 i 和 j 合并到其中(第 11～13 行)，否则需要将 i 和 j 形成另一个新社区，新社区序号为 $n+1$(第 15～16 行)，最终输出合并后社区 C.

3.3.3　社区合并与更新

社区的合并与更新是整个算法的核心，根据推论 1，在 Map 中将需要合并的社区的编号 i 更新为其对应的新社区序号，在 Reduce 中将所有一端具有相同新社区序号而另一端社区序号不同的边的 $\Delta \boldsymbol{Q}$ 值相加，然后将需要合并的社区与向量 $\boldsymbol{a}$ 的对应值相加构成新社区的向量值.

算法 3.　社区合并与更新.

输入：要合并的社区集合 C，向量 $\boldsymbol{a}$

输出：更新后的 G，更新后的向量 $\boldsymbol{a}$

1. $T \leftarrow V \times E$；
2. FOR (each triplet t in E)
3. IF ($getAttr(t) \in C$) THEN
4. $X,Y \leftarrow$ 查找 $t.sc$ 和 $t.dc$ 对应新社区序号及新社区内需合并的社区序号集合；
5. FOR (each node i in X and each node j in Y)
6. IF (i 和 j 有边相连) THEN
7. $\Delta \boldsymbol{Q}_{XY} \leftarrow \Delta \boldsymbol{Q}_{XY}+\Delta \boldsymbol{Q}_{ij}$；
8. ELSE
9. $\Delta \boldsymbol{Q}_{XY} \leftarrow \Delta \boldsymbol{Q}_{XY}-2 \times \boldsymbol{a}_i \times \boldsymbol{a}_j$；
10. END IF
11. END FOR

12. END IF
13. END FOR
14. FOR (each node k in $\boldsymbol{a}$)
15. IF (k 对应的社区为要合并的社区) THEN
16. $\boldsymbol{a}_c \leftarrow \boldsymbol{a}_c + k$;
17. END IF
18. END FOR

社区合并与更新如算法 3 所示,主要步骤为:

(1)首先求得顶点集合 V 与边集合 E 的笛卡尔积 T(第 1 行);然后,查找边 $t=(s,sc,d,dc,\Delta\boldsymbol{Q})$ 中 sc 和 dc 对应的新社区序号以及这两个社区中包含的本轮需合并的社区序号集合 X,Y(第 4 行);

(2)根据式(7)对于 X 中社区 i 与 Y 中社区 j,如果 i 和 j 之间有边相连,则新社区 X 和新社区 Y 之间的模块度增量应加上社区 i 和社区 j 之间的模块度增量(第 6~7 行);如果社区 i 与社区 j 之间没有边相连,则应减去社区 i 向量值 $\boldsymbol{a}_i$ 与社区 j 向量值 a_j 乘积的 2 倍(第 8~9 行);

(3)根据式(8)更新向量 $\boldsymbol{a}$,将要合并社区的值 k 相加求得新社区的向量值 $\boldsymbol{a}_c$(第 14~18 行);最终输出更新后的图 G 和向量 $\boldsymbol{a}$.

3.3.4 社区发现结果生成

社区发现结束之后,需要将数据集中的冗余数据(主要是 $\Delta\boldsymbol{Q}$ 矩阵的数据)清除,保留初始节点集合以及它们所属的社区序号.此时,图中的节点存储结构为 $V=(vId,cId)$,其中,vId 表示节点编号,cId 表示社区序号,表明各个节点归属于哪个社区.算法 4 给出社区划分结果生成的过程,如算法 4 所示.

算法 4. 社区发现结果生成.

输入:最终图 G

输出:社区集合 C

1. FOR(each $v=(vId,cId)$ in G)
2. IF($cId \in C$) THEN
3. $g \leftarrow getNode(C,cId)$;
4. $c \leftarrow insert(g,vId)$;
5. $C \leftarrow insert(cId,c)$;
6. ELSE
7. $C \leftarrow add(cId,vId)$;
8. END IF
9. END FOR
10. FOR(each community c in C)
11. 输出 c;
12. END FOR

算法主要步骤为:

(1)遍历所有节点,将具有相同社区序号 cId 的节点保存在一起,如果节点属性 cId 已经在 C 中,则表示 cId 对应的社区已经出现过,需将已经存入 C 的节点取出,与当前的节点合并后再存入 C(第 3~5 行),否则直接存入 C(第 7 行);

(2)将社区和社区中的节点集合逐个写入 HDFS 文件系统中(第 10~12 行).

3.3.5 结果可视化

本文开发了一个社区发现可视化系统,4.3 节实验通过展示小数据集上的社区结构证明本文所提算法的有效性.首先,将社区划分结果和原始的数据输入到系统中,为每一个社区随机分配颜色及图形区域;然后,为每个社区中的节点随机分配坐标,这些坐标都必须在这个社区的图形区域内,且坐标不相同;最后,根据分配的坐标和颜色画出所有节点,根据原始边集,绘出每条边.

3.4 DBCS 算法复杂性分析

对于一个由 n 个节点,m 条边构成的图 $G(V,E)$,在 Spark 平台上的预处理过程需要遍历全部的边,而算法的初始化需要遍历所有节点,更改节点属性;然后遍历所有的边,更改边的属性,其时间复杂度为 $O(n+2m)$.最终社区划分结果生成需要遍历每个社区中所有的节点,最坏情况是所有节点都属于同一个社区,即时间复杂度为 $O(n)$,最好情况是每个节点独自为一个社区,时间复杂度为 $O(1)$,但是这类网络是没有意义的.

算法时间复杂度主要从两个方面考虑:(1)查找最大模块度增量;(2)社区合并与更新.查找最大模块度增量过程中,需要遍历一次所有边,然后寻找所有等于最大模块度的边,相当于遍历了两次边集合.然后,需要对找出来的边进行处理,当网络规模极大时,因为一个网络中的节点不可能只有一个社区,因此,这部分的耗时非常小.假设平均每次有 x 个节点需要合并,这一步骤的时间复杂度是 $O(x^2)$,整个查找最大模块度增量阶段的时间复杂度为 $O(2m+x^2)$.在合并与更新的过程中,首先需要找出待更新的边,边的数量一般是待合并社区数目的倍数,设为 r;然后,遍历这些边并计算新社区与其他未参与合并的社区或者新社区与新社区之间新的模块度增量;最后,遍历所有边,并更新节点社区序号和边的属性,对顶点集合及边集合各遍历一次,因此整个过程时间复杂度为 $O(2m+n+r^2x^2)$.“查找最大模块度增量”和“社区合并与更新”这两个过程需要进行迭代计算,每次迭代过程会合并 x 个节点,相比传统的 CNM 算法每轮迭代合并两个社区的方法,极大地减小了迭代次数.

结论 1. DBCS 算法的最坏情况为每次只合并两个社区，即最坏复杂度为 $O((4m+n)n)$，假设平均每轮迭代合并 x 个社区，则平均时间复杂度为 $O((4m+n-x)\times n/x)$.

3.5 DBCS 算法与传统算法对比分析

当前使用广泛的基于模块度思想的典型社区划分方法是 fast newman 社区划分方法[3]和 CNM 算法[5]，DBCS 算法基于模块度的思想，但是区别于上述两种算法，主要体现在如下几个方面：

(1) fast newman 和 CNM 算法每轮迭代过程只合并两个社区，执行效率非常低，而本文根据图论提出的多社区同时合并的思想(参见定理 1,2)，每轮迭代合并多个社区，减少迭代次数，极大地减少了对内存数据的操作，对于处理大规模复杂网络而言，能节约大量时间.

(2) fast newman 和 CNM 算法进行一轮社区合并之后，需要重新计算社区之间的模块度增量，重新构造模块度矩阵，不利于算法的并行化，因为重新计算社区之间的模块度增量需要每个社区内边的数量和每个社区之间边的数量，在 Spark 中搜寻这两个值是困难且耗时的，而本文根据网络性质提出的在合并时使用的模块度增量计算更新公式(参见性质 1 和推论 1)，只需计算一次模块度增量，更新时只将对应的模块度增量进行加减操作，极大地方便了 Spark 运算.

(3) fast newman 和 CNM 算法使用矩阵存储模块度增量，而复杂网络是一个稀疏网络，大部分矩阵内容为空，且当网络数据过大，矩阵维度也会变得非常庞大，内存开销大. 在 Spark 平台下，将数据缓存在内存的形式无法存储这样大的数据结构，而本文算法为了满足存储要求，只存储了边和节点以及它们的属性，大大减小了内存开销.

4 实验分析

4.1 数据集描述

为了验证本文所提大规模复杂网络社区并行发现算法的有效性和时间性能，实验采用 3 类数据集：(1) 人工模拟的大规模复杂网络数据集，使用 LFR 基准程序随机生成的网络图[23]；(2) 真实的复杂网络数据集，从 Stanford 大学网络数据分析项目组 Stanford Network Analysis Project(SNAP)获得的 3 个真实数据集①；(3) 常用小规模真实社会网络数据集合，用于展示算法社区识别效果.

LFR 基准程序是由 Lancichinetti 等人[23]提出的专门用于生成模拟网络的算法，该算法主要根据输入的参数，尽可能地生成符合真实网络特征的人工合成网络，同时也会生成与之对应的社区结构，因此具有较高的实验价值，可以用来检验社区识别算法的准确性. 算法在生成图时常用的参数有网络的节点数目 n，网络中节点的平均度数 k，最小社区的节点数目 $C_{\min}$，最大社区的节点数 $C_{\max}$，重叠节点的个数 O_n，重叠节点所从属的社区个数 O_m 以及混合参数 μ，它的取值范围为 0～1，该值越大表示网络社区结构越不明显. 本文的实验生成了节点数目分别为 1 万、10 万、50 万和 100 万的大规模网络图，且这些生成图中每个社区内所包含的节点数目在 1000～10000 之间. 表 2 给出实验中所用数据集的详细信息.

表 2　实验数据集描述

(a) 仿真网络数据集

名称	节点数 V	边数 E	μ	节点平均度($2E/V$)
A-1w	10000	77334	0.3	15.4468
A-10w	100000	1981482	0.3	39.6802
A-50w	500000	9920047	0.3	39.6296
A-100w	1000000	19520912	0.3	39.0418

(b) 真实网络数据集

名称	节点数 V	边数 E	节点平均度($2E/V$)	数据集描述
soc-Epinions1	75879	508837	13.4118	Epinions.com 数据集
web-NotreDame	325729	1497134	9.1925	Notre Dame 的 Web 图数据
soc-Pokec	1632803	30622564	37.5092	Pokec 社交网络数据

(c) 小规模数据集

名称	节点数 V	边数 E	节点平均度($2E/V$)	数据集描述
karate	34	78	4.5882	空手道俱乐部网络
jazz	198	2742	27.6970	欧洲爵士乐音乐家合作网络
football	115	616	10.7130	美国秋季大学生足球队网络
facebook	5000	8194	3.2776	facebook 的 5000 节点子网络

实验在 Spark 分布式计算框架中使用 Scala 语言实现，Scala 是一种多范式编程语言，集成了 Java 的面向对象编程和函数式编程的各种特性. Spark 集群包含 16 台服务器，每个服务器节点分配 4 GB 内存，服务器硬件配置为 Intel E5620 双 CPU 四核处理器，集群中设置 1 台 Master 节点，15 台 Worker 节点，每种算法均执行 3 次，取结果的平均值.

实验对比算法包括：Zhang 等人[8]提出的 OCDI

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

(non-Overlapping Community Discovery Idea)算法，DBCS 算法的串行实现算法(DBCS-s)，在 Hadoop 平台上实现的社区并行发现算法 DBCH(Discovering Big Community on Hadoop). OCDI 算法是利用 Java 实现的串行算法，其社区识别效果好、算法执行效率高，但其识别的网络规模有限，当网络节点规模超过 2 万条，便无法运行，可以同 DBCS-s 串行算法进行比较. DBCS 算法与 DBCH 算法采用相同的社区划分过程，不同点在于：(1) DBCS是基于 Spark 平台中 GraphX 图计算模型实现的并行算法，而 DBCH 是基于 MapReduce 编程模型实现的并行算法；(2) DBCS 利用 Scala 语言实现，而 DBCH 使用 Java 实现.

4.2 社区发现准确性分析

作为社区识别质量的评判标准，定义 5 给出的 *DA* 能够直观地表示社区识别的准确性，反应节点归属社区的正确性，本文将 *DA* 作为社区识别准确性的评判标准.

表 3 和图 4 给出了每种算法在不同数据集上的社区发现准确率，可以得出如下结论：

(1) OCDI 和 DBCS 算法节点划分质量较高，具有很高的社区发现准确率，DBCS 平均高出 OCDI 算法约 2.4%，而对于 facebook 数据集，OCDI 比 DBCS 算法的准确性高出 2.7%，原因在于使用多社区同时合并，导致下一轮的模块度增量矩阵与传统社区发现算法不同，但大多数情况下计算出 $\Delta \boldsymbol{Q}$ 增量矩阵更加准确. 而本文采用的 facebook 数据集与其他数据集不同之处在于，该数据集是从 facebook 数据集中截取的小数据集，节点编号小于 5000，拓扑结构不完整，含有较多小社区. 小社区较多会降低社区识别准确率，因此 DBCS 算法对 facebook 数据集的识别准确率较低.

(2) 串行算法 DBCS-s 的识别效果没有 DBCS 好，因为 DBCS-s 算法在进行社区合并时，进行合并的社区没有如 DBCS 算法对数据进行较好的整理，类似于快速 Newman 算法一样进行两两合并，求得的新 $\Delta \boldsymbol{Q}$ 值与 DBCS 算法有所不同.

(3) Hadoop 平台上的 DBCH 算法与 DBCS 算法的社区识别准确率相近，其原因是，虽然这两个算法是在不同的并行计算平台中实现，但它们的原理及实现的方法较为相似，因此处理结果也比较相近，但是 DBCS 算法的时间性能优势明显，这点将在 4.4 节给出说明.

(4) 如图 4 所示，当节点数目增大到 10 万，ODCI 和 DBCS-s 算法已经无法实现社区发现，故仅给出 DBCS 和 DBCH 算法的结果. 图 4 表明：DBCS 和 DBCH 算法在大规模复杂网络社区发现准确率均能保证在 70%以上，识别效果较好，两个算法在社区的识别准确率上也是比较接近的.

表 3 小规模数据集上社区发现准确率对比

	karate/%	football/%	jazz/%	facebook/%
OCDI	100	91.30	90.66	85.91
DBCS-s	100	83.47	80.19	81.73
DBCH	100	94.26	91.87	84.03
DBCS	100	94.57	92.10	83.21

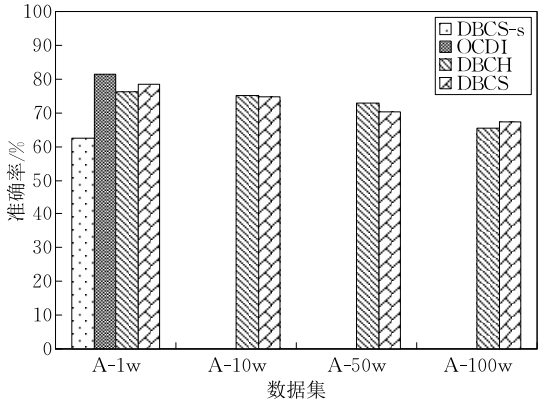


图 4 仿真数据集上社区发现准确率对比

用 LFR 基准程序生成仿真网络时，参数 μ 决定了网络社区结构的明显程度，它的取值越大说明社区结构越不明显. 这部分实验使用 A-1w 数据集为实验，取不同的 μ 值随机生成网络图.

图 5 给出了各种算法在 μ 值变化时，相同网络节点数和平均度不变时的社区发现准确率的对比，当 μ 值不断增大，社区结构越来越模糊，4 个算法的识别率都呈下降趋势. 可见参数 μ 对算法社区发现质量具有较大的影响. 从图 5 中可以发现：(1) DBCS、DBCS-s、DBCH 这 3 个算法在 μ 值较小时，识别准

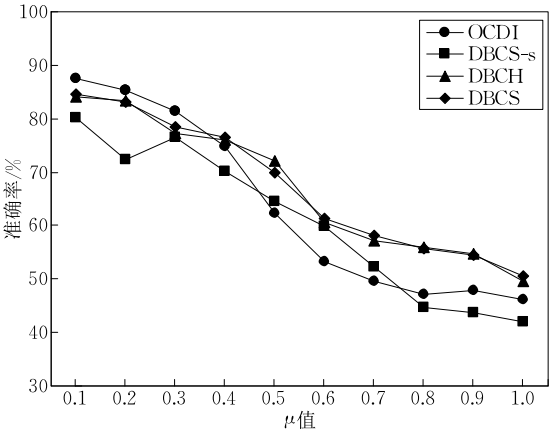


图 5 μ 值变化时不同算法社区发现准确率对比

确率低于OCDI算法.而当$\mu>0.4$时,DBCS和DBCH算法识别的准确率比OCDI算法高,平均高出7.4%,这表明本文提出算法在对社区结构较为模糊的网络中具有较好的性能优势.这是因为多社区同时合并,一方面减少了算法的迭代次数,进而减少了模块度增量的计算;此外,DBCS采用的新的模块度增量计算方法是通过加减已有的模块度增量,在社区结构模糊的网络中,计算的模块度增量更加准确;(2)而DBCS-s算法的识别率始终比DBCS算法低,在$\mu\in(0.5,0.75)$时效果比OCDI算法好,原因与上一节实验结论(2)类似.

4.3　社区识别质量分析

聚类系数(Clustering Coefficient,CC)[24]是一个比较重要的社区划分评价指标,网络的社区结构和搜索性能等问题均与其有密切关系,因此本文使用聚集系数来衡量DBCS算法在真实大规模复杂网络数据上的社区识别的质量.

定义6(聚集系数)[24].　用于描述网络的聚集度,计算公式如下:

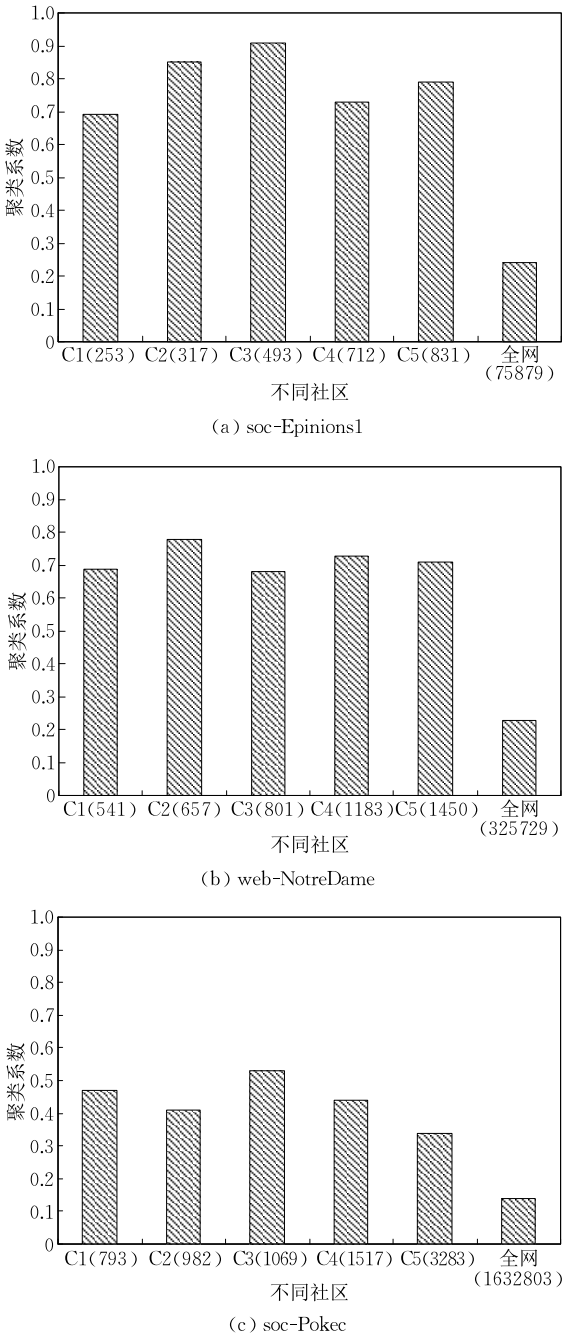
$$C_i=\frac{2|e_{jk}|}{k_i(k_i-1)} \tag{9}$$

其中,e_{jk}表示节点i的相邻节点j和k之间的连接边,k_i表示节点i的度.聚集系数$C(i)\in[0,1]$,当$C(i)=1$时,表示该社区是一个完全图,任意节点间都有边相连.整个网络的聚集系数(全网CC)是所有节点聚集系数的平均值,即

$$\overline{C}=\frac{1}{n}\sum_{i=1}^{n}C_i \tag{10}$$

根据参考文献[24],若一个社区内所有节点的平均CC大于或等于整个网络作为一个社区时所有节点的平均CC的3倍,说明所识别出的社区结构是有意义的,进而说明了社区识别质量很高.

图6实验采用3个真实大规模网络数据集,每个网络随机抽取了5个社区分别计算其社区CC(社区选取公式为:$i=k\times(n\%256)$,i表示社区序号,$k=1,2,\cdots,n$为社区数目)以及3个数据集的全网CC.可以发现:(1)不同数据集上,社区CC和全网CC的值各不相同,且社区CC均高于全网CC;(2)3个数据集中,大部分社区CC都在全网CC的3倍以上,只有图6(a)、(c)中都有1个社区CC低于全网CC的3倍,从网络聚集系数的角度进一步证明了DBCS算法的社区识别质量比较高,原因在于DBCS算法每轮迭代求得的模块度增量更加准确,社区划分更为合理.



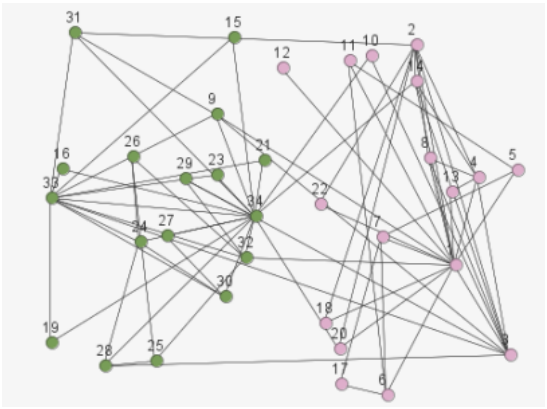
(a) soc-Epinions1

(b) web-NotreDame

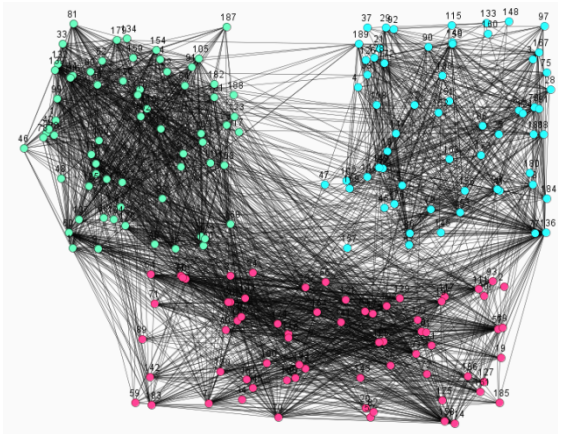
(c) soc-Pokec

图6　真实数据集上DBCS算法聚集系数

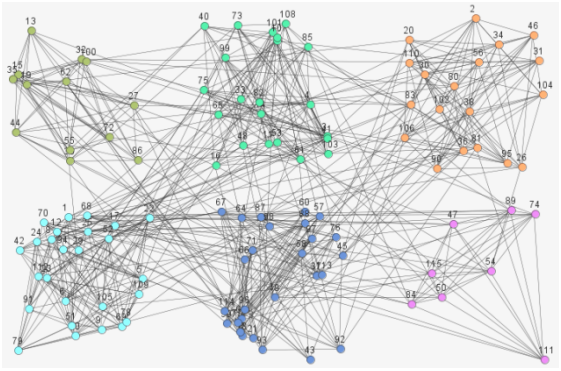
图7(a)~(c)分别展示了DBCS算法识别karate、jazz、football这3个网络的社区发现的可视化结果,可以发现,DBCS算法在小规模数据集上(如表3所示),识别质量很高,发现的社区数目以及其包含的节点都同传统小社区识别算法相似.此外,对比图7(b)和图7(c)可以发现,虽然jazz数据集同football数据集节点规模相差不大,但是jazz数据集的网络复杂度要比football数据集高很多,jazz社区识别准确率相比football数据集要低2%,说明网络复杂程度对DBCS算法社区发现质量也有一定程度影响,与客观事实相符.



(a) karate社区结构



(b) jazz社区结构



(c) football社区结构

图 7 社区识别结果可视化

4.4 DBCS 算法运行效率分析

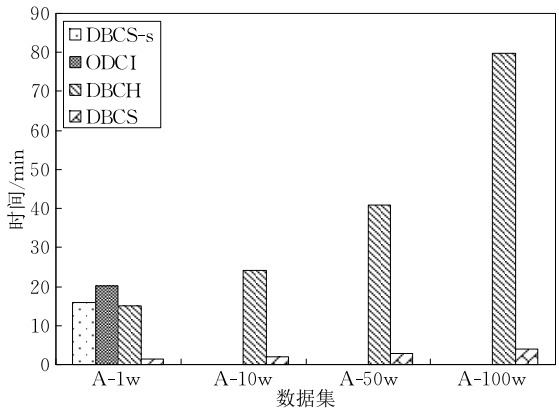
DBCS 算法是基于 Spark 平台对大规模网络的社区并行发现算法,对于大数据处理而言,算法的运行时间是评判算法性能的主要条件之一. 在该实验部分,分别在小规模数据集(参见表 2(c)),大规模真实网络数据集(参见表 2(b))以及大规模仿真网络数据集(参见表 2(a))上进行,实验结果如表 4 所示.

从表 4 可以发现,在规模小数据集上,算法 OCDI 和 DBCS-s 比算法 DBCS 和 DBCH 要快很多,这是因为 DBCS 和 DBCH 在处理小规模数据时,初始化任务分配和节点之间的数据传输占用了大部分时间,导致 DBCS 算法和 DBCH 算法比其他两个串行算法要慢很多. 表 4 中实验结果表明:DBCS 算法较 DBCH 算法执行效率快 9.5 倍左右,这是因为 DBCH 算法基于 Hadoop 平台,其与 Spark 最大的不同点在于 Hadoop 迭代过程中包含较多的文件操作,使得 Spark 效率比 Hadoop 平台高很多[21]. 此外,与两个平台采用的编程环境也有关系. 而 DBCS-s 算法执行时间比 OCDI 算法执行时间快 9.8 倍左右,可以看出本文的串行算法 DBCS-s 在小数据集上的表现较 OCDI 算法要好,因为 DBCS-s 算法每一轮合并的社区数目增多,总的迭代次数减少.

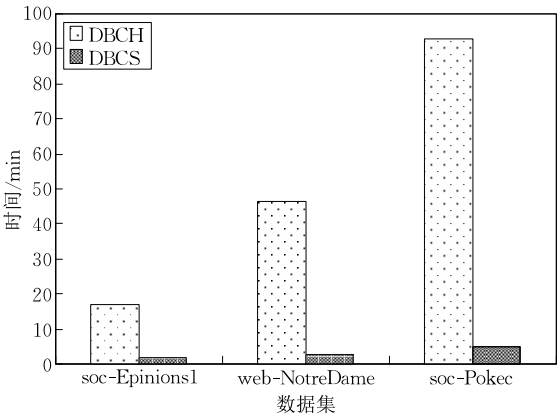
表 4 小规模数据集上不同算法运行时间 (单位:s)

	OCDI	DBCS-s	DBCH	DBCS
karate	0.023	0.003	20.20	2.185
football	0.080	0.003	44.024	3.951
jazz	0.226	0.050	77.848	6.047
facebook	126.730	28.351	581.374	67.794

当节点数目不断增大到 10 万以上时,由于 OCDI 和 DBCS-s 算法无法处理这么大规模的社区数据,因此图 8 中只能显示 DBCS 和 DBCH 的执行时间.



(a) 仿真网络下运行时间



(b) 真实网络下并行算法运行时间

图 8 社区并行发现算法运行时间对比

从图 8(a)、(b)可以看出，大规模社区(节点数目＞10 万时)，在仿真网络和真实网络上，DBCS 算法比 DBCH 算法分别快 13.3 倍和 14.6 倍，DBCS 算法执行效率非常高，这主要依赖于 Spark 的并行计算的优势，即将数据缓存在内存中，省去了如在 Hadoop平台中大量的文件操作.

5　总　结

本文重点阐述了大规模复杂网络并行社区发现算法 DBCS 的工作原理，算法基于 Spark 并行计算框架，解决了大规模复杂网络中社区的识别问题. 算法思想是基于模块度增量计算方法，初始化属性图的各个结构，然后结合图的相关理论，同时从网络中找出多个满足合并条件的节点，减少算法迭代次数. 在更新时根据网络的性质，将多个节点合并，同时计算新社区与其他社区之间的模块度增量. 实验在仿真和真实大规模复杂网络上进行，与串行和并行算法进行对比，验证了本文所提算法对大规模复杂网络社区识别的有效性和时效性，进一步证明了 DBCS 算法在大规模复杂网络社区挖掘中的优越性.

未来工作包括：算法执行时间仍然达不到实时的效果，需要进一步提高系统运行效率[25]. 本文算法主要针对静态复杂网络进行社区识别，而且目前重叠社区的识别能够更加真实地反应网络结构特征，下一步将深入研究 Spark 平台下大规模网络中重叠社区识别问题. 此外，大规模复杂社区进化问题也是重要的研究方向，对网络结构特性演变的认识和个体行为趋势的分析，可以控制网络舆情的传播，具有重要的社会意义.

参　考　文　献

[1] Barabási A, Albert R, Jeong H, Bianconi G. Power-law distribution of the World Wide Web. Science, 2000, 287(5461): 2115

[2] Newman M E J, Girvan M. Finding and evaluating community structure in networks. Physical Review E, 2004, 69(2): 026113

[3] Newman M E J. Fast algorithm for detecting community structure in networks. Physical Review E, 2004, 69(6): 066133

[4] Newman M E J. Networks: An introduction. Astronomische Nachrichten, 2010, 327(8): 741-743

[5] Clauset A, Newman M E J, Moore C. Finding community structure in very large networks. Physical Review E, 2004, 70(6): 066111

[6] Pan Lei, Jin Jie, Wang Chong-Jun, Xie Jun-Yuan. Detectinglink communities based on local information in social networks. Acta Electronica Sinica, 2012, 40(11): 2255-2263(in Chinese)
(潘磊，金杰，王崇骏，谢俊元. 社会网络中基于局部信息的边社区挖掘. 电子学报，2012，40(11)：2255-2263)

[7] Leng Zuo-Fu. Network community discovery algorithm based on greedy optimization techniques. Chinese Journal of Electronic, 2014, 42(4): 723-729(in Chinese)
(冷作福. 基于贪婪优化技术的网络社区发现算法研究. 电子学报，2014，42(4)：723-729)

[8] Zhang Xue-Wu, You Huang-Bin, Zhu W, et al. Overlapping community identification approach in online social networks. Physica A, 2015, 421: 233-248

[9] Huang Wei-Ping. Web Community Discovery Algorithm [M. S. dissertation]. Beijing University of Posts and Telecommunications, Beijing, 2013(in Chinese)
(黄伟平. Web 社区发现算法的研究[硕士学位论文]. 北京邮电大学，北京，2013)

[10] Xiong Zheng-Li. The Study of Community Detecting Technology and Application in Online Social Networks [M. S. dissertation]. Central South University, Changsha, 2012(in Chinese)
(熊正理. 在线社交网络中社区发现技术及其应用研究[硕士学位论文]. 中南大学，长沙，2012)

[11] Blondel V D, Guillaume J L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008, 30(2): 155-168

[12] Parsa M G, Mozayani N, Esmaeili A. An EDA-based community detection in complex networks//Proceedings of the 7th International Symposium on Telecommunications. Tehran, Iran, 2014: 476-480

[13] Oliveira J E M, Quiles M G. Communiry detection in complex networks using coupled kuramoto oscillators//Proceedings of the 14th International Conference on Computational Science and Its Applications. Guimaraes, Portugal, 2014: 85-90

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

[15] Li Jin-Peng. Overlapping Community Discovery Algorithm Based on Hadoop [M. S. dissertation]. Jilin University, Changchun, 2014(in Chinese)
(李金鹏. 基于 Hadoop 平台的重叠社区发现算法研究[硕士学位论文]. 吉林大学，长春，2014)

[16] Riedy J, Bader D A, Meyerhenke H. Scalable multi-threaded community detection in social networks//Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum. Shanghai, China, 2012: 1619-1628

[17] Moon S, Lee J-G, Kang M. Scalable community detection from networks by computing edge betweenness on MapReduce// Proceedings of the 2014 International Conference on Big Data and Smart Computing. Bangkok, Thailand, 2014: 145-148

[18] Staudt C L, Meyerhenke H. Engineering parallel algorithm for community detection in massive networks. IEEE Transactions on Parallel and Distributed Systems, 2015, 27(1): 171-184

[19] Chen W-Y, Song Y, Bai H, et al. Parallel spectral clustering in distributed systems. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(3): 568-586

[20] Papadimitriou S, Sun J. DisCo: Distributed co-clustering with map-reduce//Proceedings of the 8th IEEE International Conference on Data Mining. Pisa, Italy, 2008: 512-521

[21] Zaharia M A. An architecture for fast and general data processing on large clusters. University of California, Berkeley: Technical Report UCB/EECS-2014-12, 2014

[22] Zhang Ai-Dong. Protein Interaction Networks. New York: Cambridge University Press, 2009: 44-47

[23] Lancichinetti A, Fortunato S. Limits of modularity maximization in community detection. Physical Review E, 2011, 84(6): 066122

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

[25] Qiao Shao-Jie, Li Tian-Rui, Han Nan, et al. Self-adaptive trajectory prediction model for moving objects in big data environment. Journal of Software, 2015, 26(11): 2869-2883 (in Chinese)
(乔少杰, 李天瑞, 韩楠等. 大数据环境下移动对象自适应轨迹预测模型. 软件学报, 2015, 26(11): 2869-2883)

QIAO Shao-Jie, born in 1981, Ph.D., professor. His current research interests include big data, online social networks and moving objects databases.

GUO Jun, born in 1991, M. S. His current research interests focus on community discovery in complex networks.

HAN Nan, born in 1984, Ph. D., lecturer. Her current research interests focus on complex networks.

ZHANG Xiao-Song, born in 1968, Ph. D., professor. His current research interests focus on complex networks.

YUAN Chang-An, born in 1964, Ph. D., professor. His current research interests focus on data mining.

TANG Chang-Jie, born in 1946, M. S., professor, Ph. D. supervisor. His current research interests focus on databases.

Background

This work is a part of the "Research on Topological Property Analysis and Community Evolution Mechanism of Online Social Networks", which is mainly supported by the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No. 20110184120008. The project focuses on the research of analyzing the static topological structures and the dynamic evolution mechanism of communities in online social networks. The difficulties and essential problems contain: (1) using the topological property analysis theories in complex networks to evaluate the shrinking diameter property and densification law; (2) proposing new approaches to discover communities in large-scale networks; (3) clarifying the evolution mechanism and theories of communities, using the event-based framework to formalize the phase of social evolution, and revealing the key factors and dynamic theories in each phase of evolution in online social networks.

This paper presents a parallel algorithm for discovering communities in large-scale complex networks based on Spark. The proposed approach employs the basic idea of clustering method on modularity. Extensive experiments are conducted on large-scale real and synthetic network data and the results demonstrate that the proposed approach can effectively deal with the problem of partitioning the large-scale networks, which only takes about four minutes to process more than one million nodes.