基于流式分析的大规模网络重叠社区发现算法

李 辉,张建朋,陈福才
(信息工程大学信息技术研究所,河南郑州 450002)

摘 要: 为了提高在大规模网络中发现社区的效率,提出一种基于流式分析的大规模网络重叠社区发现算法(Streaming-based Overlapping Community Detection algorithm,SOCD). 算法对网络中的边进行流式处理,每次只处理一条边且每条边仅被处理一次. 根据节点的度、节点对社区的贡献度以及节点移动前后社区间连边数量的变化等信息对节点进行划分. 在人工合成网络和真实大规模网络上的一系列实验表明,SOCD算法在时间消耗和内存占用上具有较大的优势,比传统方法快10倍以上,且具有较强的鲁棒性,能够在线性时间和空间复杂度下高效、准确地挖掘网络中的重叠社区结构.

关键词: 社区发现; 重叠社区; 大规模网络; 流式分析; 节点贡献度
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112(2022)08-1951-08
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20201422

A Streaming-Based Overlapping Community Detection Algorithm in Large-Scale Network

LI Hui,ZHANG Jian-peng,CHEN Fu-cai
(*Institute of Information Technology*,*Information Engineering University*,*Zhengzhou*,*Henan* 450002,*China*)

Abstract: To improve the efficiency of community detection in large-scale network, a streaming-based overlapping community detection algorithm(SOCD) is proposed. SOCD processes the edges in a streaming fashion and handles one edge at a time, and each edge is processed only once. The algorithm discovers the community based on the degree of node, the contribution of node to the community, and the change of the number of connected edges between communities. Extensive experiments on synthetic networks and real large-scale networks show that SOCD has great advantages in time consumptions and memory footprints. It is more than 10 times faster than traditional methods and has strong robustness. It can efficiently and accurately mine the overlapping community structures in the network under linear time and space complexity.

Key words: community detection; overlapping community; large-scale network; streaming analysis; node contribution

1 引言

复杂网络普遍存在于物理自然世界和人类社会之中,近些年来受到了人们的广泛关注. 复杂网络可以将各个领域的实体以及实体之间的关系进行抽象. 复杂网络虽然形态各异,但仍然表现出高度的有序性和组织性,呈现出明显的社区结构. 一个良好的社区应该具有内部连接紧密、外部连接稀疏的特性. 利用网络拓扑中所蕴含的信息从网络中发现模块化的结构则被称为社区发现,也被称为图聚类或社区检测. 社区发现对复杂网络的拓扑分析、功能分析和行为预测具有重要的理论意义和实践价值. 早期的社区发现算法主要将节点划分到单个社区. 然而,真实网络中的社区经常相互重叠,一个节点可以属于多个社区,这些重叠节点对信息的传播和网络的演化具有重要价值[1]. 一方面,重叠节点具有“多面性”,对重叠节点的挖掘有利于更全面地了解节点的特性. 另一方面,重叠节点是社区间的桥梁,对社区间的交互和社区演化能够发挥关键作用. 因此,识别出这些重叠社区对理解真实网络的结构和功能具有重要意义.

近些年来,众多社区发现算法相继被提出来解决

收稿日期:2020-12-11;修回日期:2021-06-17;责任编辑:王天慧
基金项目:国家自然基金青年基金(No.62002384);郑州市协同创新重大专项(No.162/32410218);中国博士后科学基金面上项目(No.2020M683760)

各个领域的实际问题,如基于模块度优化[2]、基于局部扩展[3,4]、基于图表示学习[5]等算法. 然而,随着互联网的发展,网络的规模越来越大,已有的很多算法在计算时间和内存消耗上都无法对如此规模巨大的网络进行处理. 目前,解决网络数据规模过大问题主要有2种思路. 一是采用并行算法[6,7],利用多个集群来加速运算,将原有的一些经典算法进行改进以适应分布式运算,但这种方法需要大量的计算资源. 二是基于流式分析[8,9],核心思想是以流的方法对网络中的边进行单遍扫描处理,每条边只需处理一次,不需要一次性将整个网络读入内存,时间复杂度和空间复杂度都较低,适用于处理大规模网络. 然而,文献[8]所提算法需要先验信息,使得算法实用性不高. 文献[9]利用的网络结构信息较少,算法精度有待进一步提升,且不支持重叠社区的检测. 为了解决上述存在的问题,本文提出一种基于流式分析的大规模网络重叠社区发现算法(Streaming-based Overlapping Community Detection algorithm,SOCD). 本文主要贡献包括:

(1)算法通过对网络中的每条边进行单遍扫描处理,每条边仅被处理一次,拥有线性的时间和空间复杂度,无需任何先验信息,能够快速地挖掘出大规模网络中隐藏的社区结构,并且可以识别出网络中的重叠社区;

(2)算法引入节点的贡献度来衡量节点与社区之间的紧密程度,并且与节点移动前后社区之间的连边数量等信息共同指导节点的划分,提高了社区划分的精度;

(3)在人工合成网络以及6个大规模真实网络上的实验结果表明了SOCD算法的高效性和鲁棒性,SOCD算法在运算时间上具有较大的优势,相较于非流式算法,SOCD算法要快10倍以上,并且在大规模网络上能够取得近似甚至更好的社区划分精度.

2 相关工作

随着社交网络、物联网的兴起,网络的规模呈爆炸式增长,已有的许多算法无法有效处理这些规模庞大的网络. 近些年,一种新的解决方案被提出用来解决网络规模过大的问题,即以流的方式对网络进行处理. 基于流式方法的核心思想是对网络中的边进行单遍扫描处理[10],无需一次性将整个网络读入内存,拥有较低的时间和空间复杂度,能够提升在大规模网络中挖掘社区的效率. Yun等人[11]提出了一种空间消耗随网络线性增长的算法来解决内存限制的问题. 该算法首先构建网络的邻接矩阵,然后顺序读取邻接矩阵的列向量进行处理,通过对邻接矩阵的部分信息进行局部谱聚类,从而对节点进行划分. 算法无需一次性处理整个邻接矩阵,只需顺序的读取邻接矩阵的列向量进行运算即可. 算法所需内存小,但需要提前构建网络的邻接矩阵且较为耗时. Liakos等人[8]提出了基于种子集扩展的流式社区发现算法. 算法通过流的方式读取网络中的每条边并进行处理,在时间窗口内,通过种子向外扩张,然后对社区进行剪枝以保证社区的质量,并且能够自动确定社区的大小. 算法具有较高的计算效率,通过特殊的数据结构,所需空间与网络规模成正比. 但是算法需要提前指定社区的个数,这使得实用性大大降低. Hollocou等人[9]提出了线性流式的社区发现算法. 算法基于社区内边的数量要远大于社区之间边的数量的特性,认为在网络中随机的选取一条边,这条边是社区内的边的概率要远大于是社区间边的概率. 因此,当网络中的边以随机的顺序到达,算法根据边到达的先后顺序以及节点的度信息来判定是社区内的边还是社区间的边. 算法以流的方式处理每一条边且每条边只处理一次,可以快速处理大规模网络并且所需内存小. 但算法仅通过节点度大小来判断节点的移动,处理规则过于简单,会造成很多错误的移动,并且该算法只能挖掘出非重叠社区.

针对以上流式算法存在的问题,本文提出了一种基于流式分析的大规模网络重叠社区发现算法(SOCD). 算法在2个方面进行优化以解决上述问题. 一是保证算法在具有线性计算复杂度的情况下利用更多的信息对每条边进行处理,提高社区划分的质量. 二是将其扩展到重叠社区领域以挖掘出真实网络中存在的大量重叠社区结构. 本文定义了节点的贡献度来量化节点与社区的紧密程度,并且与节点的度信息以及节点移动前后社区之间连边数量的变化信息来对每条边进行处理,更加合理地对节点进行划分,提高社区划分的精度. 同时,算法可以根据节点信息识别出网络中的重叠节点.

3 算法描述

3.1 相关定义

节点的贡献度. 节点v对社区C的贡献度定义为

$$\mathrm{Con}_C^v=\frac{\left|N_C^v\right|}{\left|N(v)\right|} \tag{1}$$

其中,$\left|N_C^v\right|$表示节点v的邻居节点在社区C中的数量,$\left|N(v)\right|$表示节点v的邻居节点数量. Con_C^v越大,表示节点v对社区C的贡献值越大,节点v与社区C的联系越紧密.

给定网络$G(V,E)$,V是网络中的节点集,$E\subseteq V\times V$代表网络中的边集,$N=|V|$表示网络中节点的数量,$M=|E|$表示边的数量. 假定$A\in V,B\in V$,给出以下定义:

$$e(A)=\left\{(u,v)\in E:u\subseteq A\ \text{or}\ v\subseteq A\right\} \tag{2}$$

$$e(A,B)=\left\{(u,v)\in E:u\subseteq A\ \text{and}\ v\subseteq B\right\} \tag{3}$$

$$\overline{A}=V\backslash A \tag{4}$$

由以上定义可知$e(A,A)\subseteq e(A)$.

假定$C\subset V$为要挖掘的社区,随机不重复地从网络中选取边,定义从$e(C)$中最先选取的k条边属于$e(C,C)$为事件$\mathrm{Intra}_k(C)$,则此事件的概率为

$$P\left[\mathrm{Intra}_k(C)\right]=\prod_{l=0}^{k-1}\frac{\left|e(C,C)\right|-l}{\left|e(C)\right|-l}=\prod_{l=0}^{k-1}\left(1-\phi_l(C)\right)\quad(5)$$

其中

$$\phi_l(C)=\frac{\left|e\left(C,\bar{C}\right)\right|}{\left|e(C,C)\right|+\left|e\left(C,\bar{C}\right)\right|-l}\quad(6)$$

当l较小时,$\phi_l(C)$非常接近社区C的连通度(conductance)$\phi(C)$,

$$\varphi(C)=\frac{\left|e\left(C,\bar{C}\right)\right|}{\left|e(C,C)\right|+\left|e\left(C,\bar{C}\right)\right|}\quad(7)$$

根据社区的性质,一个良好的社区应该内部连接紧密,外部连接稀疏,具有较低的连通度.因此,当C为一个良好的社区时,l取较小值,则最先选择的k条边有更大概率是社区内的边.

3.2　基于流式分析的大规模网络重叠社区发现算法

算法在处理每一条边$e=(u,v)$时,如果两端的节点在一个社区之内,则对此条边不做处理.如果两端节点位于不同社区,则需要对两端节点的社区归属重新判定.在文献[9]中,根据d_u和d_v是否小于阈值D来进行判定.如果d_u或d_v大于给定的阈值D,算法认为节点u或v在之前已被处理过多次,大概率是社区间的边,则不做处理;否则,将度小的节点移动到度大的节点所在的社区.在实验中发现,仅利用节点的度信息来判断节点的社区归属会造成一些错误.这样的判定准则太过简单,利用的信息太少,会在节点移动的过程中使移动方向发生错误,造成节点u本应该从社区A移动到社区B,但实际却是节点v从社区B移动到了社区A.这个错误的移动不仅损害了两个社区的质量,还会对后续节点的划分造成错误累计,从而使得整个网络的社区划分精度不高.发生这种问题的主要原因在于仅利用节点的度信息无法准确衡量节点与社区之间的紧密程度,无法准确判断移动方向.基于以上问题,本文定义节点贡献度来衡量节点与社区的紧密程度.由贡献度定义可知,节点u在社区C中的邻居节点越多,则节点u与社区C的联系越紧密,对社区的贡献度也越大,有着更大的概率被划分到社区C中.因此,在判断节点移动方向时不仅需要考虑节点的度信息,还需要考虑节点对每个社区的贡献度大小.由社区的特性可知,一个社区内部连接越紧密,外部连接越稀疏,则社区的质量越高.节点在移动的过程中会造成两个社区间连边的数量的变化,为了提高发现社区的质量,期望节点移动后社区之间的连边数量变少.综上所述,在判断节点的移动方向时,需要将节点的度、节点对每个社区的贡献度以及移动前后社区之间连边数量的变化等信息考虑在内.这样可以更加精细地对节点进行划分,降低节点划分的错误率.

如图1所示,当一条新的连边(2,4)到达时,若仅根据节点度信息来移动,则节点2应该移动到节点4所在的社区.但是很显然,这破坏了社区D的结构,而且使得社区D和社区C之间的连边增多.这是因为,虽然节点4的度比节点2的度大,但是节点2的邻居节点都在社区D内部,而节点4的邻居节点只有1个在社区C,节点2与社区D的紧密程度要大于节点4与社区C之间的紧密程度.为了解决上述问题,本文采用度信息来判断节点是否需要移动,用节点对社区的贡献度以及移动前后社区间的连边数量变化来判断节点的移动方向.

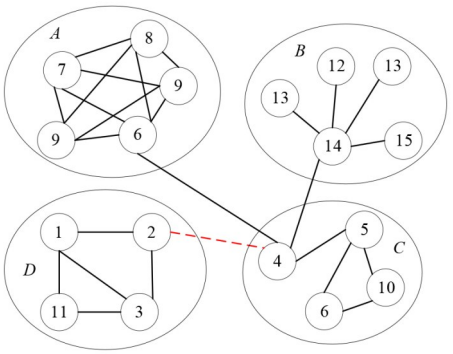

图1　一条新边到来时的处理示意图

本文所提SOCD算法的处理过程如算法1所示.算法的输入为网络的边的序列以及阈值D,在实验中阈值D取网络中节点度的众数,阈值D的选取会在后续内容进行探讨.为了能够随机读取网络中的边,首先将网络中边的顺序打乱,并对所有节点进行初始化.当读取一条新的边$e=(u,v)$时,节点u和v的度增加1.如果其中一个节点的度为1,则将其加入到另一个节点所在的社区;如果都为1,则将两个节点划分到一个社区,这样可以避免孤立节点的小社区.如果d_u和d_v都小于或等于D,则需要考虑节点是否为重叠节点以及节点是否需要移动并判断移动方向.此处需要衡量两个指标:一是节点对每个社区的贡献度,节点对社区的贡献度越大,节点与社区之间的联系越紧密,在移动时应将贡献度小的节点移动到贡献度大的节点所在的社区;二是节点移动前后两个社区之间连边数量的变化,连边数量越少,社区的质量越高,因此应该使移动后社区之间的连边数量比移动前社区间的连边数量少.根据这两个指

标,对每一条新来的边进行节点划分.

首先计算节点u和v对各自社区的贡献度,若$\mathrm{Con}_C^u>\mathrm{Con}_C^v$,根据贡献度指标应将节点$v$移动到节点$u$所在的社区,但还需要考虑移动前后社区之间连边数量的变化.定义节点移动后社区间连边数量与移动前社区间连边数量的差值为ΔN.如果$\Delta N<0$,说明此移动方向使两个社区之间的连边数量变少,让两个指标都得到了优化,提升了社区的质量.如果$\Delta N>0$,则说明节点v有较多邻居节点在其他社区.根据社区的性质,节点v大概率是一个重叠节点,因此把节点v作为重叠节点并加入到u所在的社区.如果$\mathrm{Con}_C^u=\mathrm{Con}_C^v$,若移动前后$\Delta N<0$,使社区间的连边数量变少,则说明也是一个正向的移动;若$\Delta N=0$,那么移动前后对两个目标都没有得到优化,因此不做任何处理.由以上的判断准则,可以对网络中的每一条边进行处理并对节点进行划分.

3.3 复杂度分析

SOCD算法的运算时间主要包括三部分.第一部分计算阈值D,阈值D为网络中节点度的众数,只需要将所有的边扫描一遍,时间复杂度为$O(M)$;第二部分是将网络中边的顺序随机化,复杂度为$O(M)$;第三部分是算法的核心处理过程,算法流式的对每一条边进行处理,时间复杂度为$O(M)$.因此,SOCD算法总的时间复杂度为$O(M)$.相较于其他算法,SOCD降低了时间复杂度,能够快速处理大规模网络.算法对每条边进行处理时,仅需比较节点的度、节点对社区的贡献度以及节点移动前后社区间连边数量的变化,这些信息计算简单,可以非常快速地完成.这使得算法可以在极短的时间内对每条边进行处理.SOCD算法的运行时间与边的数量成线性关系,随着网络规模的持续增长,算法仍能快速地对其进行处理.

对于算法的空间复杂度,计算节点度的众数时需要大小为n的数组,复杂度为$O(N)$;算法在处理每一条边时,需要存储每个节点当前的度数,复杂度为$O(N)$;需要存储节点当前所在社区的编号,复杂度为$O(N)$;需要存储节点对当前所在社区的社区贡献度,复杂度为$O(N)$;还需要存储每个社区内的边,复杂度为$O(M)$.因此,SOCD算法总的空间复杂度为$O(N+M)$,表明了算法拥有线性的空间复杂度.

4 实验评估

为了验证本文所提SOCD算法的有效性和高效性,分别采用不同规模的人工合成网络以及真实世界的大规模网络来进行实验,并与其他几种高效算法进行对比实验.实验环境为:Ubuntu16.04操作系统,Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz,128 GB内存的服务器,算法实现为Python 3.6.

算法1 基于流式分析的大规模网络重叠社区发现算法(SOCD)

输入: 网络中边的序列$e=\{e_1,e_2,\cdots,e_m\}$,阈值D

输出: 社区集合$C=\{C_1,C_2,\cdots,C_n\}$

```
随机打乱网络中边的顺序
FOR all nodes i=1,2,⋯,n, d_i=0, c_i=i  AND  Con_c^i=0
FOR j=1,⋯,|E| DO
    (u,v)=j^th edge of E
    d_u=d_u+1  AND d_v=d_v+1
    IF d_u=1  THEN C_v←C_u
    END IF
    ELSE IF d_v=1 THEN C_u←C_v
    END IF
    ELSE IF d_u≤D  AND d_v≤D  THEN
        计算ΔN
        IF Con_C^u>Con_C^v  AND ΔN≤0  THEN C_u←C_v
        END IF
        IF Con_C^u>Con_C^v  AND ΔN>0  THEN
            节点v是重叠的节点,并将v添加到社区C_u
        END IF
        IF Con_C^u=Con_C^v  AND ΔN=0  THEN
            PASS
        END IF
        IF Con_C^u=Con_C^v AND ΔN<0 THEN C_u←C_v
        END IF
    END IF
END FOR
RETURN C
```

4.1 对比算法

为了评估本文提出算法的性能,将本文算法与其他6种高效算法进行了对比,其中既有非重叠社区算法也有重叠社区算法,既有流式算法也有非流式算法.

4.1.1 非重叠社区发现算法

Louvain[2]:基于模块度优化的高效算法.

4.1.2 重叠社区发现算法

CoEuS[8]:基于流式的局部扩展社区发现算法.

BigClam[12]:基于非负矩阵分解的重叠社区发现算法.

OCSC(Overlap Compact Structure Community detection algorithm)[13]:基于结构紧密性的重叠社区发现算法,可分布式并行处理大规模网络.

CommunityGAN[14]:基于生成对抗网络的社区发现算法,通过模体级(motif-level)生成器和判别器之间的竞争,迭代地对学习到的向量进行优化并输出最终的社区结构.

LCDNN(Local Community Detection algorithm based on NGC Nodes)[4]:基于节点中心性的社区发现算法,通过局部扩展来挖掘网络中的重叠社区.

4.2 数据集

为了评估算法的性能,分别将7种算法在人工合成网络以及真实网络上进行对比实验.

4.2.1 人工合成网络

采用LRF-benchmark基准程序[15]生成两组人工合成网络D1和D2. D1用来测试算法在不同规模网络数据集下的性能,详细参数如表1所示. 其中,N表示网络的节点数,k表示网络的平均度,on表示网络中重叠节点的比例,μ表示混淆系数,μ越小说明网络的社区结构越明显. D2为了验证SOCD算法的鲁棒性,生成了不同混淆系数下的网络,其中N=100 000,k=20,μ为0.1~0.7.

表1　人工合成网络D1参数

N	k	on	μ
1 000	15	0.1	0.1
10 000	15	0.1	0.1
100 000	15	0.1	0.1
1 000 000	15	0.1	0.1

4.2.2 真实网络

使用斯坦福大学社交网络分析项目(Stanford Network Analysis Platform,SNAP)提供的6个公开的真实网络对算法进行实验评估,这些网络包括商品共同购买网络(Amazon)、科研协作网络(DBLP)以及社交网络(YouTube,LiveJournal,Orkut,Friendster). 这些网络的规模从三十万节点到六千万节点,从一百万条边到2亿条边,每一个网络都带有节点真实的社区标签. 这些大规模且带有标签的数据集可以很好地测试算法的性能,网络的特征如表2所示.

表2　6个大规模真实网络的特征

数据集	节点数	连边数	平均度	平均集聚系数	社区数
Amazon	334 863	925 872	5.52	0.396 7	75 149
DBLP	317 080	1 049 866	6.62	0.632 4	13 477
YouTube	1 134 890	2 987 624	5.26	0.080 8	8 385
LiveJournal	3 997 962	34 681 189	17.34	0.284 3	287 512
Orkut	3 072 441	117 185 083	76.28	0.166 6	6 288 363
Friendster	65 608 366	1 806 067 135	55.06	0.162 3	957 154

4.3 评价指标

由于上述网络节点的社区标签已知,因此采用有监督的2个评价指标平均F1-Score和扩展的归一化互信息ENMI来评估算法性能.

F1-Score通常用来评价分类结果的好坏. 它综合考虑了精确率(precision)和召回率(recall),可以用来衡量检测到的社区与真实社区之间的差异. 给定一个检测到的社区C_f和真实社区C_t,F1-Score定义如下:

$$F_1\left(C_f,C_t\right)=2\frac{\text{Precision}\left(C_f,C_t\right)\times\text{Recall}\left(C_f,C_t\right)}{\text{Precision}\left(C_f,C_t\right)+\text{Recall}\left(C_f,C_t\right)} \tag{8}$$

其中,

$$\text{Precision}\left(C_f,C_t\right)=\frac{\left|C_f\cap C_t\right|}{C_f} \tag{9}$$

$$\text{Recall}\left(C_f,C_t\right)=\frac{\left|C_f\cap C_t\right|}{C_t} \tag{10}$$

社区发现算法将网络划分为K个社区,其中,$C_f=\{C_1,C_2,\cdots,C_K\}$,真实的社区划分$C_t=\{C_1,C_2,\cdots,C_L\}$,定义算法划分社区的$F$1-Score为

$$F_1\left(C_f,C_t\right)=\frac{1}{K}\sum_{k=1}^{K}\max_{1\leqslant l\leqslant L}F_1\left(C_k,C_l\right) \tag{11}$$

平均F1-Score为

$$\overline{F_1}\left(C_f,C_t\right)=\frac{1}{2}F_1\left(C_f,C_t\right)+\frac{1}{2}F_1\left(C_t,C_f\right) \tag{12}$$

平均F1-Score越大,说明算法的精度越高,检测到的社区与真实社区越相近.

ENMI是基于信息熵的评价指标,用于量化2个社区之间的相似程度. 取值[0,1],ENMI越大,表示发现的社区与真实社区越接近.

4.4 实验结果

4.4.1 人工合成网络实验结果

图2展示了SOCD算法和其他6种算法在不同规模合成网络上的实验结果,其中BigClam和Community-GAN算法计算复杂度较高,在节点为100万的网络上12小时内未运算出结果. 通过图2可以发现,随着网络规模的增加,算法的准确度呈下降趋势. 在网络规模从1 000增加到10 000时,准确度只是稍微下降,当规模增加到十万甚至一百万时,准确度有一个明显的下降. 这个结果与文献[16]的结论一致,算法的精度随着混淆系数以及网络规模的增大而下降. 这主要是由于使用的LFR基准图参数会造成分辨率极限和视野极限从而造成性能下降. 从图中还可以看出,在网络规模较小时,SOCD算法的平均F1-Score值要比传统的方法精度低,但两者相差并不悬殊. 随着网络规模逐渐增大,SOCD算法的平均F1-Score和传统的方法较为接近;这表明在大规模网络上,相较于传统算法,SOCD算法能取得近似的结果.

图3展示了SOCD算法在不同混淆系数下的实验结果,由图中可以看出,随着混淆系数μ不断增大,网络的社区结构愈发的不明显,SOCD的精度在下降. 当$\mu<0.4$时,算法精度缓慢下降,当$\mu>0.4$时,算法精度才开始有一个较大的下降;这说明了SOCD算法具有较强的鲁棒性. 综上,通过在合成网络上进行的一系列实验,

证明了SOCD算法的有效性和鲁棒性.

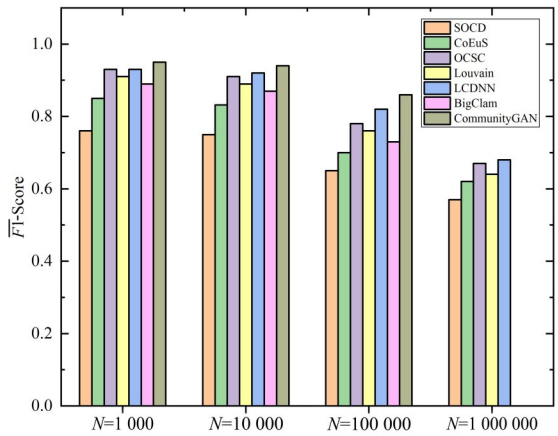

图2 不同规模的合成网络上的准确度比较

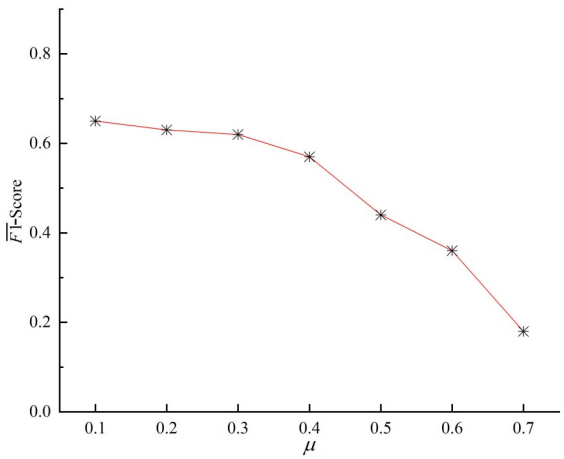

图3 不同μ值下SOCD算法的精度

4.4.2 真实网络上的实验结果

表3列出了SOCD算法和其他算法在6个大规模真实网络上运行的时间,表格中标"—"的数据对应执行时间超过12小时未返回结果的算法. 由表可知,SOCD算法处理100万条边只需要18 s,处理1.8亿条边大约需要7个小时,这是非常高效的. 当网络规模到达一定程度时,常规的方法就无法在有限的时间内返回结果,而本文算法可以很快速地将这些大规模网络进行处理,挖掘出其中的社区,这说明了本文算法在处理大规模网络时具有巨大的时间优势. 而基于流式的局部扩展算法CoEuS只在前4个数据集上运算出结果,无法在12个小时内在剩下的大规模网络中返回结果,这主要因为CoEuS的时间复杂度与社区个数以及连边数量成正比. 表2中6个数据集的社区个数和连边数量相差较大,特别是Orkut数据集的社区个数以及Friendster数据集的连边数量都远大于其他数据集,导致CoEuS没能在规定时间返回结果. 但相较于非流式的局部扩展算法LCDNN,采用流式的CoEuS算法极大地提升了运算速度,这表明了以流的方式对网络进行处理的高效性.

图4显示了7种算法在不同边的规模下的运行时间. 由图中可知,SOCD算法运行时间至少比非流式方法快一个数量级,且与边的规模成线性关系. 这是因为SOCD算法降低了时间复杂度,所以计算速度能够有数量级的提升.

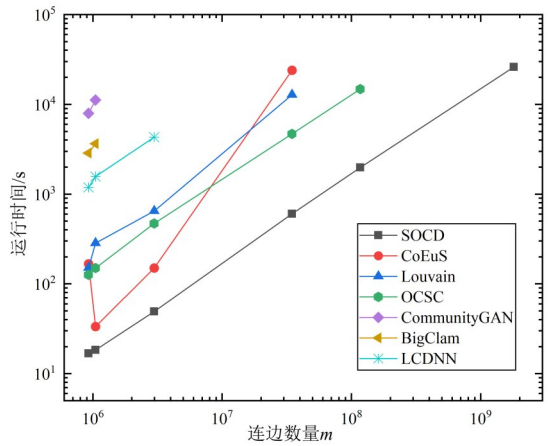

图4 不同边规模下算法的运行时间

表3 算法在大规模真实网络上的运行时间

网络	SOCD	OCSC	Louvain	CoEuS	LCDNN	BigClam	CommunityGAN
Amazon	16.8 s	2.1 min	2.5 min	2.8 min	19.8 min	47.6 min	2.2 h
DBLP	18.3 s	2.5 min	4.7 min	33.4 s	26.2 min	1.0 h	3.1 h
YouTube	49.2 s	7.9 min	10.8 min	2.5 min	1.2 h	—	—
LiveJournal	10.0 min	1.3 h	3.6 h	6.4 h	—	—	—
Orkut	33.0 min	4.1 h	—	—	—	—	—
Friendster	7.2 h	—	—	—	—	—	—

图5和图6分别展示了SOCD算法和其他算法在6个真实大规模网络上社区划分的平均F1-Score和ENMI. 从图中可以看出,在Amazon数据集上,CommunityGAN算法有着最好的实验效果,这是因为CommunityGAN利用生成对抗网络能够更好地提取网络的特征,但是模型预训练和训练时间较长,没办法处理较大规模的网络. 当网络的规模增大到1亿条边时,就只有SOCD算法和并行算法OCSC能够在有限的时间内返回结果.

并行算法OCSC设置了8个集群,但在12个小时内仍然没能成功处理Friendster数据集,这是因为分布式算法对效率的提升是有限的,当集群规模到达一定数量时,继续增加集群数量也无法提升效率,而SOCD算法可以高效的挖掘出Friendster网络中的社区结构. 从图中还可以发现,SOCD算法在YouTube,LiveJournal以及Orkut数据集上与分布式算法OCSC的准确率较为接近,在YouTube,LiveJournal数据集上还要优于Louvain算法. 而基于流式的局部算法CoEuS在大幅提高运算效率的情况下还可以取得与OCSC和LCDNN近似的结果,这都说明了对大规模网络进行流式处理的巨大优势. 综上所述,基于流式分析的SOCD算法具有线性的时间复杂度,在处理大规模网络上具有很大的时间优势,并且能够和非流式算法获得近似甚至更优的社区划分结果.

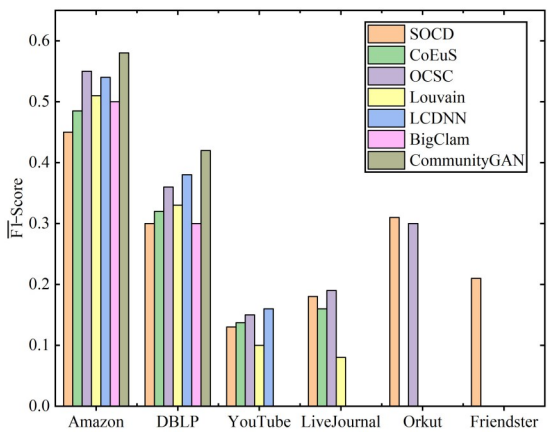

图5　算法在真实网络上的平均F1-Score

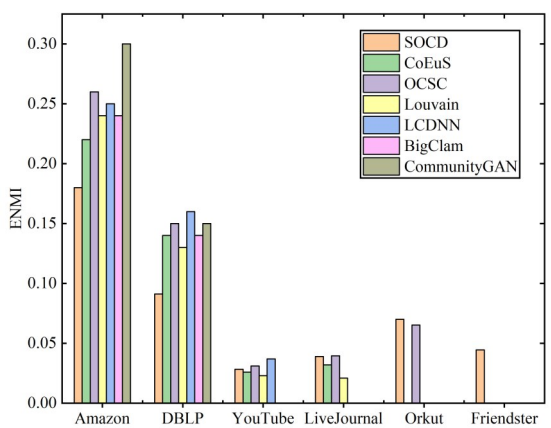

图6　算法在真实网络上的ENMI

4.5　参数D敏感性分析

在处理每一条边的过程中,根据阈值D来决定节点是否需要移动. 当D较小时,会将大社区划分为多个小社区,极端情况D=1时,每个社区仅包含2个节点;当D太大时,算法对社区间的边不敏感,会造成挖掘出的社区质量下降. 因此,D与网络中的度分布有着很大的关系,可以考虑从以下几个选项来选择此参数.

平均度:$D=d\_\mathrm{avg}$,网络中节点度的平均值,在实验中四舍五入取整.

中位度:$D=d\_\mathrm{med}$,网络中节点度的中位数.

众数度:$D=d\_\mathrm{mode}$,网络中节点度的众数.

定义$Q(D)=\dfrac{F_1(D)}{\max\left(F_1(D')\right)}$为参数对应社区的相对质量比,其中$D=\{d_\mathrm{avg},d_\mathrm{med},d_\mathrm{mode}\}$,$\max F_1(D')=\max\left(F_1(d_\mathrm{avg}),F_1(d_\mathrm{med}),F_1(d_\mathrm{mode})\right)$.

在2个真实网络以及2个合成网络上的实验结果如图7所示,由图中可知,除了在N=100 000网络上众数度对应的社区质量不是最高,在其他网络上都是最优. 因此,选择网络中节点度的众数作为阈值D可以获得更优的结果.

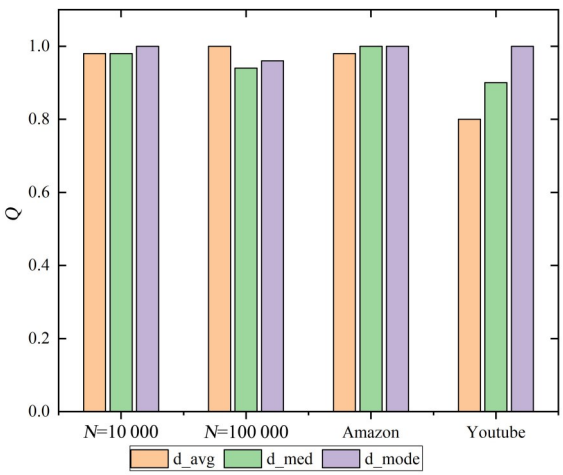

图7　D的不同选择对应的相对质量Q

5　结束语

针对网络数据规模庞大的问题,本文提出了一种基于流式分析的大规模网络重叠社区发现算法(SOCD). 该算法定义了节点对社区的贡献度来量化节点与社区之间的紧密程度,并利用社区间的连边数量等信息来共同指导节点的划分. 算法拥有线性的时间和空间复杂度,能够快速地挖掘出大规模网络中隐藏的社区结构. 在合成网络和真实网络上的实验结果表明了SOCD算法具有较强的鲁棒性和较高的运算效率,并且能够挖掘出网络中的重叠社区. 相较于非流式算法,SOCD要快10倍以上,这是由于算法降低了时间复杂度,运算时间与边的规模成线性关系,所以能够处理大规模甚至超大规模的网络;并且SOCD算法可以和非流式算法获得相近甚至更优的社区划分. 可以看到,网络的规模越大,SOCD算法的优势就越显著.

参考文献

[1] LIERDE H VAN, CHOW T W S, CHEN G R. Scalable spectral clustering for overlapping community detection in large-scale networks[J]. IEEE Transactions on Knowledge and Data Engineering, 2020, 32(4): 754-767.

[2] BLONDEL V D, GUILLAUME J L, LAMBIOTTE R, et al. Fast unfolding of communities in large networks[J]. Journal of Statistical Mechanics: Theory and Experiment, 2008, 2008(10): P10008.

[3] HE K, SHI P, BINDEL D, et al. Krylov subspace approximation for local community detection in large networks[J]. ACM Transactions on Knowledge Discovery from Data, 2019, 13(5): 52(1-30).

[4] LUO W J, LU N N, NI L, et al. Local community detection by the nearest nodes with greater centrality[J]. Information Sciences, 2020, 517: 377-392.

[5] 陈洁, 李锐, 赵姝, 等. 面向图表示社区检测的新型聚类覆盖算法[J]. 电子学报, 2020, 48(9): 1680-1687.
CHEN J, LI R, ZHAO S, et al. A new clustering cover algorithm based on graph representation for community detection[J]. Acta Electronica Sinica, 2020, 48(9): 1680-1687. (in Chinese)

[6] GHOSH S, HALAPPANAVAR M, TUMEO A, et al. Distributed Louvain algorithm for graph community detection [C]//2018 IEEE International Parallel and Distributed Processing Symposium. Vancouver, BC, Canada: IEEE, 2018: 885-895.

[7] GHAFFARI M, LATTANZI S, MITROVIC S. Improved parallel algorithms for density-based network clustering [C]//International Conference on Machine Learning. Long Beach, California, USA: PMLR, 2019: 2201-2210.

[8] LIAKOS P, PAPAKONSTANTINOPOULOU K, NTOULAS A, et al. Rapid detection of local communities in graph streams[J]. IEEE Transactions on Knowledge and Data Engineering, 2022, 34(5): 2375-2386.

[9] HOLLOCOU A, MAUDET J, BONALD T, et al. A linear streaming algorithm for community detection in very large networks[EB/OL]. (2017-03-08)[2020-12-11]. https://arxiv.org/abs/1703.02955.

[10] 李辉, 陈福才, 张建朋, 等. 一种基于流式分析的重叠社区发现方法及装置: CN202010780625.X[P]. 2020-12-22.

[11] YUN S, LELARGE M, PROUTIERE A. Streaming, memory limited algorithms for community detection[C]// NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems. Montreal Canada: MIT Press, 2014: 3167-3175.

[12] YANG J, LESKOVEC J. Overlapping community detection at scale: A nonnegative matrix factorization approach [C]//WSDM'13: Proceedings of the sixth ACM international conference on Web search and data mining. Rome, Italy: ACM, 2013: 587-596.

[13] 潘剑飞, 董一鸿, 陈华辉, 等. 基于结构紧密性的重叠社区发现算法[J]. 电子学报, 2019, 47(1): 145-152.
PAN J F, DONG Y H, CHEN H H, et al. The overlapping community discovery algorithm based on compact structure[J]. Acta Electronica Sinica, 2019, 47(1): 145-152. (in Chinese)

[14] JIA Y T, ZHANG Q Q, ZHANG W N, et al. Community-GAN: Community detection with generative adversarial nets[C]//WWW'19: The World Wide Web Conference. San Francisco, CA, USA: ACM, 2019: 784-794.

[15] LANCICHINETTI A, FORTUNATO S, RADICCHI F. Benchmark graphs for testing community detection algorithms[J]. Physical Review E, 2008, 78(4): 046110.

[16] EMMONS S, KOBOUROV S, GALLANT M, et al. Analysis of network clustering algorithms and cluster quality metrics at scale[J]. PLoS One, 2016, 11(7): e0159161.

作者简介

李 辉 男,1996年出生,湖北丹江口人.国家数字交换系统工程技术研究中心硕士生.主要研究方向为社区发现.
E-mail: lihui@alumni.hust.edu.cn

张建朋(通讯作者) 男,1988年出生,河北廊坊人.国家数字交换系统工程技术研究中心助理研究员.主要研究方向为大数据分析、图聚类.
E-mail: zjp@ndsc.com.cn

陈福才 男,1974年出生,江西南昌人.国家数字交换系统工程技术研究中心研究员.主要研究方向为网络安全.
E-mail: fucai0309@163.com