基于结构紧密性的重叠社区发现算法

潘剑飞[1,2],董一鸿[1],陈华辉[1],钱江波[1],戴明洋[2]
(1. 宁波大学信息科学与工程学院,浙江宁波 315211 ;2. 北京百度在线科技有限公司,北京 100084)

摘 要: 随着网络结构的不断扩大和日益复杂,传统的重叠社区发现算法已经不能有效地处理大规模网络数据,发现合理的社区结构.本文提出了顶点引力的概念,引入顶点凝聚度和社区凝聚度作为满足社区的外部结构稀疏性和社区内部结构紧密性的判定指标,构造了基于结构紧密性的重叠社区发现算法 OCSC.该算法经过预处理,核心子图划分以及核心社区的扩展三个步骤,能有效地发现重叠社区,通过对人工合成网络和真实网络结构的社区发现实验,运用 NMI 和 F1Score 等指标验证 OCSC 算法的合理性和优越性.

关键词: 社区发现; 重叠社区; 核心社区; 大规模网络结构; spark
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2019)01-0145-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2019.01.019

The Overlapping Community Discovery Algorithm Based on Compact Structure

PAN Jian-fei[1,2], DONG Yi-hong[1], CHEN Hua-hui[1], QIAN Jiang-bo[1], DAI Ming-yang[2]
(1. *Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo, Zhejiang* 315211, *China*;
2. *Baidu Online Technology Co. Ltd.*, *Beijing* 100084, *China*)

Abstract: With the continuous expansion and complexity of network structure, the traditional overlapping community detection algorithm can not effectively discover reasonable community structure in large-scale network structure. Based on the concept of vertex gravity proposed in this paper, we introduce vertex cohesion and community cohesion as indexes for community structure-close internal structure and sparse external structure, and then put forward overlapping community structure algorithm OCSC. The steps of OCSC algorithm include pre-processing, core sub-mapping and core community expansion. Finally, NMI and F1Score confirm the rationality and superiority of OCSC algorithm by experimentation on synthetic and real network structures.

Key words: community discovery; overlapping community; core community; large-scale network structure; spark

1 引言

存在于现代社会的各种复杂的网络结构,都呈现出一种重要的社区结构的特性—内部结构紧密,外部结构稀疏[1].随着互联网的发展和网络结构的扩张,网络内部的顶点数量不断增加,如 Facebook、腾讯和新浪微博等用户量已达到百亿级别,京东,阿里的网络用户购物商品构成的网络结构顶点也达到亿级别,发现这种庞大网络的内在社区结构对建立用户画像、进行商品推荐等都有一定指引作用.传统的社区发现方法单纯考察顶点之间的连接边,文献[2~4]将整个结构作为一个整体社区,文献[5]只能发现内部完全连接的团.如何保证社区内部结构的紧密性的同时,确保社区结构的可扩展性,文献[6]用三角形作为社区结构判定和社区扩展的最小单元,但该算法只能发现非重叠社区结构,无法进行重叠社区的识别.

为了解决现存算法不能同时保证社区内部结构的紧密性和社区结构的可扩展性的重叠社区识别问题,提出了基于结构紧密性的重叠社区发现算法(Overlap Compact Structure Community detection algorithm, OCSC),主要贡献包括:

1. 针对社区形成的三角形结构对顶点的牵引力,提出顶点引力的概念,引入顶点凝聚度和社区凝聚度作为社区外部结构稀疏性和内部结构紧密性的判定指标;

2. 提出基于结构紧密性的重叠社区发现算法 OCSC;

3. 基于单机环境和 spark 分布式平台实现 OCSC

收稿日期:2018-01-29;修回日期:2018-08-01;责任编辑:李勇锋
基金项目:国家自然科学基金(No. 61572266, No. 61472194);浙江省自然科学基金(No. LY16F020003);宁波市自然科学基金(No. 2017A610114)

重叠社区发现算法. 通过对人工合成数据集和真实数据集运用 NMI 和 F1Score 等各项指标判定社区发现算法的合理和优越性.

2 相关工作

面对大型复杂的网络社区发现的需求,如何找到网络合理划分的精确解是一个 NP 难问题. Newman 等人提出的 Fast-Newman 算法[2]用模块度来衡量社区划分的优劣,Zhang 等[7]对模块化计算进行了改进,大大提高了社区识别的效率. 近期 Prat-P[6]从社区结构合理性出发,提出用三角形作为社区结构判定和社区扩展的最小单元的 SCD 算法.

对于重叠社区的研究,Gregory 等人提出了基于 G-N 算法的 CONGA 算法[8]对重叠顶点进行分裂,能进行社区的重叠发现,但复杂度高,不适合大规模复杂网络的社区发现. Gregory 等人和 JR 等人将非重叠的标签传播算法改造为重叠社区发现算法,分别提出了 COPRA 算法[9]和 SLPA 算法[3],但是 COPRA 算法鲁棒性差,SLPA 算法需根据不同的网络结构设置不同的概率准则和参数才能选取相应的标签,实用性不强. Gopalan 等人运用混合概率模型的贝叶斯思想提出 SVINET 算法[10],通过计算后验分布来估计社区的结构,同时使用平均场变分推断随机优化算法来计算后验分布来确定社区的划分. Adamcsek 等人提出的 Cfinder 算法[5],社区内部边密度高容易形成派系,通过发现派系然后融合进行重叠社区发现. Ming 等人[11]基于遗传算法提出从边的聚类的 Meme-Link 算法,运用模块度衡量社区的质量并在带权线性图上运用遗传算法进行优化,最终通过基因编码将边社区转为顶点社区.

3 基本概念

表 1 为本文所用的符号与对应的描述.

表 1 符号表

符号	描述
G_{cr}	核心顶点构成的子图
V_{cr}	核心顶点的集合
E_{cr}	核心顶点间组成边的集合
C	社区
C_{cr}	核心社区
$pt_C(x)$	社区 C 内顶点成员 x 的凝聚度
$pt(C)$	社区 C 的凝聚度
$PR(x)$	顶点 x 的 PageRank 值
NT_x	与 x 顶点形成三角形邻接顶点
$d_{tri}(x)$	与 x 顶点能形成三角形的邻接度
$\mathrm{vt}_C(x)$	x 顶点与社区 C 至少一个三角形的顶点数
β	限制核心顶点的数量的阈值

定义 1 核心顶点:图 $G=(V,E)$ 中,核心顶点集 $V_{cr}=\{\forall v_x\in V, PR(v_x)\geq\beta\}$,其中 $PR(v_x)$ 表示顶点 v_x 的 PageRank 值. 即所有 PageRank 值大于阈值 β 的顶点构成核心顶点.

定义 2 核心子图:图 $G=(V,E)$,$e\in E$,e 两端点 v_1,v_2,$E_{cr}=\{e\in E, v_1\in V_{cr}, v_2\in V_{cr}\}$,则 $G_{cr}=(V_{cr},E_{cr})$,$G_{cr}\subset G$ 为核心子图,核心子图是核心顶点的导出子图.

定义 3 顶点引力 $PRD(v)$:图 $G=(V,E)$ 中,$v\in V$,顶点引力 $PRD(v)=\dfrac{PR(v)}{d_{tri}(v)}$.

定义 4 顶点凝聚度 $pt_c(x)$:图 $G=(V,E)$,$v\in V$,C 为社区,顶点 x 加入社区 C 则需计算顶点 x 的凝聚度.

$$pt_C(x)=\begin{cases}0, & \text{if } vt_C(x)=0\\ \underbrace{\dfrac{\sum\limits_{y\in(NT_x\cap C)}PRD(y)}{\sum\limits_{y\in NT_x}PRD(y)}}_{\text{顶点归属关系}}\cdot\underbrace{\dfrac{vt_C(x)}{|C|}}_{\text{社区内部结构}}, & \text{if } vt_C(x)\neq 0\end{cases}\tag{1}$$

其中,$vt_C(x)$ 表示 x 顶点与社区 C 内至少能构成一个三角形的顶点数. $y\in NT_x$ 表示 y 属于与 x 顶点能形成三角形的 x 顶点的邻接顶点.

定义 5 社区凝聚度 $PT(C)$:社区 C 凝聚度 $PT(C)$ 可由社区内各个顶点成员 x 的凝聚度 $pt_C(x)$ 的均值表示,即 $PT(C)=\dfrac{1}{|C|}\sum\limits_{x\in C}pt_C(x)$.

社区的凝聚度 $PT(C)$ 强的社区主要包括两个特点:1. 社区内点与社区外的点结构孤立;2. 社区内的点有强的链接性. 社区的凝聚度 $PT(C)$ 的值越大,说明该社区的内部连接越紧密且外部结构越稀疏;反之,社区的凝聚度 $PT(C)$ 的值越小,说明该社区的内部连接越松散或社区内外结构不能合理的分离.

模块度 M 值是衡量网络中社区稳定度的常用指标:

$$M=\frac{I}{E}-\left(\frac{2I+O}{2E}\right)^2\tag{2}$$

其中 I 表示两个端点均在同一社区中的边的数目,O 表示其中一个端点在社区中,而另一个端点不在社区中的边的数目,E 表示所有社区内边的数目.

定义 6 核心社区:通过模块度 M 值划分核心子图 $G_{cr}=(V_{cr},E_{cr})$,当 M 达到最大值时得到的核心顶点集构成核心社区.

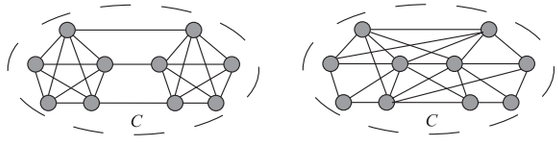
定义 7 扩展社区:从核心社区 C_{cr} 出发通过社区凝聚度指标不断扩展,使社区凝聚度值达到最大,得到的顶点集合为扩展社区.

4 社区凝聚度特性

一个好的社区发现算法应该具备两个方面:社区外部结构的稀疏性和社区内部结构的紧密性. 提出的社区凝聚度衡量指标 $PT(C)$ 具有以下特性.

4.1 特性 1:内部结构紧密

很多算法[2~4]将社区内的边同等对待,没有考虑该边的引入是否影响社区内的结构,使社区内结构不紧密. 顶点加入社区时,能否保证社区内部结构是衡量社区好坏的重要标准. 因此不仅仅考虑社区内边的数量,也要考虑社区内边的结构化程度. 图 1 显示了两个具有相同顶点数和边数的社区所对应的凝聚度值.



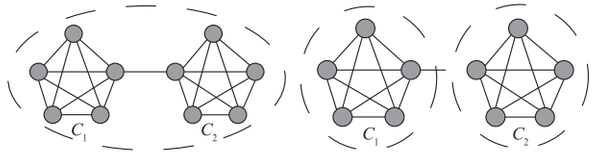
(a) $PT(C)=0.400$　　(b) $PT(C)=0.460$

图1　内部结构与社区的凝聚度值的关系

图 1(b) 的社区结构内部更紧密,社区凝聚度值也更大,说明社区内部结构越紧密社区凝聚度值越大.

4.2 特征 2:社区内不存在桥接

独立的社区 C_1,C_2 通过一条边连接称之为桥接. 桥接对于社区而言是弱连接,对社区内部结构性有很大影响. 优越的社区发现指标,社区内不应该存在桥接. 图 2 显示了社区内是否桥接与社区的凝聚度之间的关系.



(a) $PT(C_1 \cup C_2)=0.400$　　(b) $PT(C_1)=PT(C_2)=0.800$

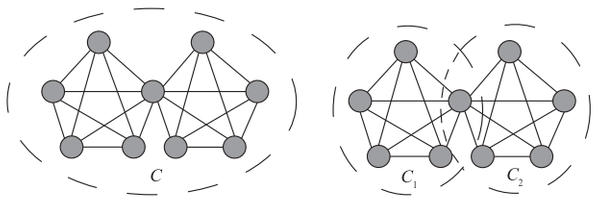
图2　社区内是否存在桥接与社区的凝聚度值的关系

定理 1　C_1,C_2 为图 G 内两个社区:若 C_1 和 C_2 是两个独立社区,若两个社区间存在桥接,则 $PT(C_1) > PT(C_1 \cup C_2)$,$PT(C_2) > PT(C_1 \cup C_2)$(证明略)

定理 1 说明社区内不应该存在桥接现象,反之,存在桥接的两个社区不应该合并.

4.3 特征 3:切点

切点表示两个社区之间通过一个节点相连,是一个弱连接,因为两个社区之间仅通过切点连接,两者不存在相互结构连接性. 如果针对社区内部结构来考虑,可将切点分配给两个不同的社区.



(a) $PT(C)=0.494$　　(b) $PT(C_1)=PT(C_2)=0.800$

图3　社区切点分配与社区的凝聚度值的关系

图 3(b) 的划分比图 3(a) 更具合理性,对应的社区凝聚度,图 3(b) 大于图 3(a),因此社区凝聚度能有效的说明社区结构的合理性.

通过以上的三个特性可知,运用社区凝聚度既能保证社区内部结构的合理性,又能保证社区的可扩展性,最大限度地发现合理的社区结构,因此算法考虑如何扩展使各个社区的凝聚度达到最大.

5 OCSC 算法描述

基于核心社区的社区结构增量的社区发现算法(OCSC),主要分为预处理,核心子图划分和核心社区的扩展三个阶段.

5.1 预处理过程

预处理过程包括剪枝和核心顶点的选择.

(1)剪枝. 根据社区凝聚度的特性 2 所指出的"社区内不存在桥接"的特性,可首先去除整个网络中未形成三角形的边,大大提高社区发现的效率.

(2)核心顶点的选择. 通过 PageRank 值确定网络内顶点的重要程度,同时计算每个顶点对邻接点的引力后,选择合适的 γ 值以确定阈值 β,得到一组核心顶点.

5.2 核心子图划分

对网络的顶点和边过滤只留下核心顶点和连接边,构成的核心子图,对核心子图用式(2)的 M 值进一步划分形成多个核心社区.

1. 初始化,每个核心点表示独立的不同的核心社区;

2. 对于每个核心社区的邻接,通过 M 值的增益判定是否合并到核心社区;

3. 重复第二步,直到不发生合并,核心子图划分完成.

核心子图划分的实现算法如下所示:

算法 1　核心子图划分

输入:整个核心子图 subgraph

输出:核心社区的集合

```
1.  var newGraph = InitGraph(subgraph).cache()/*初始化图操作,
    每个核心顶点表示为不同的核心社区,获取核心社区度信息*/
2.  While(updated >0)/*循环到最终社区划分不发生变化*/
3.  Val vertexRdd = newGraph.mapReduceTriplets
    (edgeFunc._ + + _).cache()/*获取每个核心顶点的邻接顶点
    的信息,edgeFunc 方法映射每个核心顶点邻接顶点*/
4.  val idComm = vertexRdd.map{case(vid,vArr) = >(vid,modulari-
    ty(vArr))}.cache()/*对于每个核心顶点的邻接顶点数组进行
    Modularity 的增益判定,将其分配到最合适的核心社区*/
5.  val upMessage = getUpdateMessage(idComm)
6.  newGraph = newGraph.joinVertices(upMessage) /*根据第三步
    核心顶点社区的分配,更新核心子图中顶点信息*/
7.  updated = newGraph.vertices.filter(_._2.changed).count /*当前
```

划分顶点变化的数量 */
8. newGraph. vertices. groupby(idComm)/ * 输出核心社区的集合 */

5.3 核心社区的扩展

核心社区的扩展,通过判定顶点 v 是否被当前状态的社区 C 接纳,需要计算新的社区凝聚度是否增大:

$$PT(C\cup\{v\})-PT(C)=\frac{1}{|C\cup\{v\}|}\cdot\sum_{x\in C\cup\{v\}}pt_{C\cup\{v\}}(x)-\frac{1}{|C|}\cdot\sum_{x\in C}pt_C(x)$$

当 $PT(C\cup\{v\})-PT(C)\geq 0$ 时将顶点 v 添加进社区,维持社区 C 的凝聚度值不断增长,使社区 C 的凝聚度值增大. 核心顶点的社区扩展,需要重新计算社区内每个点 x 的 $pt_C(x)$ 值才能得到社区 $PT(C)$ 值的变化,时间复杂度将达到 $O(d*n)^2=O(e^2)$. 为了降低算法的复杂度,引入定理 2.

定理 2 如图 4,顶点 v 为判定顶点,社区 C 内部 F 的顶点与顶点 v 邻接,社区内 G 的顶点与顶点 v 不邻接,

$r=|C|$,则 $PT(C\cup\{v\})-PT(C)$

$$=\frac{1}{(r+1)^2}\cdot\sum_{x\in F}\frac{1}{\sum_{y\in NT_x}PRD(y)}\cdot\Big(\Big(\sum_{y\in(NT_x\cap C)}PRD(y)+PRD(v)\Big)\cdot\delta(x)+PRD(v)\cdot \mathrm{vt}_C(x)\Big)-\sum_{x\in C}\frac{1}{\sum_{y\in NT_x}PRD(y)}\cdot\Big(\sum_{y\in(NT_x\cap C)}PRD(y)\Big)\cdot \mathrm{vt}_C(x)\cdot\Big(\frac{2}{r}+\frac{1}{r^2}\Big)+\frac{1}{(r+1)^2}\cdot\frac{\Big(\sum_{y\in(NT_x\cap C)}PRD(y)\Big)\cdot \mathrm{vt}_C(v)}{\sum_{y\in NT_x}PRD(y)}\quad(3)$$

其中

$$\delta(x)=\begin{cases}0, & x\text{ 与社区 }C\text{ 内的点不构成三角形}\\ 1, & x\text{ 与社区 }C\text{ 内的点构成三角形}\end{cases}$$

(证明略)

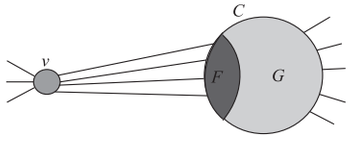


图4 顶点v加入社区C的过程

5.4 时间复杂度分析

OCSC 算法主要分为预处理,核心子图划分和核心社区的扩展三个阶段. 预处理的时间复杂度为 $O(e)$;核心子图划分是通过每个核心顶点的邻接判定顶点的归属,时间复杂度为 $O(\bar{d}_{cr}*n_{cr})\ll O(e)$,$\bar{d}_{cr}$表示核心子图的平均度,$n_{cr}$为核心子图的顶点数;核心社区扩展的主要开销为式(3),复杂度为 $O(n\bar{d})=O(e)$,n 为网络顶点数. 因此,算法的总体时间复杂度为 $O(e)$.

6 实验

6.1 实验环境描述

整个实验过程运用了 Master(1 台,CPU 为 8 Core 2.4,内存 25GB)和 Works(10 台,CPU 为 8 Core 2.4,内存 80GB),集群内部软件配置 HDFS 和 Yarn,基于 Hadoop 版本 2.6.5,Spark 版本为 1.6.3,其运行模式为 Spark On Yarn,driver 和 executor 的内存为 6GB.

6.1.1 人工合成和真实数据集

人工合成数据由 LFR(Lancichinetti Fortunato Radicchi)基准程序随机生成网络图[12],O_m 表示重叠的顶点应该属于几个社区结构,其分别设置为 3 到 9 进行实验. 随着 O_m 的增大,网络结构中的社区识别难度也随之增大. 参数设置如表 2.

表 2 人工合成数据生成参数

顶点数	平均度	最大度	重叠顶点数	混淆系数
5000	10	50	500	0.3
50000	10	50	5000	0.3

真实数据为斯坦福大学(SNAP)[13]的三个重叠社区的网络数据集:

表 3 原始数据集描述

属性 \ 数据集	Amazon	DBLP	Live Journal
Num vertices	334,863	317,080	3,997,962
Num edges	925,872	1,049,866	34,681,189
Communities	151,037	13,477	287,512

数据集的提供方同时还提供了对应网络的标准的高质量的 5000 个真实社区. 对这 5000 个社区处理:

1. 对于这 5000 个标准的社区删除重复的社区和社区大小小于 3 的社区,依据内部边密度进行排序,然后删除边密度后四分之一分位数的社区.

2. 然后根据步骤 1 获取的社区对原本的整个网络数据进行处理,删除不存在边和顶点.

最终将处理完的网络和社区作为社区发现的评估网络和标准,处理后的网络结构表 4 所示.

表 4 处理后数据集描述

属性 \ 数据集	U_ Amazon	U_DBLP	U_ Live Journal
Num vertices	8275	26,956	44,093
Num edges	22,231	88,742	871,409
Max comms per node	4	8	13
Min comm size	3	6	3
Max comm size	27	38	407
Min density	0.324	0.489	0.667

6.1.2 对比算法

为了合理的比较和评估 OCSC 算法的性能,实验选用单机算法 SLPA[3]、SVINET[10]、Cfinder[5]、Nise[14]和基于 spark 实现的 iSLPA[15]和 BigClam[16]算法,分别对人工合成数据集,表 5 和表 4 数据集进行对比:

1. SLPA:广播-监听标签传递算法. iSLPA 是对 SLPA 算法的改进,基于 spark 实现的高效分布式的算法.

2. SVINET:混合概率模型贝叶斯重叠社区发现算法.

3. Cfinder:基于派系过滤的社区发现算法.

4. BigClam:基于非负矩阵分解的社区发现算法.

5. Nise:种子传播算法.

6.1.3 性能评估指标

为了评估社区发现算法得到的社区结构是否合理,运用归一化互信息(NMI)和平均 F1Score 值($\overline{F_1}$)作为衡量社区质量的判定标准.

(1)归一化互信息 NMI 评估社区发现算法社区发现的结构和真实的社区结构之间的差异程度.

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

其中 X 表示通过社区发现算法发现的社区结构,Y 表示真正社区结构,$I(X,Y)$ 表示 X,Y 间互信息,$H(X)$,$H(Y)$表示 X,Y 的熵.

(2)平均 F1Score 值($\overline{F_1}$)是网络中社区结构的 F1Score 值的均值,F1Score 值可表示为集合 A 和集合 B 之间准确率和召回率的协和平均值. 其中 $p(A,B)$为准确率,$r(A,B)$为召回率.

$$\mathrm{F1Score}(A,B)=2p(A,B)r(A,B)/(p(A,B)+r(A,B))$$

6.2 PT 指标合理性

对表 3 中 DBLP 真实网络进行社区发现后,计算每个社区的 PT, Conductance 和 Average edge density 的值[6],根据社区的 PT 值进行排序,并切分为 20 等分后,画出箱线图,如图 5 所示.

Conductance 表示传导性,可以形象的描述社区外的稀疏度,如式(4),其中 $N(x)$表示 x 顶点的邻接顶点集.

$$\frac{\sum_{x\in C}|N(x)\cap(G\backslash C)|}{\sum_{x\in C}|N(x)|} \tag{4}$$

Average edge density 表示社区内边的平均密度,可以形象的描述社区内的紧密程度,可表示为式(5)所示:

$$\frac{1}{|C|}\sum_{x\in C}\frac{|N(x)\cap C|}{|C|-1} \tag{5}$$

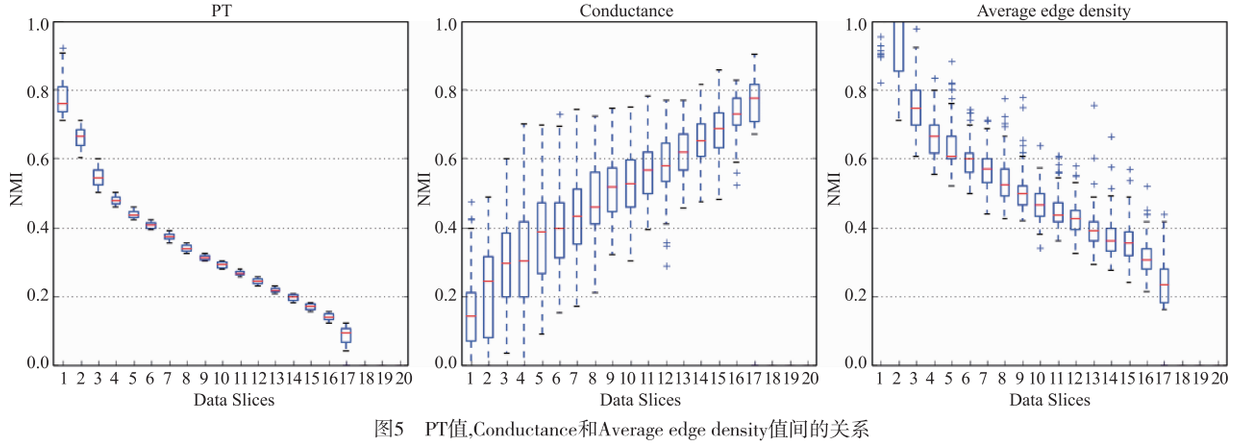


图5　PT值,Conductance和Average edge density值间的关系

图 5 显示,随着社区 PT 的降低,社区外部结构稀疏性越高,社区内部结构越不紧密,进一步通过实验证明 PT 指标用于社区发现的合理性.

6.3 OCSC 算法性能评估

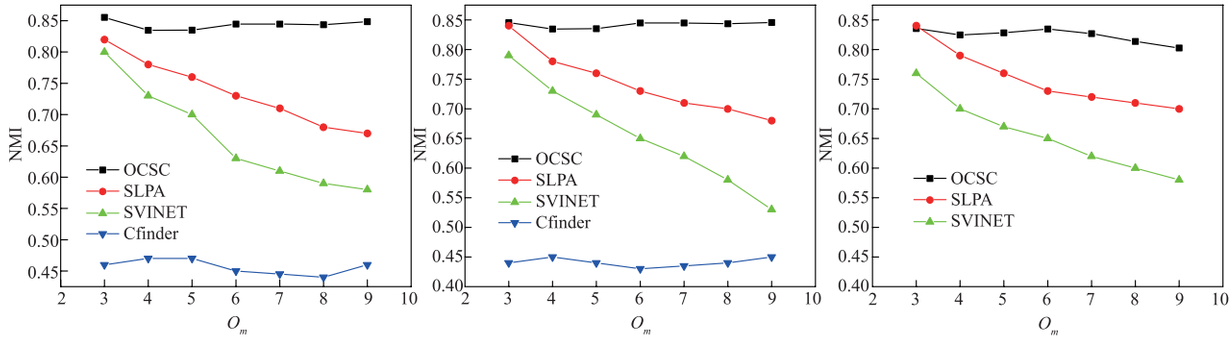
6.3.1 人工合成数据集实验

随着 O_m 的变化,比较 OCSC 算法与 SLPA,SVINET 和 Cfinder 的 NMI 指标的变化如图 6 所示,本文的 OCSC 算法相比其他的算法整体上处于较大的优势,波动程度上也较平稳,说明本文的算法能很好的克服重叠社区识别的困难,并保持高的社区识别准确率,图 6(c)中 Cfinder 算法在 4 小时内未能跑出结果.

表 5 显示超参数 β 的取值与 NMI 的关系. 实验运用模拟退火策略对超参数 β 进行启发式搜索. 无论 β 的初始值如何选取,最终搜索过程都停止在 NMI 最优解对应的 β 值,说明该策略能跳出局部最优达到全局最优解.

表 5　超参数 β 和 NMI 值的关系

Om \ β	0.4	0.5	0.6	0.7	0.8	0.9
3	0.79	0.82	0.835	0.80	0.815	0.80
4	0.809	0.813	0.795	0.792	0.79	0.785
5	0.793	0.795	0.793	0.79	0.835	0.80
6	0.823	0.832	0.825	0.820	0.822	0.815
7	0.79	0.80	0.821	0.82	0.832	0.829
8	0.80	0.81	0.816	0.831	0.842	0.82



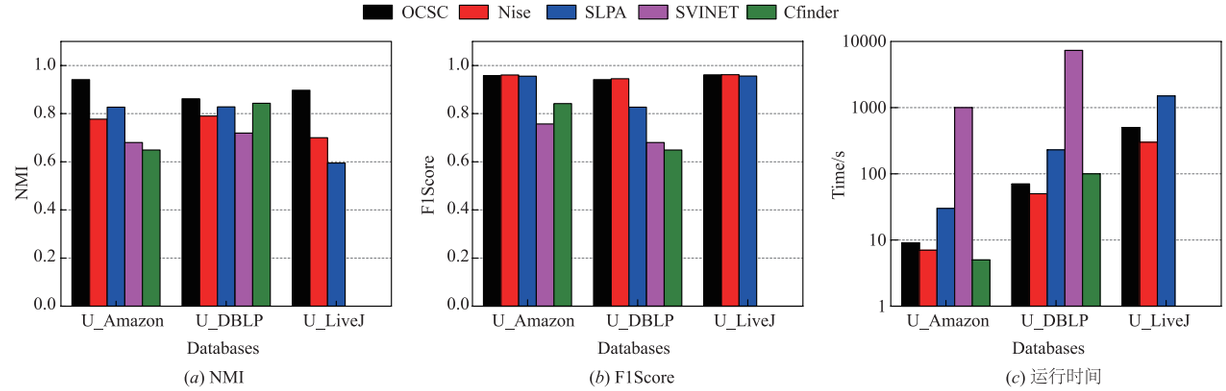
(*a*) 数据集G1 (*b*) 数据集G2 (*c*) 数据集G3

图6 合成数据集下算法性能比较

6.3.2 单机环境下性能分析

单机环境下真实数据性能分析主要对处理后的数据集(U_ Amazon,U_DBLP 和 U_ Live Journal)和原数据集(Amazon,DBLP 和 Live Journal)在 NMI,F1Score 和时间性能三个方面的比较.

针对处理后的数据集,核心点阈值 β 在 0.75 到 0.8 时社区发现的结构达到优解. 图 7(*a*)显示 NMI 指标判定上,OCSC 算法明显优于其他算法的结果;图 7(*b*)显示 F1Score 指标判定上,OCSC 算法和其他算法差不多. 对于 Live Journal' 数据集,算法 SVINET 和 Cfinder 运行时间超过了 4 个小时. 整体而言,通过 NMI 和 F1Score 两个指标的判定,OCSC 算法发现的社区更为合理. 图 7(*c*)为同等条件下单机运行的结果,时间复杂度同为 $O(e)$ 的算法 OCSC 比 SLPA 算法更具优势,因为 OCSC 算法进行预处理,并选取核心子图进行扩展,而 SLPA 选取随机种子进行扩展导致冗余;Cfinder 的算法的时间复杂度是边的指数级,当网络内边数较少时,效率上有优势,随着网络边数增加,执行的效率显著降低;SVINET 算法效率最差;Nise 的效率高于 OCSC 算法,这是由于 Nise 通过 K 来控制种子的范围,进行一次扩展,缺少迭代扩展的过程,只能发现局部社区,难以发现整体社区结构.



(*a*) NMI (*b*) F1Score (*c*) 运行时间

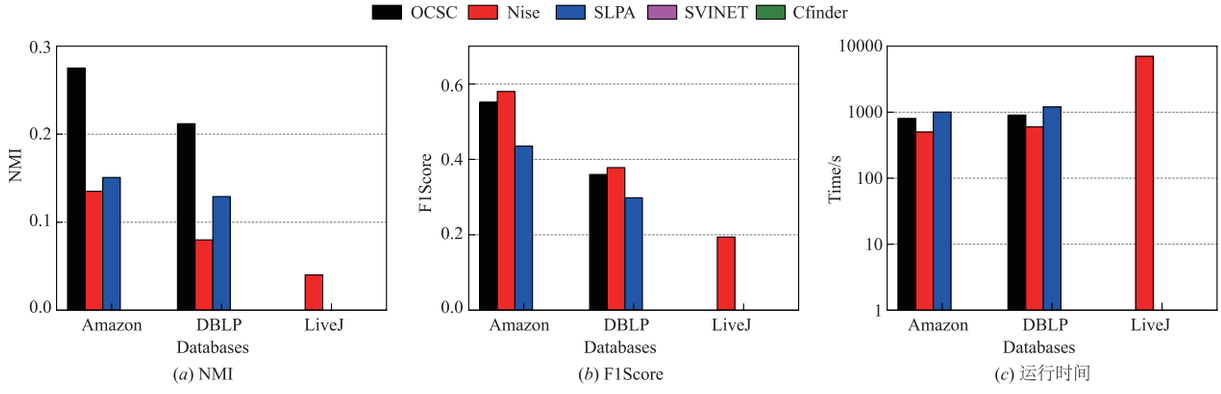
图7 处理后的数据集下算法性能比较

针对原始数据集,实验发现核心点阈值 β 在 0.35 到 0.5 时社区发现的结构达到优解. 图 8(*a*)显示 NMI 指标 OCSC 算法明显优于其他算法;图 8(*b*)显示 F1Score 指标 OCSC 算法略差于 Nise 算法,但是综合 NMI 指标和 F1Score,OCSC 算法具有优势. 未出结果的算法表示运行时间超过 4 小时,不具备衡量和比较的条件. 整体而言,通过 NMI 和 F1Score 两个指标的判定,OCSC 算法发现的社区更为合理. 从图 8(*c*)的运行时间效率上 Nise 高于 OCSC 算法是因为该算法只能发现局部社区结构.

6.3.3 分布式环境下性能分析

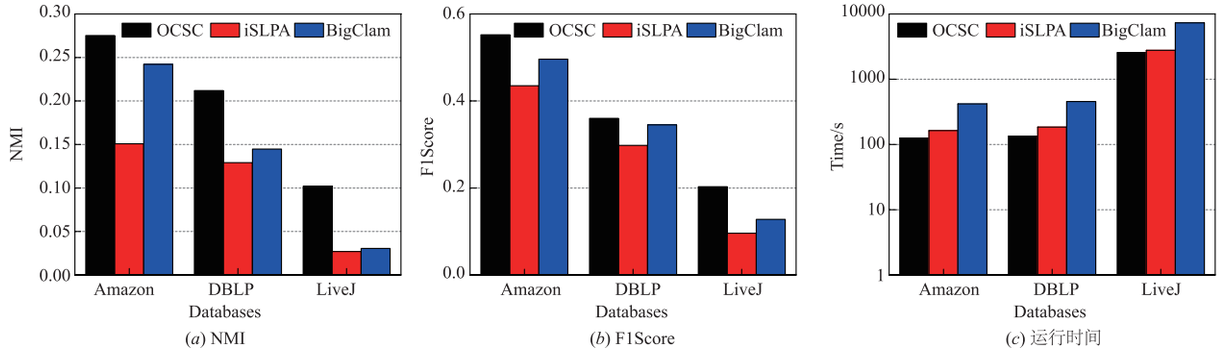
分布式环境下性能分析主要对原始数据集基于 spark 实现的 iSLPA[15] 和 BigClam[16] 算法与本文的 OCSC 算法的综合比较.

核心点阈值 β 控制在 0.35,图 9(*a*)和图 9(*b*)显示 NMI 指标和 F1Score 指标判定上,所发现的社区结构 OCSC 算法最优,BigClam 算法次之,iSLPA 算法最差. 三种算法时间复杂度都是 $O(e)$ 的算法. 图 9(*c*)显示运行效率上,OCSC 算法高于 iSLPA 算法 15% 左右,比 BigClam 算法高 200% 以上. iSLPA 算法发现的社区结构合



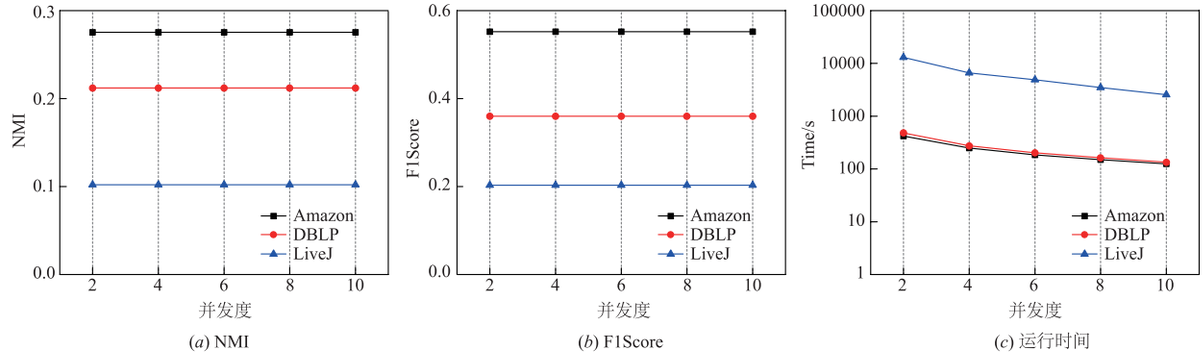
(*a*) NMI　(*b*) F1Score　(*c*) 运行时间

图8　原始数据集下算法性能比较



(*a*) NMI　(*b*) F1Score　(*c*) 运行时间

图9　分布式环境下原始数据集上的性能比较



(*a*) NMI　(*b*) F1Score　(*c*) 运行时间

图10　集群规模对算法的影响

理性要低于 BigClam 算法,但是在执行效率上要优越很多.

图 10 是对原始数据集下随集群规模变化社区结构的 NMI,F1Score 值和运行时间的结果. 从图 10(*a*)和图 10(*b*)看出,社区结构的 NMI 和 F1Score 值保持不变,表明集群规模的增大不影响算法社区发现的社区质量,图 10(*c*)表明随着集群规模的增大,算法运行时间越来越短,满足了分布式算法设计的理念,在不影响社区发现质量的同时提高算法执行的效率. 但是算法执行的时间和集群的规模并不呈线性,表明集群规模增加的同时也增加了部署节点到集群的时间,在算法执行时网络开销额外增加,说明并不是集群规模越大执行效果越好.

7　结束语

提出了社区凝聚度指标 PT 判定社区结构的合理性,基于该指标提出了重叠社区发现算法 OCSC,从理论和实验验证了算法的有效性. 实验中用 NMI,F1Score 等各项指标判定该算法发现的社区结构优于传统的社区发现算法,在单机和分布式环境下显示出很高的效率,能很好地运用于大规模复杂网络的社区结构分析. 进一步的研究将致力于动态增量的针对大规模图结构

数据的社区发现的更新,并适应实时性的要求和异质网络的社区分析.

参考文献

[1] Wang YZ, Jin XL, Cheng XQ. Network big data: Present and future [J]. Chinese Journal of Computers, 2013, 36(6): 1125 – 1138.

[2] Newman MEJ, Girvan M. Finding and evaluating community structure in networks[J]. Physcial Review E, 2004, 69(2): 026111.

[3] Xie JR, Szymanski BK, Liu XM. SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process [A]. Proc of the 2011 IEEE 11th Int'l Conf on Data Mining Workshops [C]. Washington: IEEE, 2011. 344 – 349.

[4] 王诗懿,董一鸿,李志超. 大规模复杂网络下重叠社区的识别[J]. 电子学报,2015,43(8):1575 – 1582.
Wang SY, Dong YH, Li ZC. Identification of overlapping communities on large scale complex networks [J]. Acta Electronica Sinica, 2015, 43(8): 1575 – 1582. (in Chinese)

[5] Adamcsek B, Palla G, Farkas IJ, Dere-nyi I, Vicsek T. C nder: locating cliques and overlapping modules in biological networks[J]. Bioinformatics, 2006, 22: 1021 – 1023.

[6] Prat-P, Rez A, Dominguez-Sal D, et al. Putthree and three together: triangle-driven community detection [J]. ACM Transactions on Knowledge Discovery from Data, 2016, 10(3): 22.

[7] Zhang XW, You HB, Zhu W, Qiao SJ, Li JW, Gutierrez LA, Zhang Z, Fan XN. Overlapping community identification approach in online social networks [J]. Physica A, 2015, 421: 233 – 428.

[8] Gregory S. An algorithm to find overlapping community structure in networks [A]. Proc of the European Conf on Principles of Data Mining and Knowledge Discovery [C]. Berlin, Heidelberg: Springer-Verlag, 2007. 91 – 102.

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

[10] Gopalan PK, Blei DM. Effcient discovery of overlapping communities in massive networks [J]. Proc Natl Acad Sci, 2013, 110(36): 14534 – 14539.

[11] Li M, Liu J. A link clustering based memetic algorithm for overlapping community detection [J]. Physica A Statistical Mechanics & Its Applications, 2018: 410 – 423.

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

[13] Altaf-Ul-Amin M, Shinbo Y, Mihara K, Kurokawa K, Kanaya S. Development and implementation of an algorithm for detection of protein complexes in large interaction networks[J]. BMC Bioinformatics, 2006, 7: 207.

[14] J Whang, D Gleich, I Dhillon. Overlapping community detection using neighborhood-inflated seed expansion [J]. IEEE Transactions on Knowledge and Data Engineering, 2015, 28(5): 1272 – 1284.

[15] Cai G Y. Study onlabel propagation based community detection algorithm for social semantic network [J]. Computer Science, 2013, 40(2): 53 – 57.

[16] Yang J, Leskovec J. Overlapping community detection at scale: a nonnegative matrix factorization approach [A]. ACM International Conference on Web Search and Data Mining[C]. ACM, 2013. 587 – 596.

作者简介

潘剑飞 男,1991 年出生,宁波大学信息科学与工程学院硕士生,百度算法工程师,主要研究方向为大数据、数据挖掘.

董一鸿(通信作者) 男,博士,1969 年 1 月生于浙江宁波,宁波大学教授,硕士生导师,中国计算机学会会员,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