基于结构稳定性校准的在线式社区识别

杨海陆 [1] 张健沛 [1] 杨 静 [1]

摘 要 本文探讨在线社会网络的社区识别问题, 重点研究网络演变特性对社区结构产生的影响. 首先基于节点的邻域倾向性提出社区稳定性的概念并给出稳定社区的快速识别算法, 然后设计了一种基于事件的社区稳定性校准算法以此识别新网络的社区结构. 由于算法的局部搜索策略, 该方法无需在新时间片段重复执行, 并且可以在无参数条件下识别加权网络中具有任意形状的社区结构. 在人工合成网络和真实网络上的实验结果验证了算法的可行性和有效性.

关键词 在线社会网络, 重叠社区识别, 社区稳定性, 校准策略

引用格式 杨海陆, 张健沛, 杨静. 基于结构稳定性校准的在线式社区识别. 自动化学报, 2014, **40**(10): 2151−2162

DOI 10.3724/SP.J.1004.2014.02151

Identifying Online Communities by Calibrating Structure Stability

YANG Hai-Lu[1] ZHANG Jian-Pei[1] YANG Jing[1]

Abstract This paper investigates into the problem of identifying communities in online social networks. It focuses on the effect of evolution behavior of network on the community structure. In the work, the notion of community stability and a high-speed algorithm for mining stable community are first proposed according to the tendency between nodes and its neighborhood. Then a community stability calibrating method is designed, which takes advantage of the evolution event of network to recognize the new snapshot communities. Thanks to the local search strategy of the algorithm, the new method does not require to execute repeatedly in new snapshot and is able to recognize community structures with arbitrary shapes under the condition of no input parameters. Finally, competitive experiments on both synthesized and real-world social networks demonstrate the effectiveness and efficiency of the proposed algorithm.

Key words Online social networks, overlapping community detection, community stability, calibrating strategy

Citation Yang Hai-Lu, Zhang Jian-Pei, Yang Jing. Identifying online communities by calibrating structure stability. *Acta Automatica Sinica*, 2014, **40**(10): 2151−2162

在线社会网络 (Online social networks, OSNs) 是一种具有演变特性的社会网络. 伴随着各种新兴服务的迅速普及, OSNs 成为人们日常生活中不可或缺的沟通工具. 微博、社交网络、即时通信服务、移动互联网等都是典型的在线社会网络. 社区识别是社会网络分析领域最为重要的研究内容之一. 在网络中, 稳定的社区结构能够刻画复杂系统中不同层次的结构特性并揭示出具有共同兴趣、共同背景的社会群体. 目前, 大多数社区识别方法将网络视为静态图进行研究, 然而在真实世界中网络和社区都具有强烈的演变特性.

研究社区的演化行为具有重要意义, 例如某些亚种群的疾病感染动力学特征可能对流行病的跟踪以及早期预防起到关键作用; 追踪论坛中的热门话题可以预测新技术和新潮流的演变趋势; 对大型公司中一些非正式群体的演变行为加以分析可能为公司的全球性决策提供技术保障. 目前, 面向动态社会网络的社区识别缺乏理论化、系统化、长期化的研究, 较有代表性技术和理论主要集中在文献 [1−13], 但这些方法大多受到以下 4 方面限制:

1) 无法处理加权网络. 某些算法对加权网络的处理方法是, 设定某一阈值并删除所有小于该阈值的链接. 实际上这种做法忽略了节点的局部信息, 例如网络中的富人俱乐部会使权重较小的节点变成孤立节点, 进而使整个网络失去现实意义.

2) 需要预先给定相关参数. 一些算法需要在执行前给出诸如社区个数、社区重叠度、社区内部节点数等参数的具体数值, 而这些在真实网络中通常难以预先得知.

3) 局限于某种特定的社区类型. 某些算法采用基于密度或 Clique 的方法挖掘社区结构, 这种结构

收稿日期 2013-11-18 录用日期 2014-02-20
Manuscript received November 18, 2013; accepted February 20, 2014
国家自然科学基金 (61073041, 61073043, 61202274, 61370083, 61402126), 高等学校博士学科点专项科研基金 (20112304110011, 20122304110012) 资助
Supported by National Natural Science Foundation of China (61073041, 61073043, 61202274, 61370083, 61402126), Specialized Research Fund for the Doctoral Program of Higher Education (20112304110011, 20122304110012)
本文责任编委 王飞跃
Recommended by Associate Editor WANG Fei-Yue
1. 哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001
1. College of Computer Science and Technology, Harbin Engineering University, Harbin 150001

约束导致算法无法识别星形社区. 本文将社区定义为一种稳定程度较高的局部子结构, 并认为星形社区由于叶分支对中心节点的唯一倾向性, 具有较高的稳定性.

4) 时间开销较大. 一些动态社区挖掘算法本质上是一种适应性的静态算法, 这类算法能够处理新网络中节点集和边集的变化, 但由于算法需要在不同时间片段重复执行, 因此效率较低.

针对上述 4 个问题, 本文提出 Adaptive mining stable community (AMSC) 将演化社区挖掘任务分为 2 个阶段. 如图 1 所示, 首先在第 1 阶段挖掘初始网络 G^0 中的社区结构; 在第 2 阶段, 不断对前一时刻的社区进行校准以此识别当前时间片段的社区结构. 由于算法的局部化校准策略, AMSC 在网络演化时无需重新运行算法, 因此具有较高的效率. 归纳而言, 本文的主要贡献在于:

1) 提出社区稳定性的概念以及稳定社区的快速挖掘算法. 社区的稳定性基于节点对其邻域的倾向程度, 由于避免了结构化约束, 稳定社区可以具有各种形状.

2) 提出在线式社区挖掘算法 AMSC. 通过对社区的稳定性进行校准, 算法能够快速地识别出新网络的社区结构, 并且可以对社区进行追踪.

3) 在真实数据集和人工数据集上进行了大量实验, 实验结果验证了算法的可行性和有效性.

本文第 1 节介绍相关研究; 第 2 节给出在线式社区识别问题的形式化定义以及稳定社区的挖掘方法; 第 3 节设计社区稳定性的校准方案并识别新网络社区; 第 4 节为实验与结果分析; 第 5 节总结全文并提出未来工作.

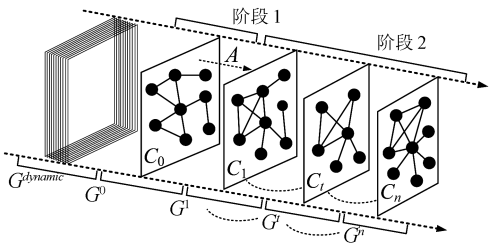


图 1 演化社区挖掘框架

Fig. 1 Framework of evolutionary community detection

1 相关研究

目前在社会网络社区挖掘方向已有大量的研究成果. 总体来讲, 可以将现有研究分为两大类, 即静态社区挖掘、动态社区挖掘. 静态网络上的社区挖掘算法深受各领域学者的广泛关注, Fortunato[14] 在 2010 年发表的综述性文章中对这些算法做过较为详细的比对和分析.

相比于静态社区挖掘, 动态社区挖掘尚处于起步阶段. 最早的研究由 Hopcroft 等[1] 于 2004 年提出, 文中采用一种层次化聚类方法, 分析了若干 NEC Citeneer 引文数据库的时间片段, 并尝试追踪社区随时间变化的演化过程. Asur 等[2] 分析不同时间片段的社区变化, 认识到网络演化过程中事件发生的必然性, 提出了分析节点与社区演化行为的特征框架. Sun 等[3] 提出一种基于互信息与熵函数的社区挖掘框架 Graphscope, 通过使用最小描述长度原则计算编码开销, 算法无需任何输入参数, 并且对流数据具有良好的匹配性. Palla 等[4] 对不同时段的网络分别进行社区挖掘, 并分析不同时刻社区之间的关联, 这种策略可以挖掘重叠社区, 但由于算法的重复运行, 因此在大规模网络上具有较高的时间开销.

上述研究在动态社区挖掘方面开展了大量开创性工作, 但研究范畴均属于离线式演化社区分析, 因此不具有实时性. 最近一些学者开始注意到在线式社区挖掘的应用价值, 较为代表性的研究有: 吴斌等[5] 在 Graphscope 框架的基础上提出一种基于 Clique 合并的半在线式社区挖掘方法, 该方法将若干平滑阶段的时间片段合并为一个近似图之后进行社区挖掘, 可以有效地降低时间开销, 但不可避免的导致了社区的信息损失. Lin 等[6] 提出动态网络重叠社区挖掘框架 Facetnet, 该方法可以收敛到一个局部最优解, 但是这种收敛速度通常很慢, 并且算法需要预先设定网络中的社区数量, 而这在现实中是无法得知的. Tang 等[7] 提出一种正则化多模聚类算法研究动态合作网络中的社区结构. Gauvin 等[8] 制定了一种非负张量分解技术, 可以实时性的识别动态重叠社区并跟踪社区的演变模式. Cazabet 等[9] 提出 iLCD (Local community discover) 算法, 通过在网络中添加链接并合并相似节点进而挖掘动态网络中的重叠社区, 但该模型无法处理节点或链接的添加与删除. 与之对应, Pérez-Suárez 等[10] 采用图论方法进行社区识别, 并实现了节点和链接的增量式处理. Lancichinetti 等[11] 提出 OSLOM (Order statistics local optimization method) 算法, 用于测试随机图模型的统计意义, 该算法通过计算社区邻接节点的归属性, 将满足条件的节点归纳进该社区. 林旺群等[12] 提出 D-SNCD (Dynamic social network community discovery) 算法, 将社区挖掘视为层次化社区树分支的更新过程. Dinh 等[13] 提出的 adaptive algorithm with approximation guarantee for identifying community structure (A^3CS) 算法被认为具有较高的在线式挖掘性能, 但由于算法采用了模块度优化策略, 因此存在分辨率极限. 与上述

在线式社区识别算法的对比工作将在文中第 4 节给出.

2　挖掘初始网络社区结构

2.1　挖掘稳定社区

目前, 尚无统一和通用的社区结构定义, 但在无权网络中一个普遍被认可的观点是社区内部链接总数远大于社区间链接总数. 借鉴该思想, 本文提出社区稳定性这一概念. 假定网络节点受到邻接节点的同质性吸引[15] (吸引程度用边权大小加以度量), 那么稳定状态的社区结构应满足社区内部节点具有较强的互吸引力, 而分处于不同社区的节点应尽可能不具有任何吸引力. 与 Newman 将模块度函数[16]扩展至加权网络的方法较为类似, 本文所定义的社区稳定性概念基于对加权网络所做的有向化处理, 其形式化定义由定义 1 给出.

定义 1. 用图 $G=(V,E,w)$ 表示加权社会网络, 其中 V 为节点集, E 为边集, $w:E\to\mathbf{R}^+$ 为每条边指派一个非负的权重, $\widetilde{C}=(C_1,C_2,\cdots,C_n)$ 是图 G 对应的社区结构, 用 $\theta_{u,v}$ 表示节点 u 对节点 v 的倾向程度, 如果对于任意 $C\neq C'\in\widetilde{C}$ 均满足:

$$\sum_{u,v\in C}(\theta_{u,v}+\theta_{v,u})\gg\sum_{u\in C}\sum_{v\in C'}(\theta_{u,v}+\theta_{v,u}) \tag{1}$$

则称社区 C 为稳定社区.

式 (1) 在退化为无权网络时同样强调社区的内外边比例. 在加权网络中, 节点受到不同程度的同质性吸引, 因此一旦某吸引力值超出使节点保持相对静止状态的 "临界值", 节点就会沿该方向发生相位迁移, 最终导致社区的形成. 由于在拓扑空间中无法采用欧式空间力的计算方法, 因此本文选择以节点的权重均值作为节点的移动阈值. 用 $\tau(u)$ 表示 u 的节点度, $\sigma(u)=\{i\mid(u,i)\in E\}$ 表示 u 的邻居集合, 则 u 的权重均值定义为:

$$\delta(u)=\frac{1}{\tau(u)}\sum_{v\in\sigma(u)}w(u,v) \tag{2}$$

对于任意节点 $i\in\sigma(u)$, 如果 $w(u,i)>\delta(u)$, 则称边 (u,v) 是以 u 为核心构成的局部环境中的强指向链接, 反之为弱指向链接. 因此可定义:

$$\theta_{u,v}=\begin{cases}1, & w(u,v)>\delta(u)\\ 0, & \text{其他}\end{cases} \tag{3}$$

如图 2 所示, 图左侧社区内部节点间具有较高的互吸引力, 而右侧社区由于叶子节点对中心节点的唯一倾向性, 因此均具有较高的稳定性. 结合定义 1, 加权社会网络中的社区识别问题就可转化成一种优化问题, 即

$$\begin{aligned}\min(\omega)=&\sum_{u,v\in C_a}(2-\theta_{u,v}-\theta_{v,u})+\\&\sum_{u\in C_a}\sum_{v\in C_b}(\theta_{u,v}+\theta_{v,u})\\ \text{s.t.}\quad & C_a\neq C_b\in C^*\in\Lambda(C)\end{aligned} \tag{4}$$

$\Lambda(C)$ 为网络中所有社区划分结果构成的集合, C^* 为 $\Lambda(C)$ 中的某一种社区划分方式. 式 (4) 前一项记录了相同社区中互倾向的不一致性. 假设节点 u, v 处于同一社区并具有互倾向性则 $\theta_{u,v}=\theta_{v,u}=1$, 此时不一致性惩罚为 $2-1-1=0$. 如果 $\theta_{u,v}$ 与 $\theta_{v,u}$ 不全为 1, 那么不一致惩罚就会出现大于 0 的情况. 同理, 式 (4) 后一项计算不同社区间互倾向的一致性惩罚, 并期望 $\theta_{u,v}+\theta_{v,u}=0$ 这类情况出现.

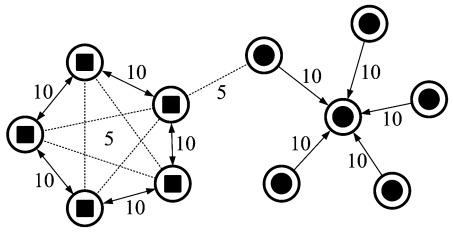


图 2　节点的邻域指向分析

Fig. 2　Neighborhood pointing analysis of nodes

对式 (4) 而言, 一种最为直观的求解方法是遍历所有的社区划分结果, 然后取使 ω 达到最小值的那个划分方式 C_i^*, 但这种方法的计算开销过于庞大, 例如当节点数为 $n=20$ 时就会存在着 2^{20} 种划分方式. 为此本文提出一种基于核心扩张的启发式策略将式 (4) 的求解过程分为两个阶段. 首先定义两种特殊的链接模式.

定义 2. 对于 $\forall(u,v)\in E$, 如果满足 $w(u,v)>\max\{\delta(u),\delta(v)\}$, 且有 $\tau(u)\geq 2$, $\tau(u)\geq 2$, $\sigma(u)\cap\sigma(v)\neq\varnothing$, 则称边 (u,v) 为核心链接.

核心链接满足优化条件 $\theta_{u,v}+\theta_{v,u}=2$, 因此以该链接为中心进行扩张可以保证所有具有互倾向性的链接都被归纳到同一社区. 定义 2 的后 3 个约束条件保证该链接不会成为星形社区的叶分支并具有可扩张性.

定义 3. 对于 $\forall u\in V$, $v\in\sigma(u)$, 令 A 表示网络的邻接矩阵, 则 u 的邻域倾向程度定义为

$$\varphi(u,v)=\sum_{i\in\sigma(u)\cap\sigma(v)}\theta_{u,i}+\sum_{i,j\in\sigma(u)\cap\sigma(v)}A_{i,j} \tag{5}$$

对于 $\forall i\in\sigma(u)$, 如果满足 $i=\arg\max_{i\in\sigma(u)}\varphi(u,i)$, 则称边 (u,i) 为以 u 为核心构成的局部环境中的准稳定链接.

某些节点没有邻接的核心链接, 因此需要判定其邻域的结构稳定性. 式 (5) 的前一项度量了假设以 (u,v) 为核心进行扩张, 模块能够归纳的倾向节点数. 最理想的情况是将 u 的倾向节点都包含进同一社区, 但这种做法通常会导致结构过于稀疏, 因此对该项进行惩罚. 式 (5) 的后一项度量扩张模块的内部链接程度, 由于核心链接与共有邻居节点之间没有关系缺失, 因此该项可以代表所生成模块的密度. 如果模块内含有较多的节点, 则该模块将被期望具有更多的内部链接, 并且随着节点数的增加模块密度 $\zeta \propto \binom{n}{2} \gg n$, 因此规模较小的模块如果具有较高的密度, 则稳定性很可能大于那些包含较多倾向节点但过于稀疏的群体. 式 (5) 同样作为不具任何指向性的链接的倾向性判别函数, 该情况中方程的前项结果为 0, 因此转为寻找局部密度最大的子结构.

定义 2 和定义 3 描述的两种链接模式使得各节点能够归纳进最具稳定性的局部社区之中, 即社区中存在大多数该节点的倾向节点. 在扩张过程中, 首先以这两种链接为核心, 然后选取链接端点的共有邻居进行合并, 即局部社区:

$$CL_{u,v} \equiv \{u,v\} \cup \sigma(u) \cap \sigma(v) \tag{6}$$

以此满足扩张节点对核心的直接相似性和结构相似性[17]. 算法 1 描述了上述扩张策略.

算法 1 (稳定社区识别算法).

步骤 1. 计算倾向性判别函数 θ.

步骤 2. 初始化稳定社区集合 $C_U^{(0)}$.

步骤 3. 随机选取节点 $u \in V$, 根据定义 2 和定义 3 判断 u 与邻居节点 $v \in \sigma(u)$ 的链接模式.

步骤 4. 根据式 (6) 对核心链接或准稳定链接进行扩张并生成稳定社区 C, 将 C 并入 $C_U^{(0)}$.

步骤 5. 从 V 中删除 u.

步骤 6. 重复步骤 3 ~ 5, 直到 V 为空集.

步骤 7. 输出稳定社区集合 $C_U^{(0)}$.

算法 1 结束后, 用参数 $s_{u,v}$ 对链接模式进行标记, 并规定当 (u,v) 为核心链接时, $s_{u,v}=2$; 当 (u,v) 为准稳定链接时, $s_{u,v}=1$. 如图 3 所示, S 矩阵中的冗余信息更少、结构更清晰, 因此可以直观地识别出重叠社区 $C_1=\{a,b,c\}$, $C_2=\{c,d,e\}$.

算法 1 的贪婪策略导致循环过程重复计算倾向性判别式 φ. 为了提高算法效率, 在步骤 1 首先对全局 θ 值进行计算. 在式 (2) 中, 节点的邻居数为 τ, 因此该步的计算复杂度为 $\mathrm{O}(\tau n) = \mathrm{O}((2m/n)\times n) = \mathrm{O}(m)$. 算法 1 的复杂度主要集中在步骤 3 的链接模式判断上, 参照步骤 3 ~ 5 的循环模式以及式 (2) 的计算方式, 核心链接的计算复杂度为 $\mathrm{O}(\tau^2 n)$. 准稳定链接需要计算式 (5), 由于 θ 值无需重新计算, 因此该项的计算复杂度为 $\mathrm{O}(\tau n)$. 综上所述, 算法 1 最终的时间复杂度为 $\mathrm{O}(m+\tau^2 n+\tau n)=\mathrm{O}(\tau m)$.

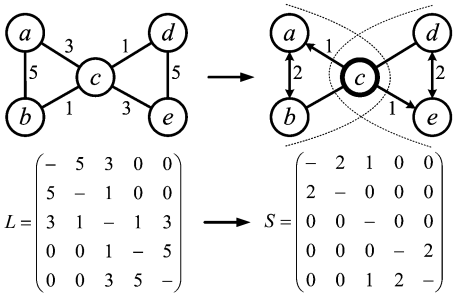


图 3 稳定社区识别实例

Fig. 3 Example of stable community detection

2.2 合并同源子社区

算法 1 结束后, 输入网络被描绘成由若干重叠社区组成的分块网络. 换句话讲, 这些仅以单独节点的倾向程度为稳定性判别标准的稳定社区可称之为网络中的基本社区. 某些基本社区之间可能共享有高度重叠的子结构或具有相同的节点集, 因此即使将其合并也不会破坏社区内部节点的稳定性. 为了实现这一目的, 下面给出同源社区的概念.

定义 4. 对于任意社区 $C_a, C_b \in C_U^{(0)}$, 令 $O_{ab}=C_a\cap C_b$ 表示社区间的重叠部分, $\bar{O}_x$ 表示 O_{ab} 相对于社区 C_x 的补集, 如果满足:

$$\left|\bar{O}_{a\vee b}\right| \le 1+\sum_{u\in C_a\vee C_b\setminus O_{ab}} \left(\max\nolimits_{v\in O_{ab}}\theta_{u,v}\right) \tag{7}$$

则称 C_a 与 C_b 为同源社区.

式 (7) 度量了合并后社区的分裂倾向, 如果社区非重叠部分节点倾向于重叠部分节点, 则合并后的社区仍具有内聚性, 进而合并操作不会破坏社区的稳定性. 反之, 如果重叠部分节点具有较高的外倾向性, 则社区面临分裂的风险. 例如图 3 中存在重叠社区 $C_1=\{a,b,c\}$, $C_2=\{c,d,e\}$, 根据式 (7) 有

$$\left|\bar{O}_{a\vee b}\right| = 2 > \left(1+\sum_{u\in C_a\vee C_b\setminus O_{ab}} \left(\max\nolimits_{v\in O_{ab}}\theta_{u,v}\right)\right) = 1+0=1$$

因此 C_1 和 C_2 不是同源社区. 假如将 C_1 和 C_2 合并, 则新社区 $C_3=\{a,b,c,d,e\}$ 会由于节点 c 的外指向性引发社区的分裂态势, 因此 C_3 的稳定性较低. 在同源社区的合并约束上, 本文借鉴了 Community finder (CFinder) 算法[18] 的合并策略, 认为至少存在 $n-1$ 个内指向节点即满足合并条件. 式 (7) 在退化为无权网络时需将 θ 替换为 s, 这是由于无权

网络的链接指向性被认为是均等的. 定义 4 的另一个重要作用是对星形社区的叶分支进行合并, 由于叶子节点对中心节点具有唯一倾向性, 因此星形社区可以通过同源性判别合并生成. 具体来讲, 满足同源社区合并条件的两个基本社区将被合并为新社区, 实现过程如算法 2 所示.

算法 2 (同源社区算法).

步骤 1. 合并 $C_U^{(0)}$ 中具有同节点集的基本社区.

步骤 2. 初始化社区集合 $C^{(0)}$.

步骤 3. 随机选取 $C_a \subset C_U^{(0)}$, 根据式 (7) 计算邻接社区 $C_b \subset \tau(C_a)$ 的同源性.

步骤 4. 将同源社区合并后并入 $C^{(0)}$, 将无同源性的基本社区并入 $C^{(0)}$.

步骤 5. 从 $C_U^{(0)}$ 中删除 C_a.

步骤 6. 重复步骤 3 ~ 5, 直到 $C_U^{(0)}$ 为空集.

步骤 7. 输出重叠社区集合 $C^{(0)}$.

算法 2 的时间复杂度为 $\mathrm{O}\left(\tau_U|C_U^{(0)}|\right)$, $|C_U^{(0)}|$ 为基本社区的总个数, τ_U 为社区的平均重叠数. 根据定理 1, 可知算法实际可在 $\mathrm{O}(\tau m)$ 复杂度内完成基本社区的合并任务. 因此静态社区挖掘算法 MSC 的总体时间复杂度为 $\mathrm{O}(\tau m + \tau m) = \mathrm{O}(\tau m)$.

定理 1. 在具有 n 个节点, m 条边, $|C_U^{(0)}|$ 个基本社区的复杂网络中, 使用算法 2 合并同源社区的时间复杂度为 $\mathrm{O}\left(\tau_U|C_U^{(0)}|\right) = \mathrm{O}(\tau m)$.

证明. 对于任意基本社区 $C_a \in C_U^{(0)}$, 由于局部社区具有重叠性, 因此可知 $|C_U^{(0)}| < \rho(C)|C_U^{(0)}| = \sum_{C_a \in C_U^{(0)}} |C_i|$, $\rho(C)$ 为局部社区的平均节点数. 根据 MSC 的全覆盖挖掘策略可得 $\sum_{C_a \in C_U^{(0)}} |C_a| = n + \sum_{a<b} |C_a \cap C_b|$ 并且由于算法选择节点 u 与 v 的共有邻居进行合并, 因此 $C_a < \tau$, 进而有 $|C_a \cap C_b| < \tau$, $\tau = \min_{a \in \{u,v\}} \tau_a$. 显然, 满足具有重叠特性的基本社区整体数量小于 n, 令 $\rho(\tau)$ 表示网络的平均节点度, 则 $\sum_{a<b} |C_a \cap C_b| < n\rho(\tau)$. 综上所述, 可知 $|C_U^{(0)}| < (\rho(\tau)+1)n$. 由于复杂网络的平均节点度 $\rho(\tau) = 2m/n$, 因此 $\mathrm{O}\left(\tau_U|C_U^{(0)}|\right) = \mathrm{O}(\tau m)$. □

3　在线网络社区挖掘

本节给出 AMSC, 演化框架的第 2 阶段. 在网络演化过程中, 由于扩张条件没有发生变化, 因此判断社区稳定核心的迁移是对新网络社区结构校准的关键.

首先讨论一下网络演变过程中可能发生的演化行为. 直观上讲, 如果对某社区 C_a 添加内部链接或删除社区 C_a 与社区 C_b 之间的外部链接, 社区轮廓会更加清晰, 进而社区的稳定性更强. 反之, 删除社区内部边或添加外部边则可能导致社区结构的破裂. 这种思路在无权网络的增量处理上可能具有一定的效果[19−20], 然而在加权网络中, 该方法未必能够真实地反映社区在演化过程中的形态变化. 例如, 不断向社区内部某区域添加新链接, 或该区域交流频率不断增强, 则可能由于区域内节点的互指向性, 导致原社区分裂为两个新社区. 在动态加权网络中, 社区结构的校准过程较为复杂, 因为任何细微的变化都可能导致社区结构不可预料的转变. 为了能够更加细致地反映复杂网络的这些变化, 本文将各时间片段网络的变化用边权变化、节点集变化和边集变化这种更加基本的事件加以描述.

3.1　权重变化处理及邻域稳定性校准

校准社区结构首先面临的难点在于处理新网络的边权变化, 显然即使两个网络具有相同的拓扑结构 (相同的节点集和边集), 但如果网络中的边具有不同的权重, 则两个网络的社区结构明显不同.

图 4 (a) 中, $\varphi^{(t-1)}(u,v) = \max_{i \in \sigma(u)} \varphi^{(t-1)}(u,i)$, 说明节点 u 对节点 v 具有较强的邻域倾向性, 节点 v 与 v_1, v_2 构成核心链接, 因此生成基本社区 $\{v, v_1\}$, $\{v, v_2\}$. 通过判断基本社区之间的同源性, 节点 u, v_1, v_2 均倾向于节点 v, 进而形成具有较高稳定性的星形社区 $C_1^{(t-1)} = \{u, v, v_1, v_2\}$. 同理, 节点 u_1, u_3 具有互倾向性由此生成社区 $C_2^{(t-1)} = \{u_1, u_2, u_3\}$.

网络演化过后 (图 4 (b)), 边 (u, u_2) 的权重由 $w^{(t-1)}(u,u_2) = 2$ 上升至 $w^{(t)}(u,u_2) = 8$. 伴随着这种权重的变化, 节点 u 对邻域的倾向程度也随之改变. 根据定义 3, u 在 t 时刻转为指向节点 u_2 从而失去了对节点 v 的倾向性, 进而引发了社区 C_1 的收缩, 同时造成了社区 C_2 的扩张.

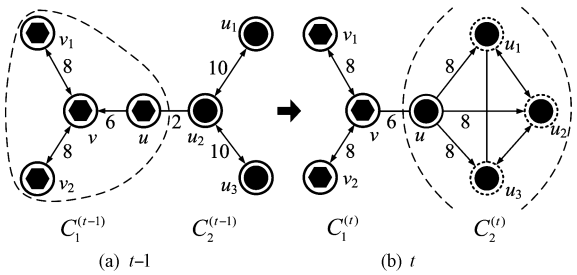


图 4　校准边权变化

Fig. 4　Calibrating weight change

图 4 描述的是权重变化对邻接社区产生的影响. 除此之外, 由于节点和链接的添加删除同样能够改变节点的权重均值, 因此这类事件对邻域产生的影响与权重变化处理方式一致. 总体来讲, 演化事件引发的权重及邻域变化可以分为 3 种情况.

情况 1. 链接权重的强弱变化. 某些幅度较小的权重变化或演化事件不会改变邻域内的权重均值, 进而不会影响临近的社区结构. 例如将图 4 (a) 中链接 (u, u_2) 的权重调整为 $w(u, u_2) = 3$, 则 (u, u_2) 在新时间片段仍然是弱指向链接, 而节点 u 仍然会指向节点 v.

情况 2. 链接的稳定性变化. 当演化事件来临时, 节点的权重均值未发生改变, 但其局部稳定性受到影响. 例如图 4 (b) 中边 (u, u_2) 是准稳定链接, 现假设将边 (u, u_2) 与边 (u, v) 的权重互换, 则权重均值不会发生改变, 但由于节点 u 失去了对 u_2 的指向性, 因此链接 (u, u_2) 的稳定性遭到了破坏, 由此导致基本社区 $CL_{u,u_2} = \{u, u_1, u_2, u_3\}$ 从候选集中移除.

情况 3. 社区的同源性变化. 某些社区可能由若干基本社区合并而来, 由于社区间的同源性涉及链接的指向性判别, 因此演化事件在影响节点权重均值的同时可能会造成邻域内同源社区的分裂. 社区的同源性判断与事件本身在处理上有许多重叠部分, 因此本节主要处理邻域变化的前两个层面, 将情况 3 放到第 3.2 节作为演化事件单独处理.

当新时间片段来临, 本文通过对原有社区进行校准, 避免社区内部节点的外部指向, 从而维持了社区的稳定性. 理论上讲, 当事件发生时就要对邻域的稳定性进行校准, 但根据定理 2, 一些无需处理的冗余操作可以从算法中排除. 除此之外, 其余邻域校准策略的实现过程如算法 3 所示.

算法 3 (权重及邻域校准算法).

输入. $t-1$ 时刻社区划分结果 $C^{(t-1)}$ 及事件 ε;

输出. 邻域校准后的社区划分结果 $C^{(t)}$;

步骤.

1) **for** $\left\{u \mid u \in \sigma^{(t)}(\varepsilon)\right\}$ **do**
2) 计算 $\delta^{(t)}(u)$;
3) **if** $\sum_i \left|\theta_{u,i}^{(t)} - \theta_{u,i}^{(t-1)}\right| = 0$ **then**
4) $S^{(t)} \leftarrow S^{(t-1)}$;
5) **else**
6) 判断节点 u 的邻域倾向性并更新 S;
7) **end if** //情况 1
8) **if** $\sum_i \left|s_{u,i}^{(t)} - s_{u,i}^{(t-1)}\right| = 0$ 或满足定理 2 **then**
9) 保持 u 邻域社区结构不变;
10) **else if** $s_{u,i}^{(t-1)} > 0$, $s_{u,i}^{(t)} = 0$ **then**
11) **if** $\{u, v\} \in CL^{(i)} \in C_U^{(t)} \backslash CL_{u,v}$ **then**
12) 保持 u 邻域社区结构不变;
13) **else**
14) $C \leftarrow CL_{i \neq \{u,v\}} \in C(CL_{u,v})$;
15) $C^{(t)} = C^{(t)} \backslash C(CL_{u,v})$;
16) **end if**
17) **else**
18) $C \leftarrow \left\{CL_{u,v} \left| s_{u,i}^{(t-1)} = 0, s_{u,i}^{(t)} > 0\right.\right\}$;
19) **end if** //情况 2
20) **end for**
21) 合并 C 与 $\sigma(C)$ 中的同源社区;
22) 更新 $C^{(t)}$.

用 $C(x)$ 表示 x 所属社区. 算法 3 在第 3) 行 ~9) 行判断邻域内的结构不变性, 第 10) 行 ~19) 行对邻域内发生变化的社区结构进行校准. 需要注意的是, 某些基本社区不止有一个支撑性架构, 因此在第 11) 行 ~12) 行保护那些失去某一社区支撑但仍具稳定性的基本社区. 当 $CL_{u,v}$ 被破坏, 其所在社区不会失去 $CL_{u,v}$ 的所有节点 (由于基本社区的重叠性), 因此首先在行第 14) 行 ~15) 行对社区中的其他基本社区做释放操作并在社区集合中删除该社区, 然后在第 21) 行对释放后的基本社区与新基本社区重新判断同源性, 并合并同源社区. 算法 3 的时间复杂度为 $\mathrm{O}(\tau\varepsilon)$, 与演化过后网络中新事件数成正比.

定理 2. 假设网络由 $G^{(t-1)}$ 演化至 $G^{(t)}$ 时有事件 ε 发生, 对于节点 $u \in \sigma(\varepsilon)$, $i \in \sigma(u) \backslash \varepsilon$, 如果对于任意 $s_{u,i}^{(t)}$ 均有 $s_{u,i}^{(t)} s_{u,i}^{(t-1)} > 0$, 则邻域 $\sigma(u) \backslash \varepsilon$ 内社区结构不变.

证明. 根据假设可知 $s_{u,i}^{(t)}$ 和 $s_{u,i}^{(t-1)}$ 同时大于 0, 说明链接 (u, i) 在网络演化前后或者是核心链接或者是准稳定链接. 根据定义 2 和定义 3 可知 u 的邻域内必然存在链接 (u, i) 使得 $s_{u,i} > 0$, 同时由于准稳定链接与核心链接具有相同的扩张条件, 因此事件邻域 $\sigma(u) \backslash \varepsilon$ 中的基本社区保持不变. □

3.2 演化事件处理

通过更细致的分析, 某些基本事件可以进一步分解为一系列其他事件. 例如, 当一个新用户 (节点) 加入网络时, 该用户通常会与网络中的其他人建立新联系. 如果将该用户看作是孤立节点, 并将其邻接边视为网络中添加的新链接, 则节点的变化可以用边变化的集合形式加以描述. 这意味着高效的链接校准算法足以处理动态网络中所有的拓扑变化.

3.2.1 添加新链接

当网络中添加新链接, 其周围的局部环境势必受到影响. 由于算法 3 已经对局部环境进行校准, 因此本节主要考察事件本身可能引发的社区变化. 本文进一步将事件 "添加新链接" 细化为 3 种子情况:

情况 1. 如果添加的新链接被判定为核心链接或准稳定链接, 则生成以该链接为核心的基本社区. 如图 5 所示, $t = 0$ 时刻节点 $a \sim e$ 同时倾向于节点 u 进而生成星形社区 C_1. $t = 1$ 时, 向网络中添加新链接 $(a, b), (b, c), (c, d), (d, e)$, 由于新链接 (c, d) 满足核心链接判定, 因此生成基本社区 $\{c, d, u\}$.

情况 2. 如果新链接破坏了社区内部基本社区的同源性, 则原社区分裂. 在情况 1 中, 由于添加新

链接后节点 $a \sim e$ 仍然对节点 u 保有一种指向性, 因此 $\{c, d, u\}$ 作为同源社区不会改变社区 C_1 的结构. 但另一种情况是, 继续向社区内部添加新链接, 进而导致某区域内部节点失去对社区其他节点的指向性. 在图 5 (c) 中, 由于边 $(a, b), (b, e), (a, e), (c, d)$ 的权重过大, 导致基本社区 $\{a, b, e, u\}$ 和 $\{c, d, u\}$ 中所有节点均失去了对节点 u 的指向性, 因此不满足同源社区合并条件, 进而分裂为重叠社区 C_1 和 C_2.

情况 3. 添加新链接 (u, v) 但不形成新的社区并不意味着节点 u 和节点 v 没有机会在同一社区. 新链接可能作为一种桥梁将节点 u 或节点 v 归纳进校准后的某核心中. 例如在图 5 (b) 中, 由于边 (a, b) 权重较小, 因此节点 a 和节点 b 互不指向, 进而未能形成基本社区. 但由于链接 (b, u) 为准稳定链接, 因此节点 a 会被 (b, u) 所吸引, 生成基本社区 $\{a, b, u\}$, 此时节点 a 和节点 b 在相同社区中.

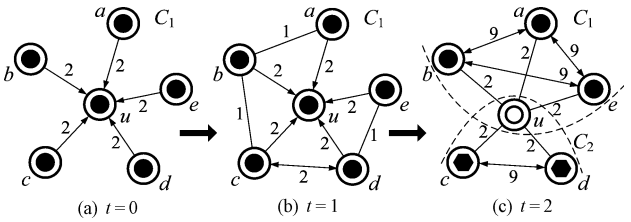


图 5　演化事件的校准实例

Fig. 5　Examples of evolutionary event calibration

上述 3 种情况的校准方案如算法 4 所示. 首先将新添加节点定义为孤立节点, 这样一来新时间片段的增量事件便可用添加新链接加以描述. 需要额外说明的是, 由于情况 1 同样面临同源社区合并, 因此不对情况 2 在算法中做额外的同源性判断, 而是对涉及指向性变化的基本社区做释放操作. 算法 4 的时间复杂度为 $\mathrm{O}(\tau\varepsilon^+)$, 与新网络增量事件总数成正比.

算法 4 (增量事件校准算法).

输入. 增量事件 ε^+;

输出. t 时刻的社区划分结果 $C^{(t)}$;

步骤.

1) 将新增节点 u 定义为新社区;
2) **for** $(u, v) \in \varepsilon$ **do**
3)　**if** $s_{u,v}^{(t)} + s_{v,u}^{(t)} > 0$ **then**
4)　$C \leftarrow CL_{u,v}$;
5)　**end if**
6) **end for**　//情况 1
7) **for** $(u, v) \in \tau(\varepsilon)$ **do**
8)　**if** $\left\{ s_{u,v}^{(t)} s_{u,v}^{(t-1)} = 0 \middle| CL^{(u)} \neq CL^{(v)} \in C_i \right\}$ **then**
9)　$C \leftarrow CL_{i \neq \{u,v\}} \in C(CL_{u,v})$;
10)　$C^{(t)} = C^{(t)} \backslash C(CL_{u,v})$;
11)　**end if**
12) **end for**　//情况 2
13) 合并 C 与 $\sigma(C)$ 中的同源社区;
14) **for** $i \in \sigma(u) \cap \sigma(v)$ **do**
15)　**if** $s_{u,i/v,i} + s_{i,u/i,v} > 0$ **then**
16)　$CL_{u,i/v,i} \leftarrow CL_{u,i/v,i} \cup \{v/u\}$;
17)　**end if**
18) **end for**　//情况 3
19) 更新 $C^{(t)}$;

3.2.2　删除原有链接

网络中的减量事件可视为增量事件的拟过程, 假设某节点 u 在新时刻离开网络, 首先将 u 的邻接边不断移除直到 $\tau(u) = 0$, 然后删除孤立点 u, 即可将事件 "删除节点" 转化为事件 "删除链接". 进一步讲, 链接的移除可分为 4 种子情况:

情况 1. 所删除的链接为孤立边 (端点的节点度均为 1). 由于孤立边没有邻域, 并且社区内部仅有两个节点, 因此直接删除该边并将链接的两个端点定义为新社区即可完成校准操作.

情况 2. 所删除的链接维持基本社区的稳定性. 核心架构 $e_{u,v}$ 移除后需要判断剩余结构 $CL_{u,v} \backslash e_{u,v}$ 的稳定性. 如果该结构仍然存在边 (i, j) 使得 $s_{i,j} > 0$, 则基本社区不会改变. 反之, 需要根据邻域校准算法判断这部分节点的归属性.

情况 3. 删除链接后导致基本社区之间的同源性增强. 这类情况与添加新链接中情况 3 类似. 不同之处在于, 添加新链接时一次指向性变化就可能导致社区的分裂, 而删除原有链接则可能需要若干次变化才能满足基本社区的合并条件. 因此对情况 3, 需要标记节点的指向性变化.

情况 4. 删除链接后导致节点从社区中脱离. 如果某节点与核心之间的 "桥梁" 被移除, 则该节点可能脱离社区. 与添加链接有所不同, 假如删除链接后, 链接某端点的节点度为 1, 则由于该节点对核心的唯一指向性, 节点可能作为星形社区的叶分支重新合并到核心之中.

算法 5 给出了减量事件的校准过程, 其时间复杂度为 $\mathrm{O}(\tau\varepsilon^-)$. 实际上算法 3 $\sim$ 5 由于所处理的对象均为演化事件本身, 因此 AMSC 算法最终的时间复杂度为 $\mathrm{O}(\tau\varepsilon + \tau\varepsilon^+ + \tau\varepsilon^-) = \mathrm{O}(\tau\varepsilon)$. 如果考虑最极端的情况, 即演化过后网络中所有链接均被替换, 则 $\mathrm{O}(\tau\varepsilon) = \mathrm{O}(\tau m)$. 极端情况缺乏合理性, 在实际操作过程中社区校准算法的时间复杂度远小于该值.

算法 5 (减量事件校准算法).

输入. 减量事件 ε^-;

输出. t 时刻的社区划分结果 $C^{(t)}$;

步骤.

1) **if** $\tau(u) = \tau(v) = 1$ **then**
2)　$C^{(t)} = C^{(t)} \backslash \{u, v\} \cup \{u\} \cup \{v\}$;
3) **end if**　//情况 1
4) **if** $s_{u,v}^{(t-1)} > 0$ **then**
5)　**if** $(u, v) \in CL^{(i)} \in C_U^{(t)} \backslash CL_{u,v}$ **then**

```
6)        保持社区结构不变;
7)     else
8)        C ← CL_{i≠{u,v}} ∈ C(CL_{u,v});
9)        C^(t) = C^(t)\C(CL_{u,v});
10)    end if
11) end if                                    //情况 2
12) for i ∈ σ(u) ∪ σ(v) do
13)    if s^(t−1)_{u,i/v,i} − s^(t)_{u,i/v,i} < 0 then
14)       if i ∈ CL^(u) ∩ CL^(v) then
15)          C ← C(CL^(u/v));
16)       end if
17)    end if
18) end for                                   //情况 3
19) 合并 C 与 σ(C) 中的同源社区;
20) for i ∈ σ(u) ∩ σ(v) do
21)    if s_{u,i} + s_{v,i} > 0, τ(u) ∧ τ(v) ≠ 1 then
22)       CL_{u,i/v,i} ← CL_{u,i/v,i} ∪ {v/u};
23)    end if
24) end for                                   //情况 4
25) 更新 C^(t);
```

4 实验与结果分析

本节通过仿真实验验证 MSC 算法、AMSC 算法的效率和性能. 实验的运行环境为 Inter Pentium IV 3.0 GHz 处理器, 2 GB 内存, Windows XP 操作系统, 算法采用 MATLAB 7.1 和 C++ 混合编程.

4.1 静态社区挖掘实验比对分析

首先将 MSC 与 4 个较为流行的重叠社区挖掘算法 OSLOMs[11]、Cfinder[18]、Greedy clique expansion (GCE)[21]、Community overlap propagation algorithm (COPRA)[22] 进行对比, 其中 Cfinder 使用较为广泛, 而 OSLOMs 被认为具有较高的重叠社区挖掘性能. 在人工网络的生成上, 选用目前被广泛认可的 LFR overlapping benchmark[23], 所生成的网络服从幂律分布, 并且内含 Ground-truth 社区. 为了避免实验结果有所偏差, 选择与上述算法相似的参数设定方式, 即: 最小社区尺寸 $c_{\min} = 10$, 最大社区尺寸 $c_{\max} = 50$, 节点的社区从属度 $o_m = 2$, 网络规模设定为 $N = 1\,000$ 和 $N = 5\,000$, 混合参数为 $\mu = 0.3$, 权重分配参数设为 $\mu_w = 0.1$, 社区重叠性比率 γ 的取值范围为 $0 \sim 0.5$. 由于 COPRA 算法具有不确定性, 因此在各对比实验中将该算法分别运行 10 次, 并取最优的运行结果. 用归一化互信息 (Normalized mutual information, NMI)[17] 度量挖掘结果与 Ground-truth 社区间的差别, 其定义为

$$NMI(C_a, C_b) = \frac{H(C_a) + H(C_b) - H(C_a, C_b)}{\sqrt{H(C_a)H(C_b)}} \tag{8}$$

这里 $H(C)$ 表示划分 C 的香农熵, 当划分 C_a 与划分 C_b 完全一致时, $NMI(C_a, C_b) = 1$, 当划分 C_a 与划分 C_b 完全不同时, $NMI(C_a, C_b) = 0$.

本文以节点数 N 为基础, 将实验分为 2 种情况. 对于每种情况, 改变重叠性比率 γ 并分析各算法的挖掘结果. 图 6 描述了算法所得的社区个数, x 线型代表 Ground-truth 社区个数. 图 6 表明, 随着重叠率 γ 的增加, MSC 挖掘的社区个数与网络实际存在的社区个数十分接近. 但在 $N = 1\,000$ 时 (图 6 (a)), MSC 识别出的社区个数要小于 Ground-truth 社区个数, 原因在于随着混合参数 μ 的增加, 社区边界更加模糊, 节点将会有更多的边链接于其他社区, 进而增加了 MSC 中同源社区的合并几率. 因此对于该情况, MSC 识别出数量较少, 但尺寸较大的社区结构. 在 NMI 指数对比中 (图 7), MSC 算法的挖掘结果在所有算法中最贴近 Ground-truth 社区, 并具有较高的稳定性. 在这类实验中, 不同参数下 NMI 指数的衰减速度也是重要的评价指标. 因此尽管在 $N = 1\,000$ 时 MSC 略逊于 COPRA 和 GCE. 但综合来看 MSC 仍表现出较大的优势.

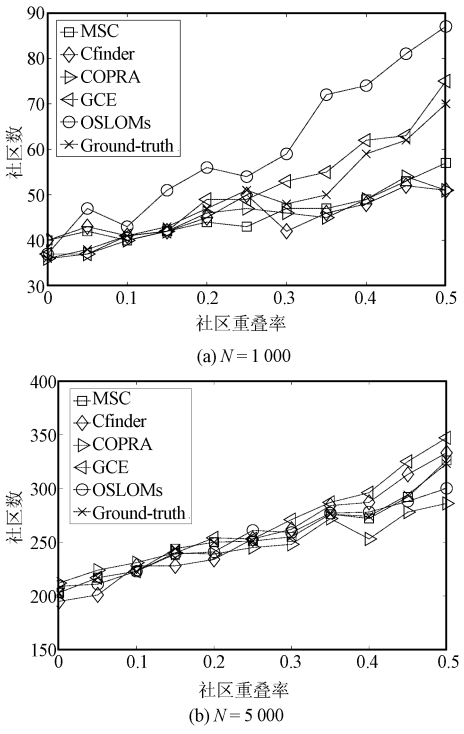
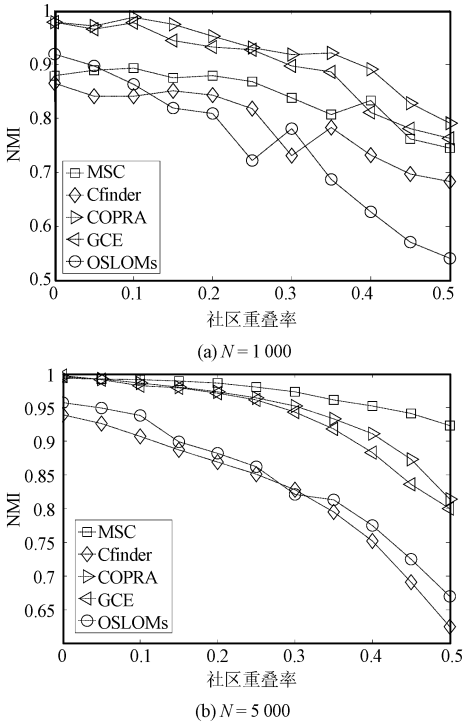


图 6 静态人工合成数据集上的社区挖掘数分析

Fig. 6 Numbers of communities on static synthesis datasets

MSC 在人工网络具有较为理想的性能, 原因在于: 首先, MSC 以节点的局部指向程度为基础构建基本社区的过程满足加权 LFR 基准网络 Ground-truth 社区内部节点强度大于外部节点强度这一原

则; 其次, MSC 的全覆盖挖掘方式符合 LFR 基准网络每个节点至少属于一个社区这一策略.



(a) $N = 1\,000$

(b) $N = 5\,000$

图 7　静态人工合成数据集上的 NMI 指数对比分析

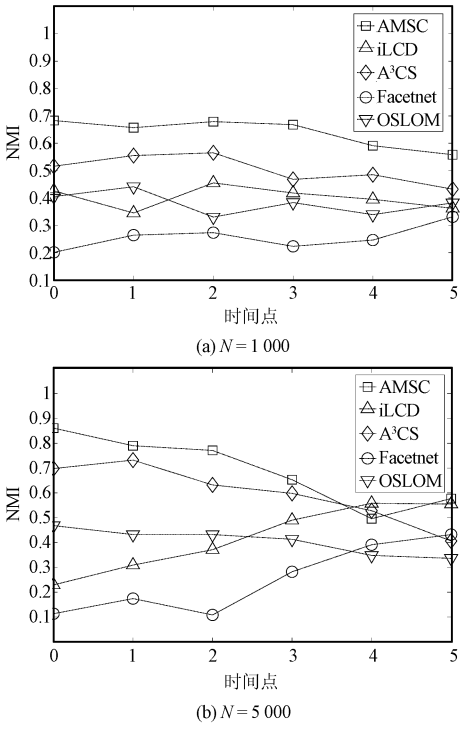
Fig. 7　Comparison results of NMI score on static synthesis datasets

4.2　动态社区挖掘实验比对分析

4.2.1　人工合成网络社区挖掘实验

接下来, 将 AMSC 算法与 4 种动态社区挖掘算法 Facetnet[6]、iLCD[9]、OSLOM[11]、A^3CS[13] 进行对比. 实验指标为 NMI 指数和社区挖掘个数. 为了使网络具有演化特性, 使用 LFR 基准网络生成具有 5 个时间片段的演化网络. 每个时间片段相比于前一时段的变化率约为 10 %. 由于 Facetnet 需要预知社区数, 因此将 Ground-truth 社区个数作为该算法的输入参数. 其余算法采用与原文一致的参数设定.

由图 8 可以看出, AMSC 算法的 NMI 指数分别超出 A^3CS、iLCD、OSLOM 和 Facetnet 约 20.6 %, 62.4 %, 69.4 % 和 2.6 %. OSLOM 的 NMI 指数较低, 主要原因是该算法需要预先得知网络的整体结构, 而这通常不可预知. A^3CS、iLCD 和 Facetnet 则表现出较强的波动趋势, AMSC 保持了最高的 NMI 指数, 并随网络演化进程的不断深入表现出较高的稳定性. 可见相比于其他 4 种算法, AMSC 与 Ground-truth 社区最为接近, 尤其适用于长演化网络.

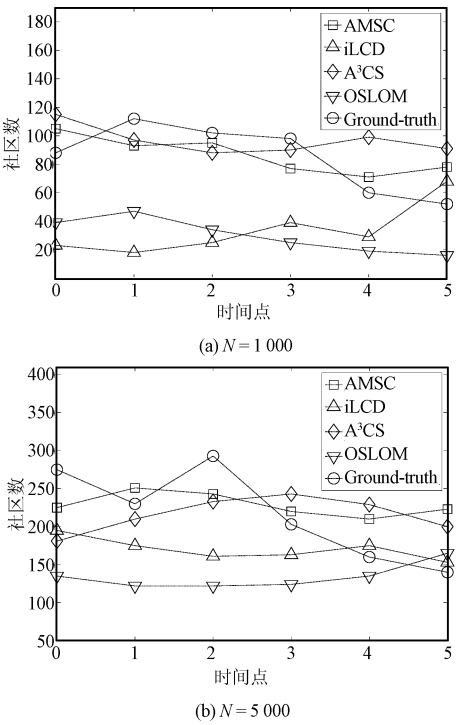


(a) $N = 1\,000$

(b) $N = 5\,000$

图 8　动态人工合成数据集上的 NMI 指数对比分析

Fig. 8　Comparison results of NMI score on dynamic synthesis datasets



(a) $N = 1\,000$

(b) $N = 5\,000$

图 9　动态人工合成数据集上的社区挖掘数分析

Fig. 9　Numbers of communities on dynamic synthesis datasets

在挖掘的社区数量方面 (图 9), AMSC 总体上

接近于 Ground-truth 社区个数, 即使社区边界较为模糊 ($\mu = 0.3$), AMSC 依然保持了较高的稳定性.

4.2.2 真实网络社区挖掘实验

本节测试算法在真实网络上的挖掘性能, 比对算法选取在人工合成网络上性能较为优秀的 A^3CS 和 OSLOM. 真实网络社区挖掘相比于人工网络更具挑战性, 由于真实网络不存在 Ground-truth 社区, 因此本节主要从模块度、社区数量和运行时间三方面考察算法的性能.

实验 1. Enron email network. 安然邮件数据集源自安然公司内部邮件网络, 包含的 150 个用户大都为公司高管, 该数据集中链接的权重被设定为用户之间收发邮件的总数. 数据集的时间跨度为 111 周 (1999 年 12 月 ∼ 2002 年 3 月), 在这段时间内, 发生了许多标志性事件, 如 CEO 变动、公司破产等. 通过对原始数据进行精炼, 本文选取了约 50 % 的全体链接生成了 20 个随时间不断变化的时间片段, 其中链接的变化量约为每时间段 1 000 条.

如图 10 所示, AMSC 算法所得社区的模块度远大于 OSLOM, 略低于 A^3CS. 但 AMSC 在网络演化时仍然能够挖掘出相同数量的社区, 因此具有更高的稳定性, 而 OSLOM 和 A^3CS 则波动较大. 在运行时间方面, AMSC 的社区识别时间不到 1 s, OSLOM 和 A^3CS 分别需要 1.5 s 和 2.5 s, 可见 AMSC 的增量式处理方案是有效的, 并且具有较高的效率.

实验 2. arXiv e-print citation network. arXiv 数据集得自 KDD CUP 2003, 主题面向动态复杂网络数据挖掘. 该数据集包含从 1996 年 2 月 ∼ 2003 年 5 月的超过 225000 篇文献, 权重设定为文献之间的互引次数. 在本实验中, 使用 MSC 算法对 1996 和 1997 两年的引文数据进行初始网络社区识别, 接下来将 1998 ∼ 2003 年间的电子文档生成 30 个以 2 月为周期的时间片段.

图 11 表明, AMSC 算法在社区的模块度方面具有较大的优势. 具体来讲, AMSC 算法高出 A^3CS 算法约 6 % ∼ 10 %, 至少是 OSLOM 算法的 1.6 倍. 在引文网络中, OSLOM 最多挖掘出约 300 个社区, 而 AMSC 和 A^3CS 识别出的社区数则相对较少. A^3CS 算法在各时间片段几乎识别出相同的社区数量, 一个主要的原因是 A^3CS 算法基于模块度优化, 因此存在分辨率极限. 由于引文网络中社区普遍具有同源性 (一个可能的原因是研究人员对学术领袖的指向性较强), 因此 AMSC 挖掘出规模略大, 数量较少的社区结构. 运行时间上, AMSC 能够在不到 2 s 的时间内完成对社区稳定性的校准工作, 而其他两种算法则平均需要 15 s 左右才能完成社区识别任务.

实验 3. Facebook network (New Orleans). 该数据集包括 Facebook 在新奥尔良区域约 60000 用户的友谊信息, 时间跨度为 2006 年 9 月 ∼ 2009 年 2 月, 其中用户间的社交频率被设定为数据集的链接权重. 在本实验中 2006 年 9 月 ∼ 2007 年 1 月中的累积数据被用作识别初始社区, 其余月份以 1 月为周期生成 25 个独立的时间片段.

实验结果在图 12 中给出, AMSC 识别的社区仍然具有较高的质量, 而 OSLOM 算法在人工数据集上的优势没能在真实数据集上得以体现. 在运行时间上, 大规模数据对所有算法均造成较大的负担. AMSC 算法识别演化社区的时间平均需要 4 ∼ 6 s, 而 OSLOM 算法则超出 AMSC 约 10 倍左右时间. 这一现象的主要原因是网络在演化时实际发生的事件数很少, 而 OSLOM 需要在新网络上重复计算, 因此导致过高的时间开销. 对于 AMSC, 由于在 Facebook 网络中节点 (人) 通常以亲密程度为基础生成社区, 因此在网络演化过程中很少出现社区同源性破坏的情况. 在这类实验中需要额外说明的是,

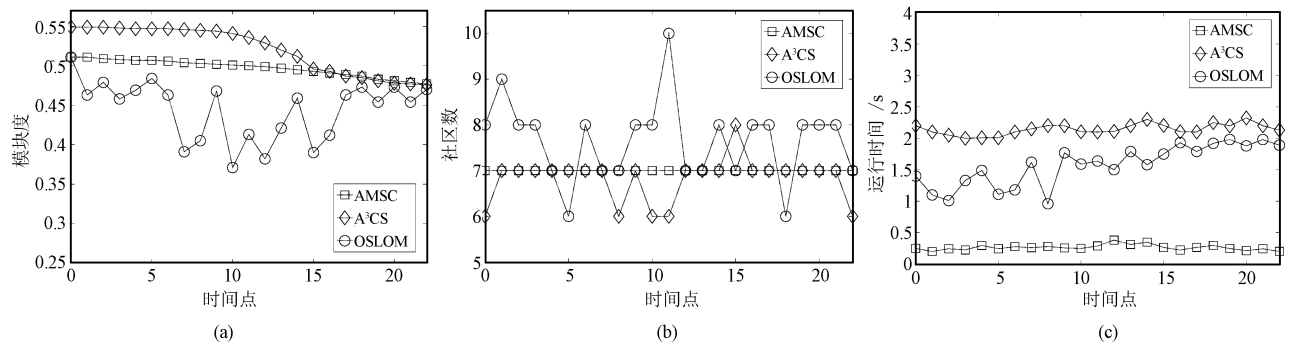


图 10 Enron 数据集仿真结果 ((a) 社区模块度; (b) 所识别社区个数; (c) 运行时间)

Fig. 10 Simulation results on Enron dataset ((a) Modularity of communities; (b) Number of identifying communities; (c) Running time)

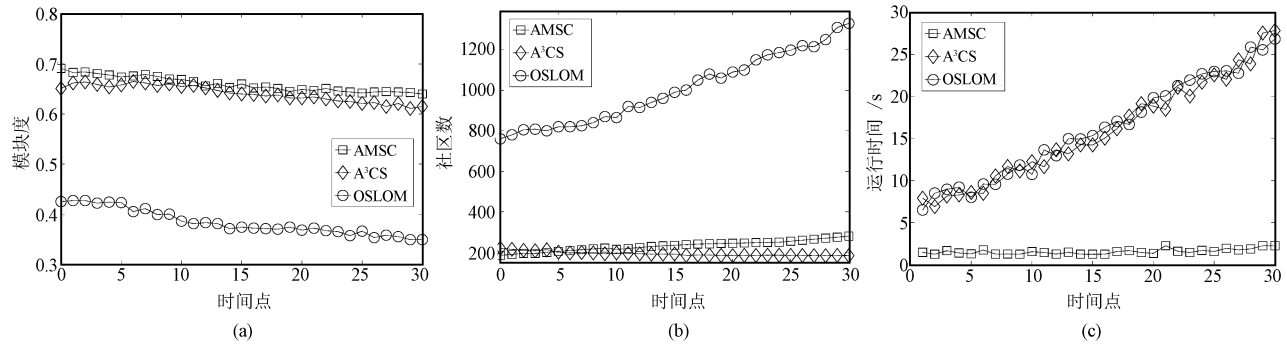


图 11 arXiv 数据集仿真结果 ((a) 社区模块度; (b) 所识别社区个数; (c) 运行时间)

Fig. 11 Simulation results on arXiv dataset ((a) Modularity of communities; (b) Number of identifying communities; (c) Running time)

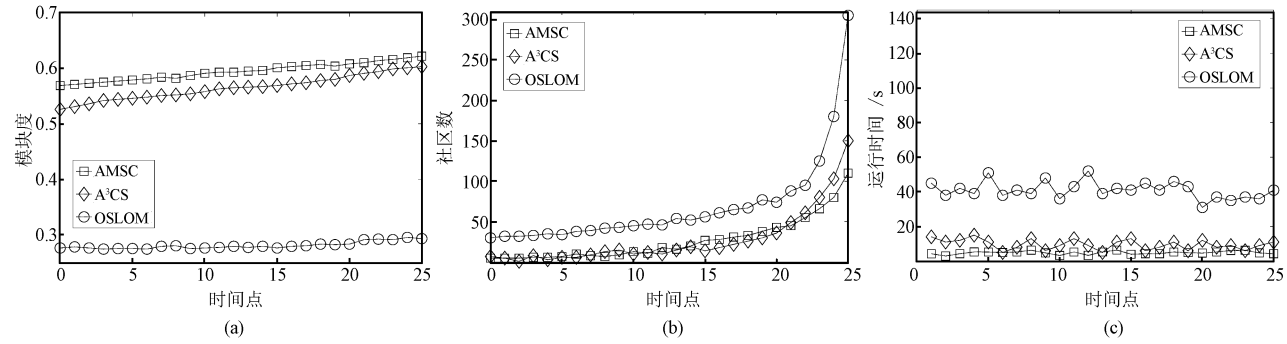


图 12 Facebook 数据集仿真结果 ((a) 社区模块度; (b) 所识别社区个数; (c) 运行时间)

Fig. 12 Simulation results on Facebook dataset ((a) Modularity of communities; (b) Number of identifying communities; (c) Running time)

随着网络的不断演化, 算法识别出的社区数量越来越多. 从社区的稳定性角度来讲, 新加入网络的节点通常会选择参与到某一主题中形成一种大规模的星形社区. 然而随着演化进程的不断深入, 人们开始以这些核心为纽带形成独特的导引型群体, 进而形成大量局部的内聚性较强的小型社区. 此外局部交流过于频繁造成的社区同源性分裂也是 AMSC 识别出较多社区的主要原因.

5 结论

本文提出一种增量式动态网络社区识别算法 AMSC. 不同于以往的动态社区挖掘方法, AMSC 无需在新时间片段重复运行, 并且可以在无参数条件下识别加权网络中具有任意形状的社区结构. 文中首先基于节点的邻域倾向性将社区定义为一种具有较高稳定程度的局部子结构, 然后针对网络的快速演变提出一种基于事件的社区稳定性校准策略以此获得新网络社区. 由于算法排除掉部分可预知的社区识别结果, 因此具有较高的效率. 在人工合成网络和真实网络上的仿真结果表明, AMSC 具有较高的挖掘性能和较低的时间开销, 并且随网络的演化表现出较高的稳定性.

在未来的工作中, 我们拟针对具有异构特性的动态网络社区挖掘展开研究, 实现对真实世界的精准感知. 此外基于社区结构的动态网络蠕虫遏制策略也是下一步的研究重点.

References

1 Hopcroft J, Khan O, Kulis B, Selman B. Tracking evolving communities in large linked networks. *Proceedings of the National Academy of Sciences of the United States of America*, 2004, **101**(1): 5249−5253

2 Asur S, Parthasarathy S, Ucar D. An event-based framework for characterizing the evolutionary behavior of interaction graphs. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM, 2007. 913−921

3 Sun J, Faloutsos C, Papadimitriou S, Yu P S. Graphscope: parameter-free mining of large time-evolving graphs. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM, 2007. 687−696

4 Palla G, Barabási A L, Vicsek T. Quantifying social group evolution. *Nature*, 2007, **446**(7136): 664−667

5 Wu Bin, Wang Bai, Yang Sheng-Qi. Framework for tracking the event-based evolution in social networks. *Journal of*

Software, 2011, **22**(7): 1488−1502
(吴斌, 王柏, 杨胜琦. 基于事件的社会网络演化分析框架. 软件学报, 2011, **22**(7): 1488−1502)

6 Lin Y R, Chi Y, Zhu S H, Sundaram H, Tseng B L. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In: Proceedings of the 17th International Conference on World Wide Web. Beijing, China: ACM, 2008. 685−694

7 Tang L, Liu H, Zhang J P. Identifying evolving groups in dynamic multimode networks. *IEEE Transactions on Knowledge and Data Engineering*, 2012, **24**(1): 72−85

8 Gauvin L, Panisson A, Cattuto C. Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach. *Plos One*, 2014, **9**(1): e86028

9 Cazabet R, Amblard F, Hanachi C. Detection of overlapping communities in dynamical social networks. In: Proceedings of the 2010 IEEE Second International Conference on Social Computing. Minneapolis, Minnesota, USA: IEEE, 2010. 309−314

10 Pérez-Suárez A, Mart í nez-Trinidad J F, Carrasco-Ochoa J A, Medina-Pagola J E. An algorithm based on density and compactness for dynamic overlapping clustering. *Pattern Recognition*, 2013, **46**(11): 3040−3055

11 Lancichinetti A, Radicchi F, Ramasco J J, Fortunato S. Finding statistically significant communities in networks. *Plos One*, 2011, **6**(4): e18961

12 Lin Wang-Qun, Deng Lei, Ding Zhao-Yun, Wu Quan-Yuan, Jia Yan, Zhou Bin. Hierarchical dynamic community detection by parallel computing. *Chinese Journal of Computers*, 2012, **35**(8): 1712−1725
(林旺群, 邓镭, 丁兆云, 吴泉源, 贾焰, 周斌. 一种新型的层次化动态社区并行计算方法. 计算机学报, 2012, **35**(8): 1712−1725)

13 Dinh T N, Nguyen N P, Thai M T. An adaptive approximation algorithm for community detection in dynamic scale-free networks. In: Proceedings of the 32nd IEEE International Conference on Computer Communications. Piscataway, NJ, USA: IEEE, 2013. 55−59

14 Fortunato S. Community detection in graphs. *Physics Reports*, 2010, **486**(3): 75−174

15 Wang G, Zhao Y, Shi X, Yu P S. Magnet community identification on social networks. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM, 2012. 588−596

16 Newman M E J. Analysis of weighted networks. *Physical Review E*, 2004, **70**(5): 056131

17 Liu Xu, Yi Dong-Yun. Complex network community detection by local similarity. *Acta Automatica Sinica*, 2011, **37**(12): 1520−1529
(刘旭, 易东云. 基于局部相似性的复杂网络社区发现方法. 自动化学报, 2011, **37**(12): 1520−1529)

18 Palla G, Derenyi I, Farkas I, Vicsek T. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 2005, **435**(7043): 814−818

19 Yang Bo, Liu Jie, Liu Da-You. A random network ensemble model based generalized network community mining algorithm. *Acta Automatica Sinica*, 2012, **38**(5): 812−822
(杨博, 刘杰, 刘大有. 基于随机网络集成模型的广义网络社区挖掘算法. 自动化学报, 2012, **38**(5): 812−822)

20 Zhang Xin-Meng, Jiang Sheng-Yi. Complex network community detection based on core graph incremental clustering. *Acta Automatica Sinica*, 2013, **39**(7): 1117−1125
(张新猛, 蒋盛益. 基于核心图增量聚类的复杂网络划分算法. 自动化学报, 2013, **39**(7): 1117−1125)

21 Lee C, Reid F, McDaid A, Hurley N. Seeding for pervasively overlapping communities. *Physical Review E*, 2011, **86**(3): 066107

22 Gregory S. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 2010, **12**(10): 103018

23 Lancichinetti A, Fortunato S. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 2009, **80**(1): 016118

杨海陆 哈尔滨工程大学计算机科学与技术学院博士研究生. 主要研究方向为数据挖掘, 社会计算和社会网络安全.
E-mail: yanghailu@hrbeu.edu.cn
(**YANG Hai-Lu** Ph. D. candidate at the College of Computer Science and Technology, Harbin Engineering University. His research interest covers data mining, social computing and social network security.)

张健沛 博士, 哈尔滨工程大学计算机科学与技术学院教授. 主要研究方向为数据挖掘, 人工智能, 数据与知识工程和社会计算. 本文通信作者.
E-mail: zhangjianpei@hrbeu.edu.cn
(**ZHANG Jian-Pei** Ph. D., professor at the College of Computer Science and Technology, Harbin Engineering University. His research interest covers data mining, artificial intelligence, data and knowledge engineering and social computing. Corresponding author of this paper.)

杨 静 博士, 哈尔滨工程大学计算机科学与技术学院教授. 主要研究方向为数据挖掘, 模式识别, 社会计算和隐私保护.
E-mail: yangjing@hrbeu.edu.cn
(**YANG Jing** Ph. D., professor at the College of Computer Science and Technology, Harbin Engineering University. Her research interest covers data mining, pattern recognition, social computing and privacy preserving.)