软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(3):552−572 [doi: 10.13328/j.cnki.jos.005699]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

复杂条件下的社区搜索方法*

竺俊超, 王朝坤

(清华大学 软件学院,北京 100084)

通讯作者: 王朝坤, E-mail: chaokun@tsinghua.edu.cn

摘 要: 社区搜索旨在寻找包含给定节点集的社区,能够快速获取个性化的社区信息.针对现有社区搜索算法难以满足复杂搜索条件的现状,提出条件社区搜索这一新问题.解决该问题有助于对社交网络进行智能分析,在复杂搜索条件下为用户提供更好的社区结果.首先,基于布尔表达式,给出条件社区搜索问题的形式化定义,可有效表达给定节点不能出现在社区内以及给定节点中至少有一个出现在社区内的要求.接着,提出解决条件社区搜索问题的通用框架,包括对搜索条件进行简化、根据简化后的搜索条件进行多次单项条件社区搜索、合并各单项条件社区搜索的结果等主要步骤.同时,提出"社区搜索+过滤"的方法和给点加权的方法来进行单项条件社区搜索.最后,真实数据集上的大量实验结果表明所提方法的正确性和有效性.

关键词: 社区结构;局部社区发现;社区搜索;条件社区搜索;布尔表达式

中图法分类号: TP311

中文引用格式: 竺俊超,王朝坤.复杂条件下的社区搜索方法.软件学报,2019,30(3):552−572. http://www.jos.org.cn/1000-9825/5699.htm

英文引用格式: Zhu JC, Wang CK. Approaches to community search under complex conditions. Ruan Jian Xue Bao/Journal of Software, 2019,30(3):552−572 (in Chinese). http://www.jos.org.cn/1000-9825/5699.htm

Approaches to Community Search Under Complex Conditions

ZHU Jun-Chao, WANG Chao-Kun

(School of Software, Tsinghua University, Beijing 100084, China)

Abstract: Community search aims to find out communities containing a given set of nodes and get personalized community information quickly. Since traditional community search algorithms can hardly meet the needs under complex conditions, a new problem called conditional community search is proposed. Solving the problem helps to analyze social networks intelligently and provides users with better community results under complex search conditions. First, based on Boolean expressions, the formal definition of conditional community search problem is given, which can effectively express the requirement that a given node cannot exist in the community and at least one of the given nodes occurs in the community. Then, a general framework is proposed to solve the problem of conditional community search, including simplifying search conditions, conducting multiple singleconditional community searches according to simplified search conditions, and combining the results of singleconditional community searches. At the same time, a community search plus filtering method and a node weighting based method are proposed to carry out the singleconditional community search. Finally, extensive experimental results conducted on real-world datasets show the correctness andeffectiveness of the proposed methods.

Key words: community structure; local community detection; community search; conditional community search; Boolean expression

由大量节点和节点间的连接关系形成的网络结构广泛存在于计算机科学、生物学和社会学[1,2]等领域,例如以网页为节点、以网页间的链接为边组成的万维网[3]和以人为节点、以人际间关系为边建立的社会网[1,2]等.

• 基金项目: 国家自然科学基金(61872207)
Foundation item: National Natural Science Foundation of China (61872207)
本文由智能数据管理与分析技术专刊特约编辑樊文飞教授、王国仁教授、王朝坤副教授推荐.
收稿时间: 2018-07-21; 修改时间: 2018-09-20; 采用时间: 2018-11-01

在网络相关研究工作中,社区(community)的概念持续受到人们的关注.一般而言,社区是指内部节点间联系较内部与外部节点间联系更为紧密的子网络.发现网络中的各种社区结构有助于进行好友推荐、犯罪团伙识别以及蛋白质功能预测[4−6],同时能够有效支持网络中传播热点选择[7]和介数中心度更新[8].

社区搜索(community search)是指给定一个或多个节点,寻找包含它们的社区[9−14].与社区发现(community detection)相比,它更关注局部的网络结构,能够返回更加个性化的社区结果.

现有的社区搜索方法包括仅与网络拓扑有关的社区搜索和与节点属性有关的社区搜索:前者旨在寻找包含给定节点集且满足 k-clique[9],k-core[10,11]或 k-truss[12,13]等特定拓扑结构的社区;后者在寻找包含给定节点集的社区时综合考虑了拓扑结构和节点属性[15−17],返回的结果社区不仅要满足特定拓扑结构,还要使内部节点的属性尽可能相近.

然而,已有的社区搜索研究成果尚不能满足人们日益增长的客观需求.例如,在实际应用中经常会遇到这样一些问题:如何准确地找到一个社区,使它不仅包含某些给定节点,同时不包含另一些给定节点?如何智能地找到一个社区,至少要包含给定的 5 个节点中的任意 3 个?本文将这类包含对节点条件约束的社区搜索问题统称为条件社区搜索问题(conditional community search,简称 CCS).据我们所知,目前国内外尚未见针对 CCS 问题的公开报道.

例 1(社交场景):用户 A 和 B 拟共同组建一个学习讨论小组,组内成员尽可能互相认识以便讨论,且不希望保险推销员 C 参与.则 A 和 B 可以邀请哪些人加入该小组?

图 1 展示了上述社交场景中的部分网络结构,其中,节点代表人物,边代表好友关系.若以 A 和 B 作为社区必须包含的节点,则依据以 2-core 为基础的社区搜索方法得到的结果社区如实线圈所示.这是因为该社区中每位用户在本社区内都有至少两位好友并且该社区仅有两条从内部连向外部的边,具有内部节点间联系较内部与外部节点间联系更加紧密的性质.但是该社区显然不满足节点 C 不参与的要求.如果应用 CCS 来计算,即找到一个包含 A 和 B,同时不包含 C 的社区,那么结果社区如虚线圈所示.这是因为该社区内每位用户都有至少两位好友,同时节点 C 不出现在此社区中.进一步而言,C 也很难通过其好友加入到该社区内.因此对于例 1,A 和 B 可以邀请虚线圈中的成员组建讨论小组.此外,在视频网站为用户提供社区推荐、购物网站为用户提供相关商品推荐等场景下,也会遇到用户不希望特定的人物或商品出现在推荐列表中的需求.

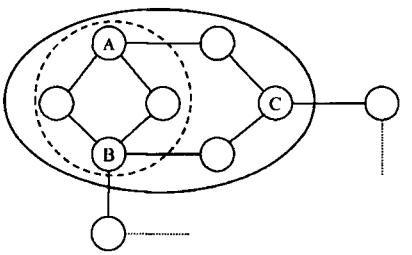

Fig.1 An example for conditional community search

图 1 条件社区搜索的示例

例 2(商业场景):某公司发生了商业机密泄露事件,内部能够接触该机密的有 A,B 两人,而竞争对手公司中取得此机密的是 C,D 两人.该公司希望对 A 和 B 展开调查,找出 A 或 B 中至少一人可能与 C 或 D 存在的联系.通过邮件、电话等联系方式建立了社会关系网络后,可以通过寻找至少包含 A,B 中一人、C,D 中一人的社区来获得相关信息.

例 2 中的场景对于结果社区提出了一类典型需求,即至少包含给定节点中的任意一个.显然,传统社区搜索方法缺乏对该需求的有效支持;借助 CCS,可以将上述需求通过布尔表达式的形式清晰地予以表示.

上述实例表明,现有的社区搜索方法难以有效解决迫切的客观需求.为此,本文提出并研究条件社区搜索问题,给出条件社区搜索的通用框架,尝试对社交网络进行智能分析,在复杂的搜索条件下,为用户提供更好的结

果社区.

事实上,传统的社区搜索问题可以看做是条件社区搜索问题的特例,其中,搜索条件退化为社区中需要包含全部给定节点.需要注意的是:本文关注如何在复杂条件下进行社区搜索来获得更加符合实际需求的结果社区,而未涉及新社区结构的设计,即现有的社区结构都可以应用到条件社区搜索中来.

本文的主要贡献包括:

(1) 提出条件社区搜索这一新问题,并给出其形式化定义;

(2) 提出条件社区搜索的通用框架,并对其中的单项条件社区搜索步骤给出"社区搜索+过滤"的方法和基于标签传播给点加权的方法(weighting by label propagation,简称 WLP);

(3) 在真实数据集上进行了大量实验,实验结果验证了所提方法的正确性和有效性.

本文第 1 节介绍相关工作,包括社区发现和社区搜索的一般方法.第 2 节给出条件社区搜索问题的形式化定义、解决该问题的通用框架和搜索条件的简化策略.第 3 节提出对于单项条件社区搜索的解决方法.第 4 节报告并分析实验结果,包括对搜索条件简化的有效性验证以及"社区搜索+过滤"方法和 WLP 方法在时间开销和社区结果质量上的比较.第 5 节总结全文.

1 相关工作

本节介绍社区发现和社区搜索的概念,并简要回顾解决这两类问题的方法.

1.1 社区发现

社区发现,亦称为全局社区发现(global community detection),指找出给定网络图中的所有社区.社区这一概念尚没有统一的形式化定义,目前研究工作中的社区定义一般依据特定拓扑结构或者描述子图紧密程度的度量指标给出.社区发现的常用方法主要包括划分、聚类、标签传播等[18].

划分方法指通过删除网络图中的一些边,将原图自然地分成若干个连通分量来得到社区结果.Kernighan-Lin 算法[19]要求最大化社区内节点连边数量与不同社区间节点连边数量的差值.SCD 算法[20]要求删除不属于任何三角形的边.KMF 算法[21]则要求删除两端点共同邻居节点的个数少于给定阈值的边.

聚类方法包括层次聚类、谱聚类、*k*-means 聚类等.层次聚类通过自上而下不断删除边或者自下而上不断加入点的方式得到网络图的层次结构,然后切分该结构来得到社区.典型方法包括 Fast-Newman[22],CNM[23],Radicchi[24],GN[25],MSG-VM[26]和 DOC[27]等.谱聚类通过将网络表示为特定的拉普拉斯矩阵,基于同一社区内的节点在矩阵中特征向量近似的思想进行聚类[28−30].*k*-means 聚类是常见的聚类方法.在社区发现问题中,常用点和点之间的距离[31]、Random Walk[32,33]等方式给出两点间相似度量,接着采用 *k*-means 聚类进行社区发现.近年来还涌现出基于深度学习的聚类方法,即学习网络节点的低维向量表示,再通过聚类得到社区,如 CoDDA[34],DeepWalk[35]和 GraRep[36]等.

标签传播方法通常为网络中的每个节点赋予初始标签,接着模拟信息传播过程为每个节点更新标签,最后通过标签分布确定社区归属.基于标签传播思想的典型方法包括 LPA[37],HANP[38]和 SLPA[39]等.

1.2 社区搜索

社区搜索也称局部社区发现(local community detection).给定一个或多个节点,社区搜索旨在寻找包含这些节点的社区.由于关注局部网络结构,社区搜索能够高效地找到用户关心的节点所在的社区.目前常见的社区搜索算法主要基于 *k*-clique[9],*k*-core[10,11]和 *k*-truss[12,13]等特定结构.例如,Cui 等人提出了寻找包含一个给定节点的 *k*-core 社区的问题(CST)和对应算法[11].该算法从给定节点向外扩展得到结果社区.Huang 等人提出了基于 *k*-truss 的社区定义,并设计了 TCP-index 来寻找包含某个给定节点的社区[12].

此外还有一类结合拓扑结构和节点属性的社区搜索方法[15−17].例如:Shang 等人根据节点在拓扑结构和属性上的相似关系构建一个 TA-graph,并在此基础上提出与节点属性相关的社区搜索算法 AGAR[15];Fang 等人在 *k*-core 结构基础上要求社区内节点共享尽可能多的标签属性,设计了对应索引结构 CL-tree[16];Huang 等人在

k-truss 结构的基础上设计了一个打分函数来度量给定节点属性在社区内的流行程度,并提出 Attribute Truss 社区定义[17].

2 条件社区搜索

条件社区搜索问题不仅要能够描述社区必须包含给定节点这一基本搜索条件,还要能够描述两种新的搜索条件:一是社区不允许包含给定节点,二是社区至少要包含若干给定节点中的一个.本节首先基于布尔表达式给出搜索条件的形式化表达,接着提出条件社区搜索通用框架,最后给出搜索条件的简化方法.

2.1 搜索条件的形式化表达

布尔表达式是由布尔变量和逻辑运算符组成的式子.布尔变量的取值为真或假,亦可用 1 或 0 来表示.逻辑运算符包括与($\wedge$)、或($\vee$)、非($\neg$)等.在每个布尔变量的取值确定后,可以判断整个布尔表达式的真假.于是,可以考虑用布尔表达式表示搜索条件.

2.1.1 搜索条件的表示形式

通常,对于一个给定社区,一个节点是否存在于该社区中可以表示为一个布尔变量.若节点存在于该社区中,则对应布尔变量取值为真;反之为假.本文将指代节点的布尔变量称为节点变量.借助节点变量和逻辑运算符构成的布尔表达式,可以有效表示条件社区搜索问题中的具体搜索条件.

定义 1(基本搜索条件). 基本搜索条件规定为:

(1) 单个节点变量 X 是基本搜索条件;

(2) 如果 X 和 Y 是基本搜索条件,那么 $X\wedge Y$ 是基本搜索条件;

(3) 当且仅当有限次地应用条件(1)和条件(2)所得到的布尔表达式称为基本搜索条件.

定义 2(搜索条件). 搜索条件规定为:

(1) 单个节点变量 X 是搜索条件;

(2) 如果 X 是搜索条件,那么 $\neg X$ 是搜索条件;

(3) 如果 X 和 Y 是搜索条件,那么 $X\wedge Y$,$X\vee Y$ 是搜索条件;

(4) 当且仅当有限次地应用条件(1)~条件(3)所得到的布尔表达式称为搜索条件.

本文将满足定义 2 但不满足定义 1 的布尔表达式称为复杂搜索条件.

例 3:例 1 中的搜索条件可表示为 $A\wedge B\wedge\neg C$.该布尔表达式在变量 A 和 B 为真,且 C 为假时取真值,对应包含节点 A 和 B 但不包含节点 C 的社区.例 2 中的搜索条件可表示为$(A\vee B)\wedge(C\vee D)$.该布尔表达式在变量 A 或 B 为真,且 C 或 D 为真时取真值,对应包含 A,B 中某一人和 C,D 中某一人的社区.

以上实例表明,布尔表达式能够有效表示搜索条件.同时,也容易根据社区中是否包含某个节点得到对应变量的值,从而验证搜索条件.接下来给出条件社区搜索问题的形式化定义.

定义 3(条件社区搜索问题 CCS). 给定连通图 $G=(V,E)$和节点集 $V'\subset V$,寻找至少一个满足如下条件的节点集 H:(1) $H\subset V$;(2) $G[H]$是连通的,其中,$G[H]$表示 H 的导出子图;(3) H 满足定义在 V' 上的搜索条件 F;(4) H 满足用户给定的社区定义 $\mathcal{C}$.

其中,搜索条件 F 按定义 2 给出,$\mathcal{C}$可以按用户需求进行指定,例如基于 k-core[10,11]或 k-truss[12,13]的社区定义.

事实上,CCS 包含了现有的社区搜索问题.易知有如下引理.

引理 1. 社区搜索问题是条件社区搜索问题的特例.

这是因为当搜索条件 F 为基本搜索条件,即 F 中仅包含节点变量和逻辑与($\wedge$)时,可以把 F 涉及的节点合成一个节点集作为社区搜索算法的输入.此时,条件社区搜索问题就退化成社区搜索问题.

2.1.2 条件社区搜索的通用框架

条件社区搜索的通用框架将用户给定的条件社区搜索问题分解为若干单项条件社区搜索(single conditional community search,简称 SCCS)分别处理,而后进行结果汇聚.需要注意的是,尽管条件社区搜索存在

一种朴素的解决方法——首先对给定网络进行全局社区发现来得到所有社区,接着判断每个社区是否满足搜索条件,最后留下满足搜索条件的社区,但是该方法需要找到全部社区,时间开销巨大.

定义 4(单项条件社区搜索 SCCS). 对于一个条件社区搜索问题,如果有且仅有一组节点变量的取值能满足它的搜索条件 F,则称其为单项条件社区搜索.

定理 1. 任意一个单项条件社区搜索的搜索条件等价于一个合取式(仅由逻辑与运算符连接节点变量或其否定构成的式子).

证明:因为只有一组能满足搜索条件的节点变量取值,所以每个节点能否存在于社区中是唯一确定的.于是可以构造这样一个等价合取式:先用逻辑非修饰不允许出现在社区中的节点变量,而后用逻辑与连接所有节点变量. □

给定一个搜索条件,可以先枚举各节点变量的可能取值,计算得到一个真值表,从而找出所有能满足搜索条件的节点变量取值组合.用 1 和 0 分别代表节点是否存在于社区中,于是,满足例 1 中搜索条件的节点变量组合仅有 1 组,即 A=1,B=1,C=0,例 2 则有 9 组.接着,对每一种取值组合尝试找到对应的单项社区搜索的结果.最后,将每个单项社区搜索的结果合并,就能得到条件社区搜索的结果.

综上,可以归纳出三阶段的条件社区搜索通用框架.

(1) 枚举所有满足条件的节点变量取值组合;

(2) 将每一个组合对应的合取式作为单项条件社区搜索的条件输入并执行;

(3) 合并所有单项条件社区搜索的结果.

2.2 条件社区搜索问题的复杂性

条件社区搜索问题的复杂性和给定的社区定义及搜索条件有关.通常情况下,CCS 是 NP 完全的.

由引理 1,在基本搜索条件下,CCS 退化为社区搜索问题.许多传统的社区搜索问题已被证明是 NP 完全的,例如寻找满足 k-core 定义的最小社区的问题(mCST)[11].

此外,若把连通子图作为社区定义,则 CCS 是 NP 完全的.这是因为 CCS 的结果能够在多项式时间内验证,是一个 NP 问题;同时,布尔逻辑的可满足性问题(SAT)可以归约到 CCS 上.归约的方法是以布尔表达式的所有变量对应的节点构造一个完全图,将布尔表达式直接对应于搜索条件.此时,CCS 每产生一个结果就等同于找到一组满足布尔表达式的解.

这意味着对于上述几类 CCS 问题很难找到快速有效的多项式时间算法.需要注意的是:并非所有的 CCS 都是 NP 完全的,如果能够在多项式时间内找到所有的社区结构(如 k-truss[12]结构就存在多项式时间的算法),那么可以利用逐一验证搜索条件的朴素方法解决 CCS,尽管这样做的效率较低.

2.3 搜索条件的简化

因为布尔逻辑的可满足性(SAT)问题是一个 NP 完全问题,所以很难对第 2.1 节提出的条件社区搜索通用框架的第 1 步进行优化.在实际应用中,考虑到用户输入的节点变量总数一般不会很多,采用枚举形式寻找满足搜索条件的变量取值是可行的.若用户输入的节点数量较多,则可以使用求解 SAT 的快速算法,例如 DPLL[40], GRASP[41]等.注意到通用框架的第 2 步对每个满足条件的取值组合都要进行一次单项条件社区搜索,于是考虑改进这一步骤,在一次搜索中同时处理多个变量取值组合,从而减少总搜索次数.

首先,在得到所有满足搜索条件的变量取值组合后,可写出与搜索条件等价的主析取范式[42],即对搜索条件进行规范化.例如,用户输入的搜索条件是$\neg A\wedge(B\vee C)$,满足该式的变量取值组合是:A=0,B=1,C=1;A=0,B=0,C=1 和 A=0,B=1,C=0,从而把搜索条件规范化为$(\neg A\wedge B\wedge C)\vee(\neg A\wedge\neg B\wedge C)\vee(\neg A\wedge B\wedge\neg C)$.主析取范式是合取式的析取,其中每个合取式包含所有节点变量,对应于一组满足条件的变量取值,即一次单项条件社区搜索.

接下来,排除主析取范式形式的搜索条件可能存在的冗余.例如,范式$(A\wedge B)\vee(A\wedge\neg B)$可化简成 A,即变量 B 存在与否对搜索条件不产生实际影响.本文用奎因-麦克拉斯基(Quine-McCluskey,简称 QM)算法[43]将主析取范式转化为等价的最简与或式,从而消除此类冗余,进而减少单项条件社区搜索次数.由于最简与或式具有最少的合

取式个数,所以根据最简与或式进行单项条件社区搜索的次数是最少的.

最后,在最简与或式基础上进一步发现可以通过提取部分合取式的公共变量来提高搜索效率.以搜索条件$(A\wedge B\wedge C)\vee(A\wedge B\wedge D)$为例,需要对该最简与或式中的两个合取式分别进行一次单项条件社区搜索.容易发现,这两个合取式里都出现了变量 A 和 B.若将其提取到外部,则该式变形为$(A\wedge B)\wedge(C\vee D)$,可看做一个合取式与一个析取式的合取.本文将一个合取式与至多一个析取式的合取称为搜索项;接下来,可以用合取式 $A\wedge B$ 作为输入进行单项条件社区搜索;最后,用析取式 $C\vee D$ 来对搜索得到的社区进行判别,使得所需单项条件社区搜索的次数减1.这种合并公共节点变量、把多个单项条件社区搜索一起进行的操作能够有效地提高整体搜索效率,减少时间开销.

为有效提取各合取式中的公共变量,本文采取一种贪心的思想:首先统计每个节点变量在最简与或式的各合取式中出现的次数,找到出现次数最多的变量;接着,合并含有该变量的所有合取式;而后,用同样策略尝试合并剩余合取式,直至不能合并为止.具体提取过程见算法 1,其中,*merge* 方法用于提取若干个合取式的公共节点变量,并合并形成新搜索项.

算法 1. 提取公共变量 *GetCommon*(*searches*,*v_set*).

输入:合取式集合 *searches*,节点变量集合 *v_set*;

输出:新的搜索项集合 *new_searches*.

```
1.  if searches==∅ then
2.      return ∅
3.  count←长度为|v_set|的数组   //记录变量在搜索项中出现次数
4.  初始化 count 的每个元素为 0
5.  for search in searches do
6.      for v in v_set do
7.          if v∈search then
8.              count[v]+=1
9.  v_max←argmax_{v∈v_set}(count[v])
10. common_searches←searches 中包含 v_max 的搜索项集合
11. new_search←merge(common_searches)   //merge 表示合并含有公共节点变量的搜索项
12. 从 searches 中删去 common_searches
13. return {new_search}∪GetCommon(searches,v_set)
```

令最简与或式涉及到的所有变量个数为 n,搜索项的数量为 x,则算法 1 进行统计节点变量出现次数的时间复杂度是 $O(nx)$,找到出现次数最多的公共变量的时间复杂度是 $O(n)$,合并含有该变量的合取式的时间复杂度是 $O(nx)$.

结合简化搜索条件的策略,改进后的条件社区搜索通用框架包括如下步骤:

1. 枚举所有满足条件的节点变量取值组合,得出与搜索条件等价的主析取范式;
2. 利用 QM 算法把主析取范式转化为最简与或式;
3. 合并最简与或式中的公共节点变量;
4. 对于每一搜索项,将其合取式作为单项条件社区搜索的输入并执行,将其析取式用于结果判别;
5. 合并各单项条件社区搜索的结果.

3 单项条件社区搜索

为解决 SCCS,第 3.1 节首先提出“社区搜索+过滤”的方法,第 3.2 节给出基于标签传播给点加权的方法.

方便起见,本文称社区中必须出现的节点为必要节点,社区中不能出现的节点为禁止节点.由定理 1,SCCS 仅需考虑合取式形式的搜索条件.因此,可以根据搜索条件中是否存在逻辑非来把对应的节点集合划分为一个

必要节点集和一个禁止节点集.本文将以这两个集合作为 SCCS 的输入.

由定义 3 可知,CCS 需要给定社区定义C.在相关研究中,基于 k-core 的社区定义是一种比较常见的方式[10,11],于是,本节基于 k-core 定义来介绍所提方法.显然,对其他社区定义,本文所提方法亦可修改适用.

定义 5(k-core 社区). 给定图 $G=(V,E)$和常数 k,节点集 $H\subset V$,称 H 为 k-core 社区,当且仅当 H 的导出子图 $G[H]$ 连通且 $\min\{\deg_{G[H]}(v)|v\in H\}\geqslant k$.

3.1 “社区搜索+过滤”的方法

针对社区搜索算法没有考虑禁止节点的情况,本节提出预先搜索社区(search-first,简称 SF)、预先过滤节点(filter-first,简称 FF)和搜索时筛出(on-the-fly,简称 OTF)等 3 种不同策略.这 3 种策略都采用“社区搜索+过滤”的形式,即:用必要节点进行社区搜索,用禁止节点进行过滤.三者的区别在于,过滤步骤分别安排在社区搜索之后、之前和过程中.

3.1.1 基于 k-core 的社区搜索方法 FindCore

为便于说明“社区搜索+过滤”方法的 3 种不同策略(SF,FF 和 OTF),本小节提出基于 k-core 的社区搜索算法 FindCore(见算法 2),用于 3 种策略的社区搜索步骤.

算法 2. 基于 k-core 的社区搜索算法 FindCore.

输入:$G=(V,E)$,v_set,k;

输出:包含 v_set 的 k-core 社区 C.

```
1.  C←v_set,visited←v_set
2.  按 G[C]的连通分量把 C 进行划分,得到 p={set_1,…,set_n}
    //G[C]表示 C 的导出子图,set_i 为第 i 个连通分量包含的节点集.
3.  Candidates←∅
4.  d_min←C 中最小点度数
5.  for v in v_set do
6.    if |N(v)|<k then
7.      return ∅   //此时不可能形成 k-core
8.    Candidates=Candidates∪(N(v)−visited)   //N(v)表示点 v 的邻居节点
9.  while |p|>1 or |d_min|<k do
10.   if Candidates==∅ then
11.     C←∅   //此时无法连通,或无法形成 k-core
12.     break
13.   if |C|>search_limit then
14.     C←∅   //超过了搜索上限
15.     break
16.   for c in Candidates do
17.     if |p|>1 then
18.       记录 c 与各连通分量的连接数 a
19.     记录 c 与 v_set 内点的连接数 b
20.     记录 c 自身度数 d
21.   if |p|>1 then
22.     对 Candidates 以 a,b,d 进行多关键字排序
23.   else
24.     对 Candidates 以 b,d 进行多关键字排序
25.   c_new←Candidates.pop(0)
```

```
26.  visited←visited∪{c_new}
27.  if |N(c_new)≥k| then
28.      C←C∪{c_new}
29.      for c in N(c_new) do
30.          if c not in visited then
31.              Candidates←Candidates∪{c}
32.      Update p
33.      Update d_min
34. if C==∅ then return global_search(G,v_set,k)
35. else return C
```

FindCore 算法从输入节点集出发,通过从邻居节点集中选取合适的节点进行扩展,得到 k-core 社区.首先判别社区的连通性要求,计算当前节点集导出子图的连通分量(算法 2 第 2 行),优先加入连接最多连通分量的节点(第 17 行、第 18 行和第 21 行、第 22 行).其次,增加当前节点集的最小点度数,优先加入与当前节点集连接数最多的节点.若连接数相同,则优先加入度数最大的节点(第 19 行、第 20 行和第 23 行、第 24 行).重复扩展过程,直至当前节点集成为连通的 k-core 社区.需要注意的是:为保证算法的正确性,在加入点的过程中需检查新增节点的度数(第 6 行、第 7 行和第 27 行),如果新增节点在原图中度数小于 k,那么它不可能是 k-core 社区的一员.同时,通过循环终止条件保证了结果社区满足 k-core 要求(第 9 行).此外,设置了搜索节点个数的上限 *search_limit*(第 13 行),以便提前终止局部搜索的过程.这一设置是为了在搜索条件所要求的 k-core 社区可能不存在时,防止无谓的节点遍历.当局部向外扩展的方法找不到合适的 k-core 社区时,调用 *global_search* 方法(第 34 行),即从全局出发不断迭代,删除度数小于 k 的节点的方法来寻找 k-core 社区.文献[11]中的引理 2 保证了这一步骤的正确性.

例 4:令图 2 为当前网络,取 k=2,社区搜索输入的节点集为{A,B,C,D}.算法 2 首先计算输入节点集导出子图的两个连通分量{A,B}和{C,D}.接着,将邻居节点{E,F,G}加入候选集.由于点 G 与两个连通分量相连,而 E 和 F 各连接一个,于是,点 G 在第 1 次扩展时加入节点集.然后,点 E 和 F 依次加入,从而增加了最小节点度数.最终,得到了{A,B,C,D,E,F,G}这一 2-core 社区.

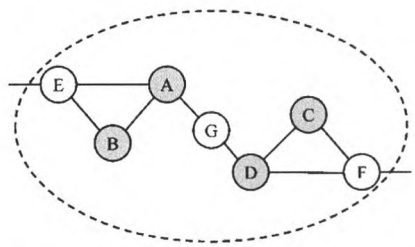

Fig.2 An example for k-core based community search

图 2 社区搜索的示例

算法 2 的时间复杂度是 $O(m+n)$,其中,m 为边数量,n 为节点数量.这是因为算法在最坏情况下可能要遍历所有节点和边.

3.1.2 预先搜索的策略

本小节介绍预先搜索(SF)的策略,其过程分为 3 步:(1) 用必要节点进行社区搜索得到社区结果;(2) 检查结果内部是否存在禁止节点,若存在,则删除其中的禁止节点;(3) 检查剩余社区结果是否还满足社区定义,若不满足,则需调整剩余社区结果.一种通用调整方式是把必要节点作为输入在剩余社区上再进行一次社区搜索.此外,也可以针对特定社区定义设计调整方案.

以 FindCore 算法为例,SF 先用该算法得到初始 k-core 社区,再检查社区内的禁止节点.如果没有找到禁止节

点,就可以直接将它作为结果返回.反之则删除禁止节点,检查社区是否还满足 k-core 定义:如果不满足,则可以通过依次删除社区内度数最低的节点来调整社区结构,尝试得到 k-core 社区.

需要注意的是,该策略可能在某些情况下得不到结果社区.以 4-core 作为社区定义,假设通过预先搜索得到了一个 4 完全图结构的社区结果,如果其中含有 3 个禁止节点,那么社区剩余部分仅有 1 个节点,无法再得到合理社区结构.因此,该策略有一定的局限性,在实验中被视作其他方法的基准.

3.1.3 预先过滤的策略

本节介绍预先过滤(FF)的策略,其过程分为 2 步:(1) 删除网络图中所有禁止节点;(2) 用必要节点集作为输入,在剩余网络图上进行社区搜索.因为在第 1 步过滤了禁止节点,所以保证了第 2 步得到的社区能满足搜索条件.

FF 策略的缺点是可能存在不必要的节点删除操作.若禁止节点原本就远离最终的结果社区,则不需要在社区搜索的过程中对其进行访问.尤其是当网络图规模较大且禁止节点较多时,这类不必要的删除操作就会影响时间开销.原因在于,从图中删除一个节点会涉及对节点自身及邻接边的访问.

3.1.4 搜索时筛出的策略

本节介绍搜索时筛出(OTF)的策略,即:在社区搜索过程中避开禁止节点,不再需要对结果进行检查,直接得到符合搜索条件的社区.

以 FindCore 算法为例,用 in_set 和 out_set 分别表示必要节点集和禁止节点集,算法 3 给出了应用 OTF 策略的 FindCore 算法伪码.它在算法 2 的基础上仅修改了第 1 行、第 20 行和第 34 行.其中,第 1 行避免了将禁止节点加入候选集,第 20 行消除了禁止节点的度数贡献,第 34 行的 $global\_search'$ 方法在迭代删除度数小于等于 k 的节点前先删除了禁止节点.于是,算法 3 保证了得到的 k-core 社区能满足搜索条件.

算法 3. 基于 OTF 策略的 FindCore 算法.

输入:$G=(V,E)$,in_set,out_set,k;

输出:社区 $C(in\_set\subset C, out\_set\cap C=\varnothing)$.

```
1.  C←in_set, visited←in_set∪out_set
    …  //第 2 行~第 9 行与算法 2 中的相同
9.  while |p|>1 or d_min<k do
    …  //第 10 行~第 19 行与算法 2 中的相同
20.     记录 c 与 V−out_set 内点的连接数 d
    …  //第 21 行~第 33 行与算法 2 中的相同
34. if C==∅ then return global_search'(G,in_set,out_set,k)
35. else return C
```

与前两种策略相比,OTF 策略修改了社区搜索算法,以便在搜索过程中检查禁止节点.该策略在保证得到正确结果的同时还排除了冗余的删除操作,因此相比而言更为高效.此外,基于 FF 策略和 OTF 策略使用 FindCore 算法得到的社区结果相同,易证得如下定理:

定理 2. 使用 FindCore 算法,通过 FF 策略和 OTF 策略所得单项条件社区搜索的结果是相同的.

证明:假定给出合理的单项搜索条件,即,从条件中得出的必要节点集和禁止节点集不相交.首先,在候选集节点提取过程中,FF 策略下禁止节点被预先删除而不会进入候选集,而在 OTF 策略下,禁止节点会在候选集扩展时被过滤掉,因此两种策略下的候选集相同.其次,在从候选集选取优先加入的节点的过程中,FF 策略里点的度数是删除禁止节点后计算的,这一度数与 OTF 策略下统计的不在禁止节点集中的邻居节点个数是相等的,因此两种策略下,每次候选集中选出的最优节点是一样的.综上,这两种策略得到的社区结果是相同的. □

3.2 基于标签传播思想给点加权的方法

例 5:如图 3 所示,一个示例网络中的节点代表个体,边表示好友关系.假定个体 A 拟建立一个不包含 B 的讨论小组,于是对应的 CCS 为寻找满足搜索条件 $A\wedge\neg B$ 的社区,即 A 为必要节点、B 为禁止节点.若以 3-core 作为

社区结构,则通过“社区搜索+过滤”的方法(FF 策略)得到的一个社区结果为虚线包围的子网络.因为该社区中的 B_1~B_3 和 B 都是好友关系,所以 B 有可能通过其中的某个/些好友了解该讨论小组的情况,而这是 A 所不希望的.

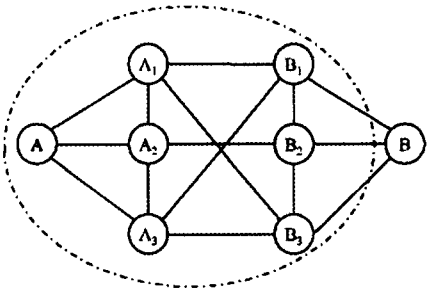

Fig.3 Result of community+search method (FF)

图 3 FF 策略下“社区搜索+过滤”方法的结果

例 5 表明,“社区搜索+过滤”的方法得到的社区虽然能够满足社区定义和搜索条件,但是难以体现出社区中的节点应尽量远离禁止节点,同时尽量靠近必要节点的潜在需求.简便起见,文中以倾向性指代节点相对于必要节点和禁止节点的接近程度.一个节点离禁止节点越远、离必要节点越近,则该节点的倾向性越大.本节提出了基于标签传播给点加权的方法(WLP),用节点的权重来表示节点的倾向性.

3.2.1 基于标签传播的加权过程

利用标签传播进行社区发现的方法包括以下步骤:首先为每个节点赋予唯一的初始标签,接着逐轮迭代,把每个节点的标签迭代更新为其邻居节点中出现次数最多的那一个.经过数轮迭代后,最终把标签相同的节点归入同一社区.

基于上述思想,本节提出一种为节点加权的方法.初始时,先将节点权重设置如下:

$$W_0(v)=\begin{cases}1, & \text{必要节点}\\ -1, & \text{禁止节点}\\ 0, & \text{其他节点}\end{cases} \tag{1}$$

接着对节点的权重进行迭代更新,但保持必要节点和禁止节点的权重固定不变.一个节点的新权重决定于其所有邻居节点的上一轮权重.为了使得新权重落在[−1,1]区间内,以如下方式更新权重:

$$W_{i+1}(v)=\begin{cases}1, & \text{必要节点}\\ -1, & \text{禁止节点}\\ \dfrac{\sum_{v'\in N(v)}W_i(v')}{|N(v)|}, & \text{其他节点}\end{cases} \tag{2}$$

其中,$N(v)$表示节点 v 的邻居节点,$W_i(v)$表示节点 v 的第 i 轮权重.在第 1 轮迭代中,必要节点和禁止节点的一跳邻居被赋予新权重.此时,与较多必要节点相连的节点权重变大,和较多禁止节点相连的节点权重变小,体现出各自的倾向性大小.在第γ轮迭代中,这种倾向性会传递给必要节点和禁止节点的γ跳邻居.

接下来,通过给定阈值λ排除权重小于λ的节点,这样可以去除那些不宜出现在社区中的倾向性较小的节点.经过阈值筛选后的节点所形成的导出子图可能不满足任何的社区定义,为此,在最后需要调整该子图的结构,例如通过 FindCore 算法在该子图里寻找 k-core 社区.综上,WLP 方法分为如下 3 个步骤.

(1) 按标签传播的方式为各个节点赋予权重;

(2) 以给定阈值筛选出权重较大的节点;

(3) 在筛出节点集的导出子图上进行社区搜索.

相对于“社区搜索+过滤”方法,WLP 方法强调了条件中必要节点和禁止节点对周围节点的影响,排除了那

些与禁止节点过近的节点,保留了与必要节点更接近即倾向性更大的节点.

WLP 方法如算法 4 所示,其中,第 2 行~第 15 行是权重赋值过程,第 16 行~第 19 行是筛选过程,最后一行调用 FindCore 在导出子图上进行社区搜索.需要注意的是,最后一行也可以用任意的社区搜索算法进行替换.

因为在权重赋值时把禁止节点权重固定为−1,所以取$\lambda>-1$就可以保证筛选出的节点集不包含禁止节点,从而满足搜索条件,保证了 WLP 方法的正确性.

算法 4. WLP 方法.

输入:$G=(V,E)$,in_set,out_set,k,λ,γ //λ表示权重的阈值,γ表示迭代次数;

输出:社区 $C(in\_set\subset C, out\_set\cap C=\varnothing)$.

1. $W,W'\leftarrow$初始化两个长度为$|V|$的数组 //W,W'用于保存权重
2. **for** v **in** V **do**
3. **if** $v\in in\_set$ **then**
4. $W[v]=1$
5. **else if** $v\in out\_set$ **then**
6. $W[v]=-1$
7. **else**
8. $W[v]=0$
9. **for** $i\leftarrow 0$ to γ **do**
10. **for** v **in** V **do**
11. **if** $v\in in\_set$ **or** $v\in out\_set$ **then**
12. $W'[v]=W[v]$
13. **else**
14. $W'[v]=\dfrac{\sum_{v'\in N(v)}W[v']}{|N(v)|}$ //$N(v)$表示点 v 的邻居节点
15. $W\leftarrow W'$
16. $V'\leftarrow\varnothing$
17. **for** v **in** V:
18. **if** $W[v]>\lambda$ **then**
19. $V'\leftarrow V'\cup\{v\}$
20. **return** $FindCore(G[V'],in\_set,\varnothing,k)$ //调用算法 3,$G[V']$表示 V'的导出子图

例 6:图 4 和图 5 展示了通过 WLP 方法解决图 3 中 CCS 的过程.在第 1 轮迭代后,各个点的权重更新为其邻居点的权重均值,得到了图 4.重复迭代,最后得到图 5.设定阈值为 0,可将节点 A,A_1,A_2 和 A_3 预先选出;接着,在这 4 个节点的导出子图上应用 FindCore 算法.可以发现,虚线圈出部分已经是一个 2-core 社区.相比于图 3,它规模更小,结构更紧凑,并且社区内部的成员更靠近节点 A,远离节点 B.

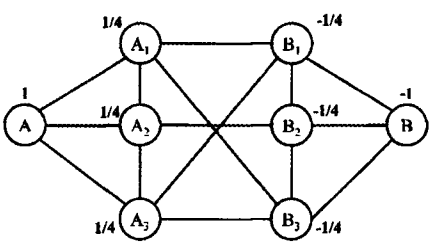

Fig.4 Result of the first propagation

图 4 第 1 轮加权结果

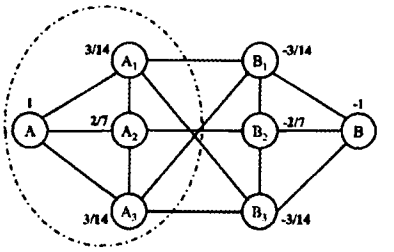

Fig.5 Result of conditional community search (WLP)

图 5 通过 WLP 进行条件社区搜索的结果

3.2.2 WLP 方法的复杂度分析

令网络图中边数为 m,节点数为 n,迭代次数为γ,则通过标签传播来给点加权的过程的时间复杂度是 $O(n)+O(\gamma m)$,其中,初始赋权重标签的复杂度 $O(n)$.鉴于每一轮迭代的计算过程需要访问每个节点的所有邻居,于是,其时间复杂度为 $O(m)$.在实际计算中,可以仅维护权重不为 0 的节点数组,随着迭代,再不断添加而不必将所有点赋予初始值.这样做可以将上述过程的复杂度降到 $O(\gamma m)$.

4 实验与分析

本节首先介绍实验环境与数据集,通过实验展示利用简化的搜索条件进行条件社区搜索的有效性,最后比较不同条件社区搜索方法的时间开销和结果社区质量.

4.1 实验环境与数据集

本文实验所用机器的 CPU 是 Intel Xeon E5-2650 2.0GHZ,内存大小 256GB,操作系统为 Windows Server 2008 R2.用 Python-3.6.1 实现了 3 种策略下(FF,OTF,SF)的"社区搜索+过滤"方法和 WLP 方法.

本文实验采用的真实数据集有 Football,DBLP,Amazon 和 Youtube,其中,Football 数据集来自 Khorasgani[44],其余数据集来自 Stanford Large Network Dataset Collection(http://snap.stanford.edu/data/),具体统计信息见表 1,均带有真实的社区分布.表 1 最后一列的 top5000 社区是指数据集提供的前 5 000 个质量最高的社区(评判标准因数据集而异,如 conductance,modularity 等,详见文献[45]).我们统计发现,DBLP,Amazon 和 Youtube 这 3 个数据集的 top5000 社区中都有 95%以上的社区规模在 50 个节点以下.因此在后续实验中,FindCore 算法的搜索上限 *search_limit* 设置为 50,以便提前终止局部搜索过程,提高算法效率.

Table 1 Datasets

表 1 数据集

网络	节点数|V|	边数|E|	社区数	50 节点以下社区占 top5000 比例(%)
Football	180	788	11	–
DBLP	317 080	1 049 866	13 477	98.7
Amazon	334 863	925 872	75 149	96.9
Youtube	1 134 890	2 987 624	8 385	95.3

为了研究简化搜索条件的方法在不同规模网络图上的效果,本文还采用人工网络生成工具 Lancichinetti-Fortunato-Radicchi(LFR)[46]来合成不同规模的网络图.具体地,合成网络图中节点数量 n 的变化范围是 10^4~10^5.LFR 工具的其他参数设置如下:节点平均度数 d 为 5,节点最大度数 $d_{\max}$ 为 50,最小社区规模 $c_{\min}$ 为 20,最大社区规模 $c_{\max}$ 为 100,拓扑结构混合参数 u 为 0.1.

4.2 社区结果的评价指标

通过参考现有社区发现和社区搜索的评价标准,本文选取 $F1$-measure 和 Q_l(局部模块度)来评估结果社区的质量,其中,$F1$-measure 衡量了结果社区的正确性,Q_l 评估了结果社区内部的紧密程度.

$F1$-measure 是准确率和召回率的调和平均值,用于衡量计算得到的结果社区与真实社区的接近程度,其计算公式为

$$F1\text{-measure}=\frac{2|C\cap C'|}{|C|+|C'|} \tag{3}$$

其中,C 代表计算出的结果社区,C'是真实社区.$F1$-measure 的值越接近 1,则计算结果越精确.需要注意的是,数据集提供的真实社区的获取方式不尽相同.例如:Football 数据集是依据球员的好友关系构建的网络,同一个俱乐部成员标定为同一社区;DBLP 是依据论文作者的协作关系构建的网络,同一个小组的成员标定为同一社区.

依据数据集给出的真实社区,一种很自然的假定是:当单项条件社区搜索中的必要节点都来自同一真实社区,而禁止节点都在该社区外时,数据集提供的真实社区就是对应单项条件社区搜索的真实社区.在其他情况下,例如必要节点和禁止节点都来自数据集提供的某个真实社区时,本文提出的"社区搜索+过滤"方法以及

WLP 方法仍有返回结果的可能.该结果往往预示着数据集提供的真实社区中存在着可以细分的子社区结构,但此时的真实社区无法预知.因此,在这种情形下就无法使用 *F*1-measure 进行正确性度量,而只关心所得社区的紧密性和社区内成员相对必要节点的远近,即,从合理性的角度去度量社区结果好坏.

局部模块度(local modularity)指一个子图内部所有的边数与原图中所有涉及到该子图中节点的边数量的比值[47],即:

$$Q_l = \frac{k_{in}}{k_{in} + k_{out}} \tag{4}$$

其中,k_{in} 表示社区内部的边数,k_{out} 表示社区内部和外部连接的边数.Q_l 越大,表明社区紧密性越好.

除此之外,为了评估社区内节点与必要节点的接近程度,本文设计了相对最短路径比这一新指标,即社区内节点分别与必要节点和禁止节点的平均最短路径距离之比(average shortest pathdistance ratio,简称 ASD-ratio):

$$ASD\text{-}ratio = \frac{|out\_set| \sum_{v_i \in C \setminus in\_set, v_j \in in\_set} dist(v_i, v_j)}{|in\_set| \sum_{v_i \in C \setminus in\_set, v_j \in out\_set} dist(v_i, v_j)} \tag{5}$$

其中,*in_set* 和 *out_set* 分别表示必要节点集和禁止节点集,$dist(v_i,v_j)$表示 v_i 和 v_j 之间的最短路径距离,*C* 表示社区结果.*ASD-ratio* 越小,意味着社区内成员与必要节点越近,与禁止节点越远.

4.3 搜索条件简化的有效性

第 2.3 节给出了简化搜索条件以减少单项条件社区搜索次数的方法.本节将对此进行实验验证.

根据第 2 节提出的条件社区搜索通用框架,无论搜索条件是否经过简化,该框架都需要先枚举所有满足原始搜索条件的变量组合,简化搜索条件的有效性在之后的步骤才体现出来.因此在本节实验中,搜索条件被设计为主析取范式的形式,相当于省去枚举变量取值过程的开销.该主析取范式包含 q 个搜索项,每个搜索项包含相同的 v 个节点变量.这 v 个节点变量按照均匀分布从网络图中随机选取.此外,对每个搜索项,按照均匀分布随机为部分节点变量添加了逻辑非运算符,表示禁止这部分节点出现在社区中.例如,$(A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge C)$表示一个包含 2 个搜索项、3 个节点变量 *A*,*B* 和 *C* 的主析取范式形式的搜索条件.因为对任意形式的搜索条件总能得出与其等价的主析取范式[40],范式中搜索项的个数和节点变量的个数允许任意指定,并且节点变量是随机选取的,所以上述搜索条件设计可以表达任意的搜索条件,其完备性得以保证.

本节以 OTF 方法为例来验证搜索条件简化的优势,其他 3 种方法的效果类似.公平起见,简化前后的条件社区搜索方法中的单项条件社区搜索步骤都应用了 FindCore 算法.为保证实验结果的普遍性,在不同的节点变量个数 v 和不同搜索项个数 q 下进行了多次对比.对固定的一组 v 和 q,重复 10 次实验,取时间开销的均值.FindCore 算法中的 k 值从 2~10 的范围内逐一尝试,保留使社区规模最大的 k 值.

图 6 展示了简化前后条件社区搜索方法的时间开销随搜索项个数 v 的变化情况,其中,简化后条件社区搜索方法的时间开销包含了化简过程自身的时间开销.如图 6(b)~图 6(d)所示:简化前的条件社区搜索方法的时间开销基本上和搜索项个数成正比,简化后的条件社区搜索方法在时间开销上具有明显优势.例如:针对 DBLP 数据集,当节点个数为 5、搜索项个数为 14 时,简化前的时间开销是简化后时间开销的 5 倍以上.这是因为固定节点数量后,搜索项越多,越容易引发搜索项冗余现象,也越容易出现公共搜索变量.于是,当冗余的搜索项数量增多时,简化后的条件社区搜索方法的提高效果更为明显.

图 7 展示了简化前后条件社区搜索方法的时间开销随节点变量个数 q 的变化情况,其中,简化后条件社区搜索方法的时间开销包含了化简过程自身的时间开销.如图 7(b)~图 7(d)所示:从时间开销随节点个数的变化趋势上看,节点个数增加并不会影响简化后的条件社区搜索方法的优势.例如在图 7(b)中,节点个数从 4 增加到 6,简化后时间开销稳定地为简化前时间开销的 1/4.

需要注意的是,图 6(a)和图 7(a)中存在简化后条件社区搜索方法的时间开销更高的情形(不过整体时间开销均在 0.3s 内).这是因为:一方面,图 6(a)和图 7(a)对应的 Football 数据集规模较小,于是运行社区搜索算法的开销较小;另一方面,社区搜索条件的化简过程也需要一定开销,条件越复杂,则化简过程自身需要越多的时间.因

此,当条件较复杂(比如搜索项数量大于 8)时,化简过程的时间开销占到较大比重,从而使简化后的社区搜索方法耗时更多.

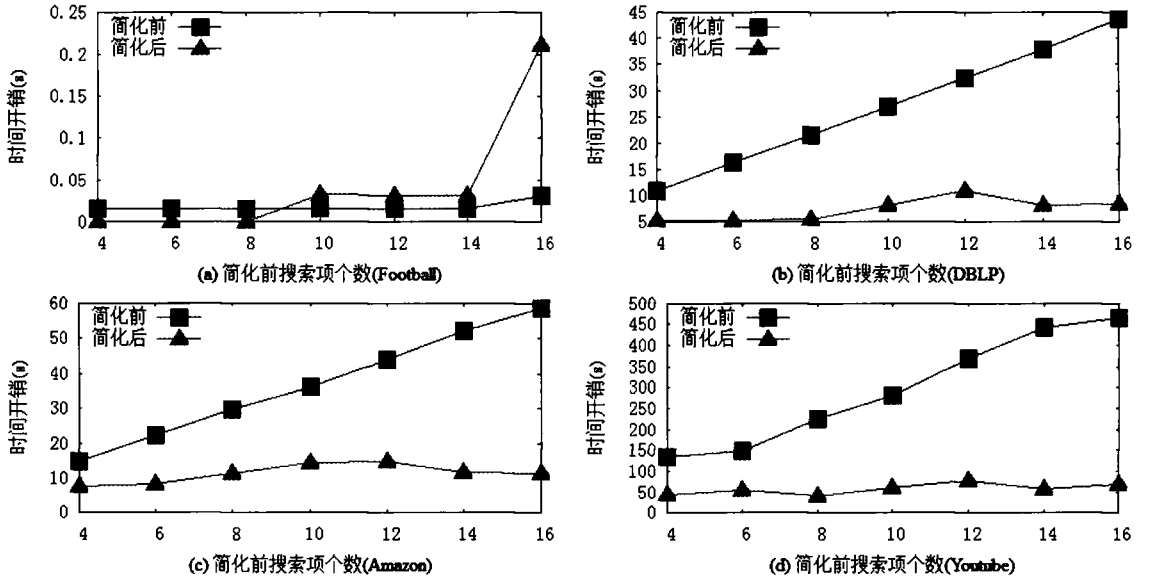

Fig.6 Time costs of conditional community search with different query numbers (# node variable v=5)

图 6 不同搜索项个数下条件社区搜索方法的时间开销(节点变量个数 v=5)

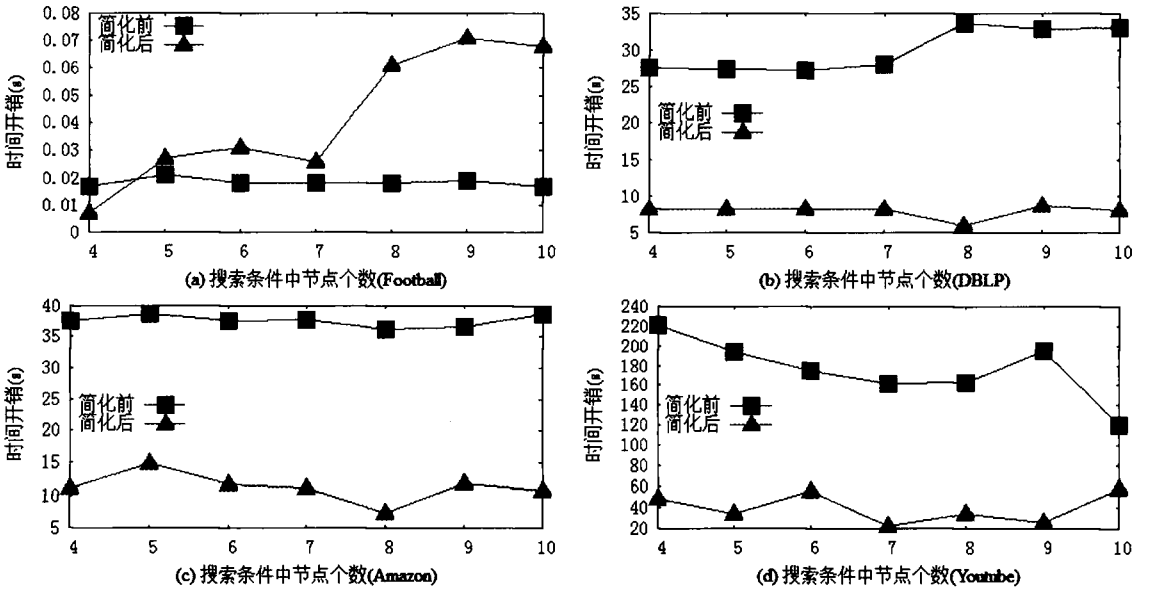

Fig.7 Time costs of conditional community search with different node numbers (# queries q=10)

图 7 不同节点个数下条件社区搜索方法的时间开销(简化前搜索项个数 q=10)

为了分析最简与或式化简程度与条件社区搜索方法的关系,给出冗余项数目的概念并展示冗余项数目对简化后条件社区搜索方法时间开销的影响.具体地,冗余项数目定义为化简为最简与或式前后减少的搜索项个数.图 8 展示了 DBLP 等 3 个数据集简化后条件社区搜索方法的时间开销随冗余项数目的变化情况,其中,节点变量个数 v 为 8,简化前搜索项个数 q 为 8.显然,随冗余项数目的增加,时间开销进一步得到缩减.需要说明的是,

因为 Football 数据集上的整体时间开销较小,所以没有进行展示.

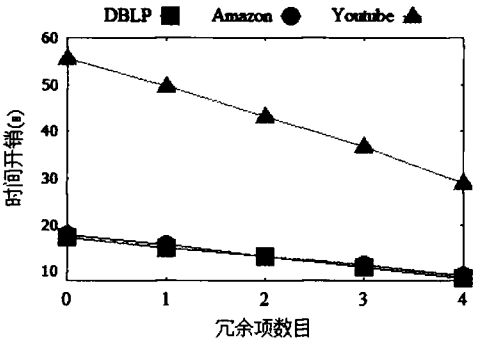

Fig.8 Effect of redundant items on the time cost

图 8 冗余项数目对时间开销的影响

图 9 对比了不同网络图规模下条件社区搜索方法的时间开销,其中,网络图的规模由图中的节点数量表示.对不同节点变量个数 v 和搜索项个数 q,简化前后条件社区搜索方法的时间开销都随网络图规模的增大而增加,并且简化后的条件社区搜索方法一直保持明显优势.实际上,社区规模大小只影响每次单项条件社区搜索的时间开销而不影响对搜索条件的化简过程,因此,化简过程在大规模网络图上依然适用.

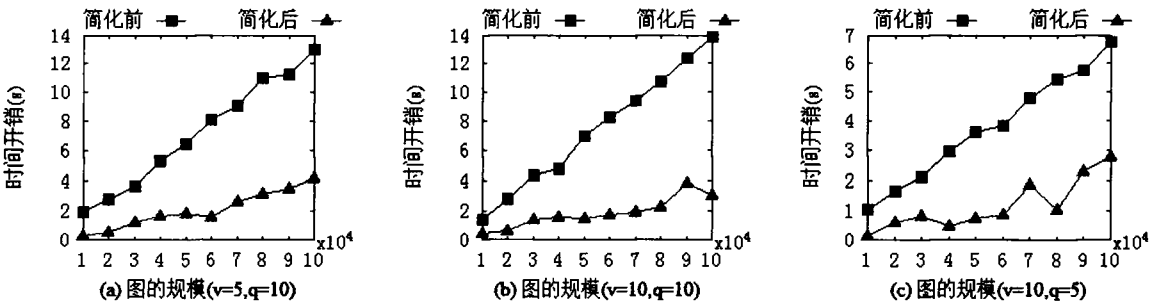

Fig.9 Time costs of conditional community search with different graph size

图 9 不同网络图规模下条件社区搜索的时间开销

4.4 单项条件社区搜索方法的对比实验

第 3 节提出了单项条件社区搜索方法,即"社区搜索+过滤"的方法(使用 3 种不同策略的方法分别记为 FF,OTF 和 SF)和基于标签传播给点加权的方法(记为 WLP).本节将从时间开销和社区结果质量角度对上述方法进行实验.

由定义 4 可知,单项条件社区搜索的输入条件只有一种可满足的取值组合,因此可以通过随机指定必要节点和禁止节点的个数来构造搜索条件.鉴于用户输入的节点总数通常较小,实验中设计了 5 种类型的单项社区搜索条件(见表 2),对应不同的必要节点和禁止节点个数.例如,编号 i 对应形如 $A \wedge B \wedge \neg C$ 的搜索条件,包含 2 个必要节点和 1 个禁止节点.在执行具体搜索实验时,节点变量替换为具体的节点编号,且每一种搜索条件对应 100 组不同的具体节点编号.不同方法中,每类搜索条件的时间开销、$F1$-measure、Q_l 以及 ASD-ratio 都是相关的 100 组实验的均值.

为公平起见,所有社区搜索算法均采用 FindCore 算法.对不同节点变量,k 值在 2~10 的范围内尝试,留下使社区规模最大的一个.

Table 2 Design of search conditions

表 2 搜索条件设计

搜索条件编号	必要节点数	禁止节点数
i	2	1
ii	2	3
iii	3	2
iv	3	4
v	4	5

图 10 展示了当搜索条件为 iii,WLP 方法中的阈值λ在[0,0.4]范围内取值时所得社区 $F1$-measure 的变化情况.当λ超过 0.2 时,在 DBLP 上难以找到合适社区,而在 Football 上社区准确性下降.需要说明的是,在两个数据集上,$-1<\lambda<0$ 时 $F1$-measure 的值与$\lambda=0$ 时 $F1$-measure 的值相等;当$\lambda>0.4$ 时,得到的 $F1$-measure 的值均为 0,所以在图中没有显示上述范围内的变化情况.考虑到取较大阈值可减小社区搜索范围,因此在后续实验中,统一将阈值λ设置为 0.2.

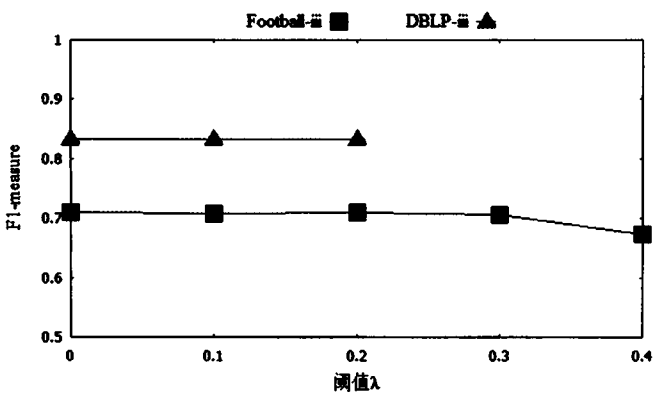

Fig.10 $F1$-measure with different thresholds of WLP method

图 10 不同阈值下 WLP 方法得到的 $F1$-measure

根据第 3.2.1 节中 WLP 方法的具体过程,第γ轮迭代中相对于必要节点和禁止节点的倾向性会传递给必要节点和禁止节点的γ跳邻居.此外,考虑到六度分离的原则,当迭代至 6 次时,网络中的大部分节点已经得到了能反映出倾向性的权重.因此,实验中的迭代次数γ被设置为 6.

图 11 对比了 4 种不同方法的时间开销.图 11(a)和图 11(b)分别是在 Football 和 DBLP 数据集上取得的实验结果,横坐标编号对应搜索条件.

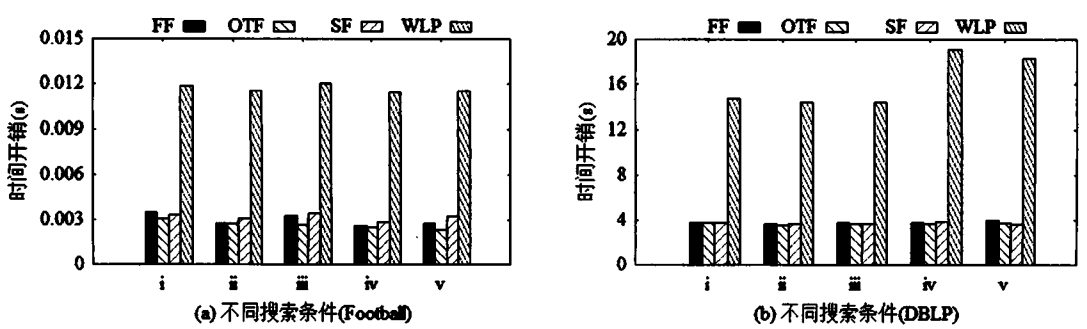

Fig.11 Time costsof different singleconditionalcommunity search methods

图 11 不同单项条件社区搜索方法的时间开销对比

因为 WLP 方法需要进行节点权重的计算,且该计算涉及图中所有边,较为费时,所以其时间开销最大."社区搜索+过滤"方法的 3 种策略 FF,OTF 和 SF 的时间开销相近,其中,OTF 策略相对稍好.这是因为 FF 策略存在部

分冗余删除,SF 策略需要进一步调整社区结构,而 OTF 策略在搜索过程中过滤掉禁止节点,从而避免了另两种策略在效率上的不足.

图 12 展示了 4 种不同方法在 Football 和 DBLP 数据集上所得社区结果的 $F1$-measure.在每次单项条件社区搜索的具体实验中,要求其搜索条件中的必要节点来自同一实际社区,同时禁止节点来自该社区外.这样就可以根据数据集附带的真实社区结果计算 $F1$-measure.从图 12(a)和图 12(b)可以看出:无论是 Football 还是 DBLP 数据集,3 种不同策略下"社区搜索+过滤"方法得到的 $F1$-measure 都基本相同.这表明给定符合真实社区分布的搜索条件,这 3 种策略都能找到相同准确程度的社区结果.WLP 方法得到的社区搜索结果在多数搜索条件下有更好的准确性,在少数条件下准确性不如其他方法,如图 12(a)的 ii 和图 12(b)的 i,iii 和 iv,但是相对差异并不明显.这是由于 WLP 方法的结果受到禁止节点的影响,从而产生波动.例如:当禁止节点离真实社区较近时,WLP 方法一般会得到比真实社区更小的社区.这是因为在真实社区内部,有部分与禁止节点相连的节点会被阈值过滤排除在外.这实际上是为使社区内成员与必要节点更接近而付出的必要开销.

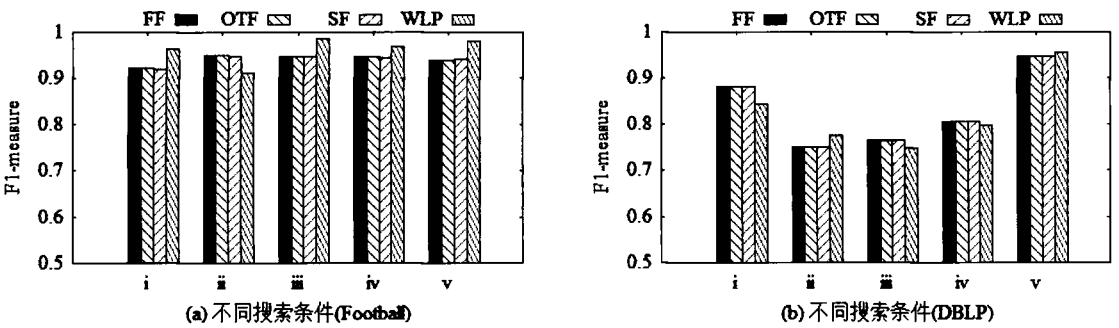

Fig.12 $F1$-measure of different singleconditional community search methods

图 12 不同单项条件社区搜索方法的 $F1$-measure 对比

图 13 对比了通过不同方法所得结果社区的局部模块度 Q_l,其中,图 13(b)的搜索条件中,禁止节点和必要节点来自同一社区,即搜索条件和数据集提供的真实结果有冲突,而图 13(a)则不存在这样的冲突.在有冲突的情形下,容易发现:在"社区搜索+过滤"方法的 3 种不同策略中,SF 所得结果对应的 Q_l 较低,而 FF 和 OTF 的 Q_l 相对较高.这是因为图中显示的 Q_l 为多次实验的均值,在搜索条件存在冲突的情形下,SF 在调整社区结构时可能无法得到社区结果,此时的 Q_l 被记为 0,从而导致 SF 的平均 Q_l 降低.在搜索条件不存在冲突时,这 3 种策略所得社区结果具有相同紧密程度.通过 WLP 方法得到的社区,在紧密程度上由于受禁止节点的影响存在较大波动.此外,图 12 和图 13 的结果也能验证定理 2,即,应用 FF 策略和 OTF 策略的 FindCore 算法得到的结果是相同的.

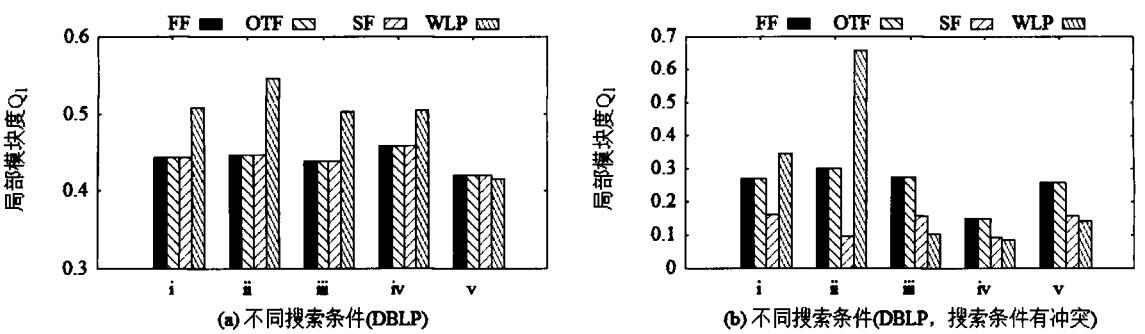

Fig.13 Q_l of singleconditional community search methods

图 13 不同单项条件社区搜索方法的 Q_l 对比

图 14 展示了不同方法所得结果社区的 ASD-ratio.显然,WLP 方法在两个数据集和不同的搜索条件下都具

有最小的 ASD-ratio.这表明 WLP 方法能够充分考虑必要节点和禁止节点的影响,得到社区成员与必要节点更接近的社区.

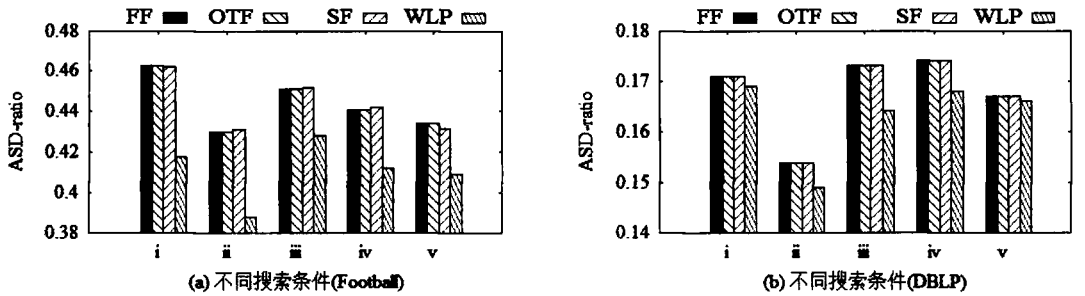

Fig.14 ASD-ratio of different singleconditional community search methods

图 14 不同单项条件社区搜索方法的 ASD-ratio 对比

上述实验结果表明:采用 FF 和 OTF 策略的"社区搜索+过滤"方法可以有效处理 CCS,使用 OTF 策略则可略微提升效率.WLP 方法尽管需要更多的时间开销,所得社区在准确性和紧密程度上存在波动,但是其结果能够体现出社区内成员对于必要节点的倾向性,排除与禁止节点相近的节点,使社区内成员与必要节点更接近,且在 ASD-ratio 上最优,具有较高的应用价值.

5 结束语

条件社区搜索问题是在传统的社区搜索问题基础上,结合实际需求提出的新问题,它包含了现有社区搜索问题,并且考虑了特定节点能否存在于社区中等复杂条件约束.

本文给出了 CCS 的形式化定义,并使用布尔表达式表示搜索条件.在此基础上,本文提出了解决 CCS 的通用框架,将 CCS 分解为多个 SCCS 来处理,并通过简化搜索条件对通用框架进行了优化.对于 SCCS,本文提出了"社区搜索+过滤"的方法(包括 FF,OTF 和 SF 这 3 种策略)和基于标签传播给点加权的 WLP 方法.通过在 4 个真实数据集上的大量实验,比较了这些方法的时间开销和社区结果.实验结果表明,使用 OTF 策略的"社区搜索+过滤"方法在时间开销和社区结果上具有优势.如果考虑用户在使用条件社区搜索时让社区成员远离禁止节点的需求,那么 WLP 方法能够充分考虑必要节点和禁止节点的影响,找到社区内成员与必要节点更接近的社区结果.

条件社区搜索的研究尚处在探索阶段,后续会考虑向 CCS 中引入更多类型的社区定义.对于 SCCS,希望找到一种新的社区搜索算法,使得社区结构在准确性和紧密程度上达到最优的同时也能考虑必要节点和禁止节点的影响.

References:

[1] Fortunato S, Hric D. Community detection in networks: A user guide. Physics Reports, 2016,659:1–44. [doi: 10.1016/j.physrep.2016.09.002]

[2] Girvan M, Newman MEJ. Community structure in social and biological networks. Proc. of the National Academy of Sciences, 2002, 99(12):7821–7826. [doi: 10.1073/pnas.122653799]

[3] Dourisboure Y, Geraci F, Pellegrini M. Extraction and classification of dense communities in the Web. In: Proc. of the 16th Int'l Conf. on World Wide Web. ACM Press, 2007. 461–470. [doi: 10.1145/1242572.1242635]

[4] Tang L, Liu H. Community detection and mining in social media. Synthesis Lectures on Data Mining and Knowledge Discovery, 2010,2(1):1–137. [doi: 10.2200/s00298ed1v01y201009dmk003]

[5] Ríos SA, Muñoz R. Dark Web portal overlapping community detection based on topic models. In: Proc. of the ACM SIGKDD Workshop on Intelligence and Security Informatics. ACM Press, 2012. 2. [doi: 10.1145/2331791.2331793]

[6] Chen J, Yuan B. Detecting functional modules in the yeast protein–protein interaction network. Bioinformatics, 2006,22(18): 2283–2290. [doi: 10.1093/bioinformatics/btl370]

[7] Shan J, Shen DR, Kou Y, Nie TZ, Yu G. Approach for hot spread node selection based on overlapping community search. Ruan Jian Xue Bao/Journal of Software, 2017,28(2):326–340 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5117.htm [doi: 10.13328/j.cnki.jos.005117]

[8] Qian J, Wang CK, Guo GY. Community based node betweenness centrality updating algorithms in dynamic networks. Ruan Jian Xue Bao/Journal of Software, 2018,29(3):853–868 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5457.htm [doi: 10.13328/j.cnki.jos.005457]

[9] Cui W, Xiao Y, Wang H, *et al.* Online search of overlapping communities. In: Proc. of the 2013 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2013. 277–288. [doi: 10.1145/2463676.2463722]

[10] Sozio M, Gionis A. The community-search problem and how to plan a successful cocktail party. In: Proc. of the 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2010. 939–948. [doi: 10.1145/1835804.1835923]

[11] Cui W, Xiao Y, Wang H, *et al.* Local search of communities in large graphs. In: Proc. of the 2014 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2014. 991–1002. [doi: 10.1145/2588555.2612179]

[12] Huang X, Cheng H, Qin L, *et al.* Querying *k*-truss community in large and dynamic graphs. In: Proc. of the 2014 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2014. 1311–1322. [doi: 10.1145/2588555.2610495]

[13] Akbas E, Zhao P. Truss-based community search: A truss-equivalence based indexing approach. Proc. of the VLDB Endowment, 2017,10(11):1298–1309. [doi: 10.14778/3137628.3137640]

[14] Huang X, Lakshmanan LVS, Xu J. Community search over big graphs: Models, algorithms, and opportunities. In: Proc. of the 33rd Int'l Conf. on Data Engineering (ICDE). IEEE, 2017. 1451–1454. [doi: 10.1109/icde.2017.211]

[15] Shang J, Wang C, Wang C, *et al.* An attribute-based community search method with graph refining. The Journal of Supercomputing, 2017:1–28. [doi: 10.1007/s11227-017-1976-z]

[16] Fang Y, Cheng R, Luo S, *et al.* Effective community search for large attributed graphs. Proc. of the VLDB Endowment, 2016,9(12): 1233–1244. [doi: 10.14778/2994509.2994538]

[17] Huang X, Lakshmanan LVS. Attribute-driven community search. Proc. of the VLDB Endowment, 2017,10(9):949–960. [doi: 10.14778/3099622.3099626]

[18] Wang M, Wang C, Yu JX, *et al.* Community detection in social networks: An in-depth benchmarking study with a procedure-oriented framework. Proc. of the VLDB Endowment, 2015,8(10):998–1009. [doi: 10.14778/2794367.2794370]

[19] Kernighan BW, Lin S. An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal, 1970,49(2): 291–307. [doi: 10.1002/j.1538-7305.1970.tb01770.x]

[20] Prat-Pérez A, Dominguez-Sal D, Larriba-Pey JL. High quality, scalable and parallel community detection for large real graphs. In: Proc. of the 23rd Int'l Conf. on World Wide Web. ACM Press, 2014. 225–236. [doi: 10.1145/2566486.2568010]

[21] Zhao F, Tung AKH. Large scale cohesive subgraphs discovery for social network visual analysis. Proc. of the VLDB Endowment, 2012,6(2):85–96. [doi: 10.14778/2535568.2448942]

[22] Newman MEJ. Fast algorithm for detecting community structure in networks. Physical Review E, 2004,69(6):066133. [doi: doi.org/10.1103/physreve.69.066133]

[23] Clauset A, Newman MEJ, Moore C. Finding community structure in very large networks. Physical Review E, 2004,70(6):066111. [doi: 10.1103/physreve.70.066111]

[24] Radicchi F, Castellano C, Cecconi F, *et al.* Defining and identifying communities in networks. Proc. of the National Academy of Sciences of the United States of America, 2004,101(9):2658–2663. [doi: 10.1073/pnas.0400054101]

[25] Newman MEJ, Girvan M. Finding and evaluating community structure in networks. Physical Review E, 2004,69(2):026113. [doi: 10.1103/physreve.69.026113]

[26] Schuetz P, Caflisch A. Multistep greedy algorithm identifies community structure in real-world and computer-generated networks. Physical Review E, 2008,78(2):026112. [doi: 10.1103/physreve.78.026112]

[27] Qiao SJ, Han N, Zhang KF, Zou L, Wang HZ, Gutierrez LA. Algorithm for detecting overlapping communities from complex network big data. Ruan Jian Xue Bao/Journal of Software, 2017,28(3):631−647 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5155.htm [doi: 10.13328/j.cnki.jos.005155]

[28] Donath WE, Hoffman AJ. Lower bounds for the partitioning of graphs. IBM Journal of Research and Development, 1973,17(5): 420−425. [doi: 10.1142/9789812796936_0044]

[29] Shi J, Malik J. Normalized cuts and image segmentation. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2000,22(8): 888−905. [doi: 10.1109/34.868688]

[30] Ng AY, Jordan MI, Weiss Y. On spectral clustering: analysis and an algorithm. Proc. of the Nips, 2001,14:849−856.

[31] Mahmood A, Small M, Al-Maadeed SA, *et al*. Using geodesic space density gradients for network community detection. IEEE Trans. on Knowledge and Data Engineering, 2017,29(4):921−935. [doi: 10.1109/tkde.2016.2632716]

[32] Pons P, Latapy M. Computing communities in large networks using random walks. In: Proc. of the Int'l Symp. on Computer and Information Sciences. Berlin, Heidelberg: Springer-Verlag, 2005. 284−293. [doi: 10.1007/11569596_31]

[33] Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. Proc. of the National Academy of Sciences, 2008,105(4):1118−1123. [doi: 10.1073/pnas.0706851105]

[34] Shang JW, Wang CK, Xin X, Ying X. Community detection algorithm based on deep sparse autoencoder. Ruan Jian Xue Bao/Journal of Software, 2017,28(3):648−662 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5165.htm [doi: 10.13328/j.cnki.jos.005165]

[35] Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations. In: Proc. of the 20th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2014. 701−710. [doi: 10.1145/2623330.2623732]

[36] Cao S, Lu W, Xu Q. Grarep: Learning graph representations with global structural information. In: Proc. of the 24th ACM Int'l Conf. on Information and Knowledge Management. ACM Press, 2015. 891−900. [doi: 10.1145/2806416.2806512]

[37] Raghavan UN, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks. Physical Review E, 2007,76(3):036106. [doi: 10.1103/physreve.76.036106]

[38] Leung IXY, Hui P, Lio P, *et al*. Towards real-time community detection in large networks. Physical Review E, 2009,79(6):066107. [doi: 10.1103/physreve.79.066107]

[39] Xie J, Szymanski BK, Liu X. Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In: Proc. of the 11th ICDMW. IEEE, 2011. 344−349. [doi: 10.1109/icdmw.2011.154]

[40] Davis M, Logemann G, Loveland D. A machine program for theorem-proving. Communications of the ACM, 1962,5(7):394−397. [doi: 10.1145/368273.368557]

[41] Silva JPM, Sakallah KA. GRASP—A new search algorithm for satisfiability. In: Proc. of the Best of ICCAD. Boston: Springer-Verlag, 2003. 73−89. [doi: 10.1007/978-1-4615-0292-0_7]

[42] Zuo XL, Li WJ, Liu YC. Discrete Mathematics. Shanghai: Shanghai Scientific and Technological Literature Press, 1982. 29−39 (in Chinese).

[43] Quine WV. The problem of simplifying truth functions. The American Mathematical Monthly, 1952,59(8):521−531. [doi: /10.2307/2308219]

[44] Khorasgani RR, Chen J, Zaiane OR. Top leaders community detection approach in information networks. In: Proc. of the 4th SNA-KDD Workshop on Social Network Mining and Analysis. 2010.

[45] Yang J, Leskovec J. Defining and evaluating network communities based on ground-truth. Knowledge and Information Systems, 2015,42(1):181−213. [doi: 10.1007/s10115-013-0693-z]

[46] Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. Physical Review E: Statistical Nonlinear & Soft Matter Physics, 2008,78(4):046110. [doi: 10.1103/physreve.78.046110]

[47] Wang X, Chen G, Lu H. A very fast algorithm for detecting community structures in complex networks. Physica A: Statistical Mechanics and its Applications, 2007,384(2):667−674. [doi: 10.1016/j.physa.2007.05.013]

附中文参考文献:

[7] 单菁,申德荣,寇月,等.基于重叠社区搜索的传播热点选择方法.软件学报,2017,28(2):326−340. http://www.jos.org.cn/1000-9825/5117.htm [doi: 10.13328/j.cnki.jos.005117]

[8] 钱珺,王朝坤,郭高扬.基于社区的动态网络节点介数中心度更新算法.软件学报,2018,29(3):853−868. http://www.jos.org.cn/1000-9825/5457.htm [doi: 10.13328/j.cnki.jos.005457]

[27] 乔少杰,韩楠,张凯峰,等.复杂网络大数据中重叠社区检测算法.软件学报,2017,28(3):631−647. http://www.jos.org.cn/1000-9825/5155.htm [doi: 10.13328/j.cnki.jos.005155]

[34] 尚敬文,王朝坤,辛欣,等.基于深度稀疏自动编码器的社区发现算法.软件学报,2017,28(3):648−662. http://www.jos.org.cn/1000-9825/5165.htm [doi: 10.13328/j.cnki.jos.005165]

[42] 左孝凌,李为鑑,刘永才.离散数学.上海:上海科学技术文献出版社,1982.29−39.

竺俊超(1994−),男,浙江上虞人,博士生,主要研究领域为社会网络,社区发现,社区搜索.

王朝坤(1976−),男,博士,副教授,博士生导师,CCF 专业会员,主要研究领域为图数据管理,社会网络,大数据系统.