组合查询条件下的属性社区搜索

王 玲[1] 刘晓清[2*] 何震瀛[2] 荆一楠[2]

[1](复旦大学软件学院 上海 200438)

[2](复旦大学计算机科学技术学院 上海 200438)

摘 要 传统的属性社区搜索问题仅研究查询属性在结果社区中存在与否。为应对复杂查询场景的需求,研究了组合查询条件下的属性社区搜索问题:给定多属性集合,各属性至少需满足的数量以及社区大小的上限约束,搜索所有节点的最小度数最大化的社区。提出通用算法解决框架,并以此为基础,提出两个优化方法,分别是:基于属性特征的搜索空间优化,以减小搜索空间;基于结构特征的搜索顺序优化,以通过搜索顺序的调整进一步提升算法效率。实验结果表明,算法可找到符合组合查询条件的属性社区。在大规模数据集上,经过两个优化后的算法效率比原算法提升 2 ~ 3 倍,同时内存开销减少约 50%。

关键词 社区搜索 组合查询条件 属性社区

中图分类号 TP311 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2024.04.006

ATTRIBUTED COMMUNITY SEARCH METHOD UNDER COMBINED QUERY CONDITION

Wang Ling[1] Liu Xiaoqing[2*] He Zhenying[2] Jing Yinan[2]

[1](*School of Software*, *Fudan University*, *Shanghai 200438*, *China*)

[2](*School of Computer Science*, *Fudan University*, *Shanghai 200438*, *China*)

Abstract The traditionally attributed community search problem only studies whether the query attribute exists in the resulting community. To address the needs of complex query scenarios, attributed community search problem under combined query condition is studied. We gave multiple attribute sets, the minimum number of each attribute and the upper limit of community size, and we searched for a community with the maximal number of the minimum degree of nodes. This paper proposed a general algorithm solution framework and two optimization methods: search space optimization based on attribute features to reduce the search space; search order optimization based on structural features to improve algorithm efficiency further by adjusting the search order. Experimental results show that the algorithm can find the attributed community that meets the combined query condition. After two optimizations, the optimized algorithm's efficiency is 2 ~ 3 times higher than the original algorithm on large datasets, and memory overhead is reduced by about 50%.

Keywords Community search Combined query condition Attributed community

0 引 言

社区是网络中一组内部连接紧密的节点[1]。社区在现实中常被赋予不同的含义,并在应用开发及科学研究中发挥重要作用。例如,在 Facebook 中,社区由关系紧密的用户组成;在万维网上,社区可以是所有共享相似主题的网站;在生物学研究中,研究者利用蛋白质相互作用网络中挖掘出的社区研究其对应的功能模块[2]。

社区搜索问题指给定一个或多个节点,寻找包含这些节点的社区。在真实的、大型的网络中,社区搜索旨在高效地找到用户关心的节点所在的社区。社区搜索具有重要研究意义,在团队组建[3]、个人情境发现[4]、

收稿日期:2021 - 02 - 06。国家自然科学基金项目(61732004,61872207)。**王玲**,硕士生,主研领域:社交网络,数据分析。**刘晓清**,教授级高工。**何震瀛**,副教授。**荆一楠**,副教授。

社交建模[5]等方面均有广泛的应用。

目前,社区搜索算法主要基于 k-core、quasi-clique、k-truss 等特定结构。Sozio 等[6]采用了组合优化的方法研究社交网络中的社区搜索问题,其目标是找到包含所有查询节点的最大连通子图。Cui 等[7]进一步针对基于 k-core 的社区搜索问题提出了局部搜索算法。Li 等[8]首先考虑了节点的权重,提出了基于 k-core 的 k-influential 社区模型。Cui 等[9]研究重叠社区搜索问题,提出了 quasi-clique 模型。单菁等[10]提出了一种基于重叠社区结构的方法来衡量节点影响力。Huang 等[11]提出了 k-truss 社区模型,它可以找到包含一个查询节点的所有重叠社区。与 Huang 等方法相比,Akbas 等[12]提出了另一种更高效的索引算法 EquiTruss。徐兰天等[13]研究了基于 k-truss 模型的时序社区搜索。此外,由于网络结构在现实中往往随时间而变化,也出现了一些基于动态图的研究[14-17]。

近年来,越来越多的研究开始关注属性图,即节点或边上带有属性信息的图。基于属性图的社区搜索问题具有一定的实际意义。例如,协作网络中的专家具有擅长的领域;社交网络中用户可能具有位置、兴趣、技能等信息。这样的网络通常建模为属性图。在这类属性图上搜索得到的社区结果被称为属性社区。属性社区搜索常见的研究都涉及属性的同质性:Fang 等[18]提出了基于 k-core 的属性社区,社区中的节点共享相同的属性;Huang 等[19]研究了查找包含查询节点和属性的连通 k-truss 子图以保证尽可能多的节点具有相似的属性。近期竺俊超等[20]研究了复杂条件下的属性社区搜索问题,基于布尔表达式提出条件社区搜索问题,并给出了解决该问题的通用算法框架。

上述工作的一个显著局限性在于:它们仅针对查询属性在结果社区中的存在与否,而忽略了属性的数量。面对实际应用中的复杂查询请求,这些方法在表述能力及算法方面都存有不足。为此,本文研究了组合查询条件下的属性社区搜索问题:搜索符合组合查询条件的结构紧密的属性社区,用户可通过组合查询条件指定结果社区中需包含的属性种类、属性数量、社区大小。

例 1　以团队组建问题为例,如图 1 所示,节点集合$\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{10}\}$中包含 10 位专家的节点,每个节点上的标签分别代表对应专家擅长的领域。这些专家擅长数据库(Database,DB)、数据挖掘(Data Mining,DM)、信息检索(Information Retrieval,IR)、机器学习(Machine Learning,ML)和自然语言处理(Natural Language Processing,NLP)领域。两个节点之间存在边代表两位专家曾成功合作。某研究项目需要一个熟悉不同领域的专家团队,需要至少两名数据库专家、至少两名机器学习专家和至少一名自然语言处理专家。为了控制研究预算,项目经理期望团队整体规模不超过 9 人。本文将这个应用请求表述为 $Q=\{\{(D_B,2),(M_L,2),(N_{LP},1)\},9\}$ 的形式,期望找到满足如下条件的社区:(1) 社区中有足够数量的专家擅长查询条件中所要求的技能;(2) 社区的结构紧密,保证尽可能多的专家之间曾相互合作过,从而降低沟通成本。基于此方式,项目的成功概率更高。

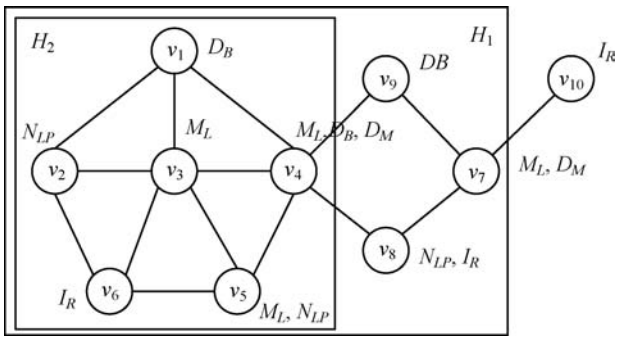


图 1　组合查询条件下的属性社区搜索示例

针对上述问题,本文研究了组合查询条件下的属性社区搜索方法。首先给出问题定义,然后提出算法整体框架,并在此基础上提出两个优化算法:基于属性特征的搜索空间优化算法和基于结构特征的搜索顺序优化算法。最后,针对上述三个算法,在四个真实数据集上做了案例研究、效率分析和内存开销分析,验证了算法能够找到有效的结果,在大规模数据集上,经过两个优化后的算法效率比原算法提升 2 ~ 3 倍,同时内存开销减少约 50%。

1　基础概念

用于社区搜索的一个经典模型是 k-core 模型。k-core 模型要求找到的社区中所有节点的最小度数大于等 k。k 越大,社区结构越紧密。本节首先给出文中用到的相关符号,并引入经典的 k-core[20]、core number[20]、k-shell[21]的定义,以便于后继的问题描述和算法描述。

给定一个无向属性图 $G=(V,E,L)$,其中:V 代表图 G 中的节点集合,E 代表图 G 中的边集合,L 代表图 G 中的属性集合。属性集合 L 与节点集合 V 之间存在映射关系:对于每个节点 $v\in V$,包含若干属性,定义映射$\mathcal{L}(v)$,$\mathcal{L}(v)=\{\alpha_1,\alpha_2,\cdots,\alpha_i\}$($\alpha_i$代表属性的名称)。例如,对于图 1 中的节点 v_4,$\mathcal{L}(v_4)=\{M_L,D_B,D_M\}$。本文用 A 表示查询中的属性集合,$A=\{A_1,A_2,\cdots,A_i\}$(A_i代表查询属性的名称),给定一个图 G 和一个查询属性 A_i,用 $V_{A_i}(G)=\{v\in V:A_i\in\mathcal{L}(v)\}$ 表示图 G 中包

含属性 A_i 的节点集合。本文用 $deg_G(v)$ 表示图 G 中节点 v 的度数。

定义 1 $deg_{G_{\min}}(v)$ 值。给定一个图 $G=(V,E,L)$，$deg_{G_{\min}}(v)$ 值为图 G 中所有节点的最小度数，即 $deg_{G_{\min}}(v)=\min_{v\in V} deg_G(v)$。

例如，给定图 1 所示 $G=(V,E,L)$，G 中度数最小的节点为 v_{10}，因此 $deg_{G_{\min}}(v)=deg_G(v_{10})=1$。

定义 2 k-core 结构。给图 $G=(V,E,L)$ 和一个整数 $k(k\geqslant 0)$，k-core 为图 G 的最大子图，且所有节点的度数均大于等于 k，记为 C_k，$deg_{C_k\min}(v)\geqslant k$。

例如，图 1 是一个 1-core，记为 C_1，子图 H_1 是一个 2-core，记为 C_2，子图 H_2 是一个 3-core，记为 C_3。它们之间存在嵌套关系，C_3 嵌套在 C_2 中，C_2 嵌套在 C_1 中。给定两个整数 i 和 j，若 $i<j$，那么 $C_j\subseteq C_i$。

定义 3 core number 值。给定图 $G=(V,E,L)$，一个节点 v 的 core number 值为包含节点 v 的 k-core 的最大 k 值，记为 $c(v)$。

定义 4 k-shell 集合。给定图 $G=(V,E,L)$，k-shell 为一个节点集合，其中所有节点的 core number 值均为 k，记为 S_k，$S_k=\{v\mid c(v)=k\}$。

例如，图 1 中 v_7、v_8、v_9 属于 1-core 和 2-core，但不属于 3-core，因此它们的 core number 值为 2，即 $c(v_7)=c(v_8)=c(v_9)=2$，它们属于 2-shell。

2 问题描述

2.1 问题定义

本节首先给出组合查询条件的定义，接着给出组合查询条件下的属性社区搜索问题定义。

在实际应用中，查询总是很复杂，用户经常需查询多种属性，且对属性的数量有要求。考虑到成本，用户希望能够限制资源大小。因此本文定义了如下的组合查询条件。

定义 5 组合查询条件。组合查询条件为形如 $Q=\{W_Q,s\}$ 的查询，它包含一个整数 s，限制查询社区结果中节点的个数不大于 s，同时包含一组属性对集合 $W_Q=\{(A_1,c_{\text{ount}_1}),(A_2,c_{\text{ount}_2}),\cdots,(A_i,c_{\text{ount}_i})\}$，属性对 (A_i,c_{ount_i}) 表示查询社区结果中至少包含 c_{ount_i} 个 A_i 属性。

在实际应用中，一个节点可能包含多个查询属性，代表一个用户可能需扮演多个角色，这在应用中并不理想。例如，在图 1 所示的团队组建例子中，专家 v_4 既需做 DB 领域的工作，又需做 ML 领域的工作，由于专家精力有限，如此安排并不合理。基于此观察，本文提出属性映射：针对具体的应用，每个节点仅选取至多一个属性。

定义 6 属性映射。给定一个图 $G=(V,E,L)$，存在一个属性映射，对于所有节点 $v\in V$，均选取节点上至多一个属性，记为 $G'=KA(G)$。

例如，图 1 所示的团队组建例子中，给定查询 $Q=\{(D_B,2),(M_L,2),(N_{LP},1),9\}$，节点 v_4 包含查询属性 ML 和 DB，v_5 包含查询属性 ML 和 NLP，它们均包含多个查询属性，本文针对每个节点仅选取至多一个属性。如图 1 中的子图 H_2 存在属性映射 $KA(H_2)$：$\mathcal{L}(v_1)=\{D_B\}$，$\mathcal{L}(v_2)=\{N_{LP}\}$，$\mathcal{L}(v_3)=\{M_L\}$，$\mathcal{L}(v_4)=\{D_B\}$，$\mathcal{L}(v_5)=\{M_L\}$。

接着，本文提出组合查询条件下的属性社区搜索问题。

定义 7 组合查询条件下的属性社区搜索问题。给定一个属性图 $G=(V,E,L)$ 和一个组合查询条件 $Q=\{\{(A_1,c_{\text{ount}_1}),(A_2,c_{\text{ount}_2}),\cdots,(A_i,c_{\text{ount}_i})\},s\}$，搜索属性社区，记为 C_{optimal}，满足以下条件：

（1）连通性。子图 $H\subset G$ 为连通图。

（2）多属性紧密性。对于 H，存在属性映射 $H'=KA(H)$，满足查询条件中属性个数要求和大小约束：$|V_{A_i}(H')|\geqslant c_{\text{ount}_i}$ 且 $|V(H)|\leqslant s$。

（3）结构紧密性。在所有满足条件（1）和条件（2）的子图 H 中，C_{optimal} 为 $deg_{H_{\min}}(v)$ 值最大的子图。

例如，给定图 1 所示的属性图和组合查询条件 $Q=\{(D_B,2),(M_L,2),(N_{LP},1),9\}$，对于图 1 中的子图 H_2 存在属性映射 $H'_2=KA(H_2)$，使得 $|V_{\text{DB}}(H'_2)|=2$，$|V_{\text{ML}}(H'_2)|=2$，$|V_{\text{NLP}}(H'_2)|=1$，满足组合查询条件中的属性个数要求，且 $|V(H_2)|=6<9$，满足大小约束。子图 H_1 和 H_2 均满足条件（1）和（2），但 $deg_{H_{1\min}}(v)=2$，$deg_{H_{2\min}}(v)=3$，$deg_{H_{2\min}}(v)>deg_{H_{1\min}}(v)$，因此 H_2 满足条件（1）、条件（2）、条件（3），是符合要求的查询结果。

2.2 问题复杂度

组合查询条件下的属性社区搜索问题是一个 NP-Hard 问题。在 Sozio[6] 等提出的问题四中，要求在给定大小约束下，使得找到社区中所有节点的最小度数最大化，该问题已被证明为 NP-Hard。本问题可构造如下：给定一个属性图 $G=(V,E,L)$ 和一组包含属性 A_1，$A_2,\cdots,A_i$ 的查询节点，且每种属性的个数均符合 $|V_{A_i}|\geqslant c_{\text{ount}_i}$，节点的个数小于等于 s，找到符合查询条件的属性社区，使得社区中所有节点的最小度数最大化。本文问题通过该构造可规约到上述问题四中，因此本问题的复杂度为 NP-Hard。

3　基础算法

3.1　组合查询条件下的属性社区搜索算法

本节针对 2.1 节中定义的问题,提出组合查询条件下的属性社区搜索算法(ACCQ),包含搜索和属性检查过程。为便于算法描述,后文算法将定义 1 中的 $deg_{G_{\min}}(v)$ 记为 d。

算法 1 展示了算法整体框架。算法具体步骤如下:初始化一个空集合,记录当前找到的社区 C(第 1 行),从图中某节点 v 开始搜索,依次访问当前节点的邻居节点,调用 Search 算法以检查当前社区是否满足查询条件(第 2 -3 行)。若当前节点的所有邻居节点均已被访问,此节点从图中移除(第 4 -5 行)。算法递归地枚举所有可能结果,最终返回一个 d 值最大的属性社区 C_{optimal}(第 6 行)。

算法 1　组合查询条件下的属性社区搜索算法

输入:$G=(V,E,L)$,$Q=(W_Q,s)$。

输出:C_{optimal}。

1. $C\leftarrow\varnothing$;
2. For all vertex $v\in V$ do
3. Search($G,Q,v,C\cup\{v\}$);
4. $G=G-\{v\}$;
5. End For
6. Return C_{optimal}

Search 算法每次返回当前找到的社区 C'_{optimal}。如算法 2 所示,Search 算法具体步骤如下:首先进行判断,若当前社区 C 的大小比查询限制的 s 值大,则直接返回当前结果(第1 -3 行)。每次加入节点的同时检查当前社区中的属性个数是否符合查询要求,进行若干判断(第 4 行):(1) 当前社区 C 的 d 值大于当前找到的社区 C'_{optimal} 的 d 值;(2) 当前社区 C 的大小大于等于所有属性的个数之和 $\sum_{1}^{t} c_{\text{ount}_i}$;(3) 社区中每种属性 A_i 的个数大于等于查询要求的数量 c_{ount_i}。(1) 保证了找到的社区结构紧密,(2)和(3)保证了进行属性检查有解。针对属性映射和属性数量检查问题,通过构造二分图,寻找最大匹配实现(第 5 -9 行)。具体做法为:利用组合查询条件中的属性和当前社区中的节点构造二分图,每次寻找多条不相交的增广路径,最终找到一个最大匹配即找到一组满足查询中的属性要求的节点。每次搜索节点时,从当前节点 v 的邻居节点 u 开始访问(第 10 行),若节点 u 的度数大于目前找到的社区 C'_{optimal} 的 d 值,节点 u 加入社区 C(第 11 -12 行)。

算法 2　Search 算法

输入:$G=(V,E,L)$,$Q=(W_Q,s)$,v。

输出:C'_{optimal}。

1. If $C.size>s$ then
2. Return C'_{optimal};
3. End If
4. If $C.d>C'_{\text{optimal}}.d$ and $C.size\geqslant\sum_{1}^{t} c_{\text{ount}_i}$ and $|V_{A_i}(C)|\geqslant c_{\text{ount}_i}$ then
5. bigraph $\leftarrow$ BiGraph(W_Q in Q, vertices in C)
6. If FindMatch(bigraph) then
7. $C'_{\text{optimal}}=C$;
8. End If
9. End If
10. For each neighbor u of v do
11. If $deg_G(u)>C'_{\text{optimal}}.d$ then
12. Search($G,Q,u,C\cup\{u\}$);
13. End If
14. End For
15. Search($G,Q,u,C\cup\{u\}$);
16. Return C'_{optimal}

3.2　算法复杂度

给定一个图 $G=(V,E,L)$ 和查询 $Q=\{W_Q,s\}$,属性检查的复杂度为 $O(\sqrt{n}\cdot m)$,其中:n 是图中节点的数量,m 是边的数量。搜索符合结构要求的社区的复杂度为 $O(C_n^s)$。因此组合查询条件下的属性社区搜索算法的复杂度为 $O(C_n^s\cdot\sqrt{n}\cdot m)$。

4　优化算法

4.1　基于属性特征的搜索空间优化

基于 3.2 节分析,算法复杂度较高,当图规模较大时,时间开销较大。另一方面,基于查询的特征,算法需找到的社区结果中必然包含查询条件中的属性。因此,本文提出基于属性特征的搜索空间优化算法(AFSO)。

如算法 3 所示,AFSO 主要步骤如下,给定图 $G=(V,E,L)$ 和查询 $Q=\{\{(A_1,c_{\text{ount}_1}),(A_2,c_{\text{ount}_2}),\cdots,(A_i,c_{\text{ount}_i})\},s\}$,初始化一个向量 $\boldsymbol{S}_{\text{tartID}}$ 以存放起始搜索节点(第 2 行),计算图 G 中包含各个属性 A_i 的节点的数量 $|V_{A_i}(G)|$(第 3 行),根据 $|V_{A_i}(G)|$ 对属性进行排序(第 4 行),数量最少的属性记为 $A_{\min}$(第 5 行),将所有包含属性 $A_{\min}$ 的节点存入 $\boldsymbol{S}_{\text{tartID}}$ 中(第 6 -10 行),从 $\boldsymbol{S}_{\text{tartID}}$ 中的节点开始做搜索(第 11 -15 行)。

算法 3　基于属性特征的搜索空间优化算法

输入:$G=(V,E,L)$,$Q=(W_Q,s)$。

输出:C_{optimal}。

1. $C\leftarrow\varnothing$;

2. Initialize a vector $\boldsymbol{S}_{\text{tartID}}$;
3. Compute $|V_{A_i}(G)|$;
4. Sort $|V_{A_i}(G)|$ according to A_i;
5. $A_{\min} \leftarrow A_i$ with min $|V_{A_i}(G)|$;
6. For $v \in V$ do
7. If $L_G(v)$ contains $A_{\min}$ in W_Q then
8. push v into $\boldsymbol{S}_{\text{tartID}}$;
9. End If
10. End For
11. For all vertex $v \in \boldsymbol{S}_{\text{tartID}}$ do
12. Search($G, Q, v, C \cup \{v\}$);
13. $G = G - \{v\}$;
14. End For
15. Return C_{optimal}

例 2 给定图 1 和查询 $Q = \{(M_L, 2), (I_R, 1), 6\}$，包含属性 *ML* 的节点有 v_3、v_4、v_5、v_7，含属性 IR 的节点有 v_6、v_8、v_{10}。根据算法 3，$A_{\min}$为属性 IR，因此 $\boldsymbol{S}_{\text{tartID}}$包含 v_6、v_8、v_{10}，搜索从这 3 个节点开始，而原搜索起始节点有 10 个，搜索空间减小。

4.2 基于结构特征的搜索顺序优化

本节根据 core 结构特征，提出基于结构特征的搜索顺序优化算法（SFOO）。由于本文要求找到的社区的 d 值最大，优化思路如下：优先从 core number 值大的节点开始做搜索，同时增加提前结束搜索的条件。

基于结构特征的搜索顺序优化算法如算法 4 所示，算法具体步骤如下：首先，利用 core decomposition[21] 计算所有节点的 core number 值（第 2 行）。如图 1 中所有节点的 core number 值和所属的 k-shell 在表 1 中列出。

算法 4 基于结构特征的搜索顺序优化算法

输入：$G = (V, E, L)$，$Q = (W_Q, s)$。

输出：C_{optimal}。

1. $C \leftarrow \varnothing$;
2. Core decomposition on G;
3. Initialize a vector $\boldsymbol{S}_{\text{tartID}}$;
4. Compute $|V_{A_i}(G)|$;
5. Sort $|V_{A_i}(G)|$ according to A_i;
6. $A_{\min} \leftarrow A_i$ with min $|V_{A_i}(G)|$;
7. For $v \in V$ do
8. If $L_G(v)$ contains $A_{\min}$ in W_Q then
9. push v into $\boldsymbol{S}_{\text{tartID}}$;
10. End If
11. End For
12. $c(v) \leftarrow$core number for each node $v \in \boldsymbol{S}_{\text{tartID}}$;
13. Sort $v \in \boldsymbol{S}_{\text{tartID}}$ according to $c(v)$ in descending order;
14. For all vertex $v \in \boldsymbol{S}_{\text{tartID}}$ do
15. Search($G, Q, v, C \cup \{v\}$);
16. If $c(v) < C_{\text{optimal}}.d$ then
17. Return C_{optimal};
18. End If
19. $G = G - \{v\}$;
20. End For
21. Return C_{optimal}

表 1 *k*-shell 及其对应的节点

S_k	节点数
S_1	$c(v_{10}) = 1$
S_2	$c(v_7) = 2, c(v_8) = 2, c(v_9) = 2$
S_3	$c(v_1) = 3, c(v_2) = 3, c(v_3) = 3, c(v_4) = 3,$ $c(v_5) = 3, c(v_6) = 3$

然后，根据这些节点的 core number 值对它们按降序排序，存入一个向量中（第 12 – 13 行），如图 2 所示。

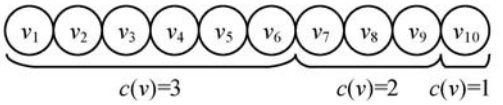


图 2 节点排序示例

算法增加提前结束条件：若当前访问节点的 core number 值比当前找到的社区结果的 d 值小，就无须搜索剩余节点，直接返回当前结果（第 16 – 17 行）。

例 3 接着 4.1 节中的例 2，给定图 1 和查询 $Q = \{\{(M_L, 2), (I_R, 1)\}, 6\}$，从 $\boldsymbol{S}_{\text{tartID}} = \{v_{10}, v_8, v_6\}$ 开始搜索，由于 $c(v_6) > c(v_8) > c(v_{10})$，因此搜索的顺序为 v_6、v_8、v_{10}。从 v_6出发可找到满足组合查询条件的属性社区 $C_{\text{optimal}} = \{v_2, v_3, v_5, v_6\}$，$C_{\text{optimal}}.d = 3$，访问 v_8时发现，$c(v_8) = 2 < 3$，算法可提前结束。

5 实 验

5.1 实验设置

（1）实验环境。本文所有程序使用 C ++ 实现。实验的硬件环境如下：Intel（R） Core（TM） i7-9700K CPU @ 3.60 GHz。软件环境如下：ubuntu 18.04。

（2）实验数据集。本文使用四种不同规模的真实数据集。数据集的具体信息如表 2 所示。Email-EuAll 数据集利用欧洲一家大型研究机构的电子邮件数据生成。Facebook 和 Deezer 数据集是社交网络数据集。Email-EuAll、Facebook 和 Deezer 数据集均来源于斯坦福网络分析平台。网络节点上属性的生成参考 Huang 等的做法，为每个网络生成属性集合，接着为每个节点随机分配 1 ~ 10 个属性[19]。DBLP 是一个专家协作网络。DBLP 数据集来源于 DBLP 网站，本文依据中国计算机学会推荐国际学术会议和期刊目录，爬取 DBLP 网站上 10 个领域中的所有文章及对应的作者，作者发

表的文章所属的领域作为该作者的属性标签,即节点上的属性。

表 2　数据集信息

网络	节点数	边数
Email-EuAll	1 005	25 571
Facebook	22 470	171 002
Deezer	28 281	92 752
DBLP	203 041	863 892

(3) 对比方法描述。由于此问题是全新的,在此前并未有相关的研究,本文以 ACCQ 作为基础算法,将 AFSO 和 SFOO 与之对比。

实验首先展示了 ACCQ 算法在 DBLP 数据集上的查询结果。接着变化查询参数:属性的种类数、属性的总数目和查询社区的大小,每次实验随机生成符合条件的 300 组查询,比较三种算法的运行时间以验证算法的效率。最后,分析了三种算法的内存开销。

5.2　实验结果及分析

5.2.1　案例研究

本文在 DBLP 数据集上做了案例研究。假如给定组合查询 $Q_1 = \{(D_B,2),(D_M,1),(I_R,1),5\}$,该查询需要找到包含至少两名擅长数据库的专家,至少一名擅长数据挖掘的专家,至少一名擅长信息检索的专家,且人数不超过五人的社区。

ACCQ 算法搜索的结果为图 3 中的子图 H,且节点所代表的人名和其擅长的领域已在节点上标注,其中:CG 表示计算机图形学领域;CHI 表示人机交互领域;AI 表示人工智能领域。它们组成的社区中包含的属性数量满足上述查询条件,A. Kumaran 和 Tanuja Josh 包含查询属性 DB,Udayan Khurana 包含查询属性 DM,Tobias Kellner 包含查询属性 IR;节点之间两两有边,社区的最小度数为 4,在符合查询条件的社区中度数最大,社区结构最紧密。

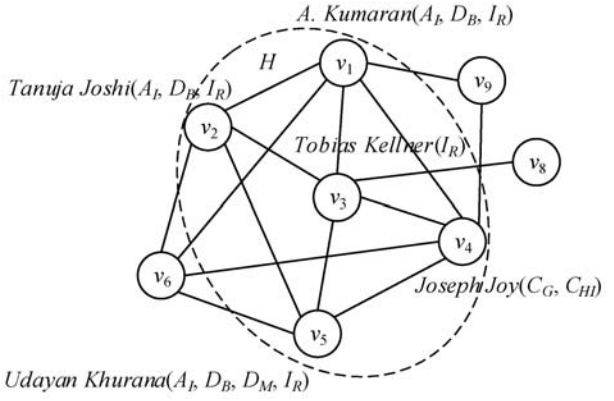
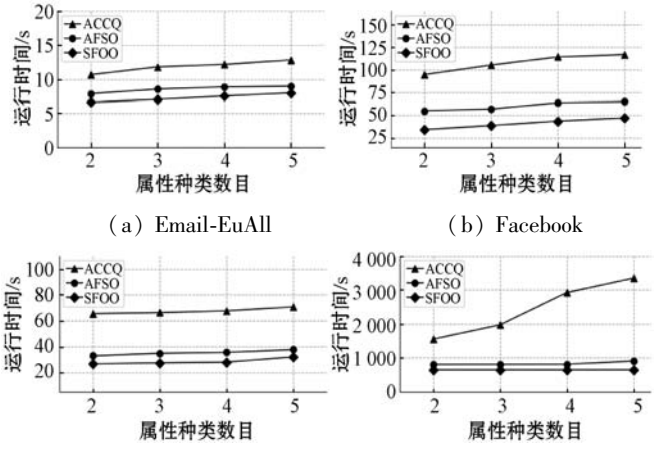


图 3　查询 Q_1 在 DBLP 数据集上的案例研究

5.2.2　效率分析

(1) 变化查询中属性的种类数 $|A|$。控制查询 Q 中的 $\sum_{i=1}^{|A|} c_{\text{ount}_i}$ 和 s 两个变量不变,变化属性种类数 $|A|$。具体做法为:设置查询中属性总数量为 5,社区大小为 5,即 $\sum_{i=1}^{|A|} c_{\text{ount}_i} = 5$, $s=5$,查询属性种类数 $|A|$ 变化范围为 2 ~ 5。三种算法在四个数据集上的运行时间如图 4 所示。



(a) Email-EuAll　(b) Facebook

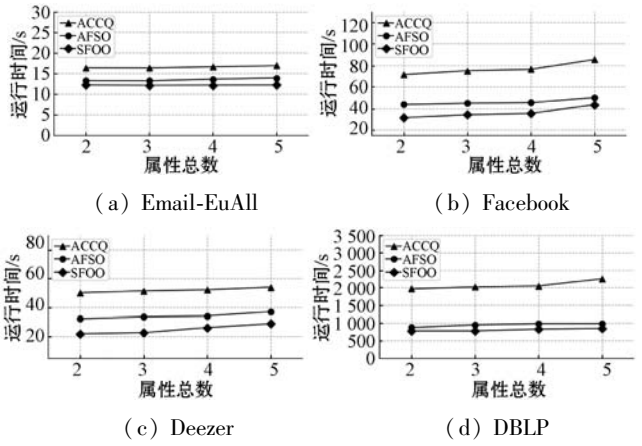
(c) Deezer　(d) DBLP

图 4　四个数据集上算法运行时间随属性种类数变化

整体上 ACCQ 运行时间随查询属性种类数增多而增加。这是由于随着查询属性种类增多,包含查询属性的节点增多,算法运行时间增加。同时,随属性种类数增多,算法优化效果越显著。这是由于 AFSO 根据属性特性将搜索起始节点数量缩小到仅包含最少属性个数的节点。

在小规模的数据集上算法能较快找到结果,如在 Email-EuAll 数据集上,算法能在几秒内找到结果。在大规模的数据集上 AFSO 和 SFOO 的优化效果更加明显。在 Facebook 数据集上,经过两次优化后的算法可将搜索时间降低一个数量级。在 DBLP 数据集上,AFSO 可使得各种搜索条件下的算法运行时间保持在 1 000 秒左右,快于 ACCQ 算法 2 ~ 3 倍左右。SFOO 相较于 AFSO 提升 30% 左右。

(2) 变化查询中属性的总数目 $\sum_{i=1}^{|A|} c_{\text{ount}_i}$。控制查询 Q 中的 $|A|$ 和 s 两个变量不变,变化属性的总数目 $\sum_{i=1}^{|A|} c_{\text{ount}_i}$。具体的做法为:设置查询中属性种类数为 2,社区大小为 5,即 $|A|=2$, $s=5$,查询属性的总数目即 $\sum_{i=1}^{|A|} c_{\text{ount}_i}$ 变化范围为 2 ~ 5。三种算法在四个数据集上的运行时间如图 5 所示。



（a）Email-EuAll （b）Facebook

（c）Deezer （d）DBLP

图 5 四个数据集上算法运行时间随属性个数变化

在不同数据集上，算法运行时间随属性变化有一定差别。这是由于不同数据集中节点的连接方式不同。在 Email-EuAll 数据集上，算法可在 10 ~ 20 秒内找到结果。AFSO 相较于 ACCQ 算法提升了 20% 左右。在 Facebook 和 Deezer 数据集上，AFSO 的优化效果更为明显，相较于 ACCQ 算法提升了 30% ~40% 左右。在更大规模的数据集 DBLP 上，AFSO 运行时间快于 ACCQ 算法 2 倍左右。SFOO 运行时间最快。

（3）变化查询中限制的社区大小 s。为了探究查询社区大小 s 对算法运行时间的影响，控制查询属性种类 $|A|$ 和总数目 $\sum_{i=1}^{|A|} c_{\text{ount}_i}$ 不变。具体做法为：设置 $|A|=2$，$\sum_{i=1}^{|A|} c_{\text{ount}_i}=5$，社区大小 s 变化范围为 2 ~ 6。三种算法在四个数据集上的运行时间如表 3 - 表 6 所示。

表 3 在 Email-EuAll 上算法运行时间随 s 变化

s	ACCQ 运行时间/s	AFSO 运行时间/s	SFOO 运行时间/s
2	0.045	0.036	0.033
3	0.307	0.245	0.243
4	2.703	2.322	2.240
5	16.404	13.197	12.640
6	63.643	50.605	48.498

表 4 在 Facebook 上算法运行时间随 s 变化

s	ACCQ 运行时间/s	AFSO 运行时间/s	SFOO 运行时间/s
2	14.704	9.705	8.324
3	16.703	10.013	8.461
4	22.863	15.576	12.286
5	70.670	43.655	31.345
6	260.489	161.543	106.933

表 5 在 Deezer 上算法运行时间随 s 变化

s	ACCQ 运行时间/s	AFSO 运行时间/s	SFOO 运行时间/s
2	7.456	5.252	4.288
3	10.673	7.362	6.932
4	20.659	16.923	14.024
5	53.044	36.162	21.926
6	214.169	173.596	164.022

表 6 在 DBLP 上算法运行时间随 s 变化

s	ACCQ 运行时间/s	AFSO 运行时间/s	SFOO 运行时间/s
2	292.525	113.509	79.256
3	293.344	116.033	83.492
4	351.158	142.446	102.000
5	1 966.819	878.064	780.881
6	53 368.084	23 100.995	14 370.860

随着查询社区大小 s 增大，算法运行时间增加，且 s 越大，时间增加得越快。在大规模的数据集上，算法优化更明显，且 s 越大，优化效果越明显。

在 Email-EuAll 数据集上，由于数据集规模较小，s 变化范围为 2 ~ 3 时，算法能在 1 秒内找到结果。随着查询社区大小 s 增大，算法能在几十秒内找到结果。在 Facebook 数据集和 Deezer 数据集上，s 变化范围为 2 ~ 3 时，两次优化后的算法能在几秒内找到结果。s 变化范围为 4 ~ 5 时，算法能在几十秒内找到结果。当 s 为 6 时，两次优化后的算法在 100 ~ 200 秒内找到结果，且算法优化效果明显。在 DBLP 数据集上，s 变化范围为 2 ~ 4 时，两次优化后的算法运行时间在 100 秒左右。当 s 增大至 6 时，算法运行时间在 10 000 秒左右，同时优化效果也更明显。AFSO 运行时间比 ACCQ 算法快 2 倍左右。SFOO 效果表现最优。

5.2.3 内存开销

本文测试了 ACCQ 算法、AFSO、SFOO 在 DBLP、Facebook、Deezer、Email-EuAll 数据集上的内存开销，如图 6 所示。

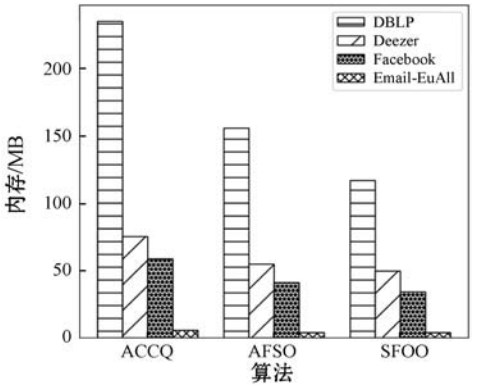


图 6 三种算法的内存开销

在相同的查询条件下,ACCQ 算法的内存开销最大。两个优化算法使内存开销减小。AFSO 使内存开销减小近 30%,这是由于算法从包含查询属性的节点搜索,减小了搜索空间。SFOO 也减少了内存开销,这是由于算法从 core number 值大的节点开始搜索能较快找到解,且能提前结束搜索。在大规模数据集上,经过两个优化后,内存开销减少约 50%。

6 结　语

本文研究了组合查询条件下的属性社区搜索问题,主要有以下贡献:(1) 扩充现有社区搜索的条件:包含属性种类、属性数量、社区大小等条件,能够应对更加复杂的组合查询。(2) 搜索社区的最小度数最大化,使得找到的社区结构紧密。接着给出通用的解决算法框架,在此基础上提出两个优化算法:基于属性特征的搜索空间优化和基于结构特征的搜索顺序优化。最后在四种真实数据集上进行实验,实验结果表明,算法能够找到有效的结果。在大规模数据集上,经过两个优化后的算法效率比原算法提升 2 ~ 3 倍,同时内存开销减少约 50%。

未来,可进一步研究最优化的属性社区搜索,如增加社区大小最小的约束条件,在相同 d 值下,搜索规模更小的社区结果,在实际应用中更能节约成本。

参 考 文 献

[1] Clauset A, Newman M E, Moore C. Finding community structure in very large networks[J]. Physical Review E, 2004,70(6):66111.

[2] Palla G, Derenyi I, Farkas I, et al. Uncovering the overlapping community structure of complex networks in nature and society[J]. Nature,2005,435(7043):814 - 818.

[3] Zhang J W, Yu P S, Lv Y H. Enterprise employee training via project team formation[C]//10th ACM International Conference on Web Search and Data Mining,2017:3 - 12.

[4] Chakraborty T, Patranabis S, Goyal P, et al. On the formation of circles in co-authorship networks[C]//21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,2015:109 - 118.

[5] Ugander J, Backstrom L, Marlow C, et al. Structural diversity in social contagion[J]. Proceedings of the National Academy of Sciences,2012,109(16):5962 - 5966.

[6] Sozio M, Gionis A. The community-search problem and how to plan a successful cocktail party[C]//16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,2010:939 - 948.

[7] Cui W Y, Xiao Y H, Wang H X, et al. Local search of communities in large graphs[C]//ACM SIGMOD International Conference on Management of Data,2014:991 - 1002.

[8] Li R H, Qin L, Yu J X, et al. Finding influential communities in massive networks[J]. The VLDB Journal,2017,26(6):751 - 776.

[9] Cui W Y, Xiao Y H, Wang H X, et al. Online search of overlapping communities[C]//ACM SIGMOD International Conference on Management of Data,2013:277 - 288.

[10] 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[J]. 软件学报,2017,28(2):326 - 340.

[11] Huang X, Cheng H, Qin L, et al. Querying K-truss community in large and dynamic graphs[C]//ACM SIGMOD International Conference on Management of Data,2014:1311 - 1322.

[12] Akbas E, Zhao P X. Truss-based community search: A truss-equivalence based indexing approach[J]. Proceedings of the VLDB Endowment,2017,10(11):1298 - 1309.

[13] 徐兰天,李荣华,王国仁,等. 面向时序图的 K-truss 社区搜索算法研究[J]. 计算机科学与探索,2019,14(9):1482 - 1489.

[14] Fang Y X, Wang Z, Cheng R, et al. On spatial-aware community search[J]. IEEE Transactions on Knowledge and Data Engineering,2019,31(4):783 - 798.

[15] Fang Y X, Cheng R, Li X D, et al. Effective community search over large spatial graphs[J]. Proceedings of the VLDB Endowment,2017,10(6):709 - 720.

[16] Li R H, Su J, Qin L, et al. Persistent community search in temporal networks[C]//34th International Conference on Data Engineering,2018:797 - 808.

[17] Li R H, Yu J X, Mao R. Efficient core maintenance in large dynamic graphs[J]. IEEE Transactions on Knowledge and Data Engineering,2013,26(10):2453 - 2465.

[18] Fang Y X, Cheng R, Luo S Q, et al. Effective community search for large attributed graphs[J]. Proceedings of the VLDB Endowment,2016,9(12):1233 - 1244.

[19] Huang X, Lakshmanan L V. Attribute-driven community search[J]. Proceedings of the VLDB Endowment,2017,10(9):949 - 960.

[20] 竺俊超,王朝坤. 复杂条件下的社区搜索方法[J]. 软件学报,2019,30(3):552 - 572.

[21] Barbieri N, Bonchi F, Galimberti E, et al. Efficient and effective community search[J]. Data Mining and Knowledge Discovery,2015,29(5):1406 - 1433.