软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2023,34(2):655−675 [doi: 10.13328/j.cnki.jos.006687]

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

异质信息网络中最大路径连通 Steiner 分量查询算法*

李 源[1], 范晓林[1], 孙 晶[1], 赵会群[1], 杨 森[1], 王国仁[2]

[1](北方工业大学 信息学院, 北京 100144)

[2](北京理工大学 计算机学院, 北京 100081)

通信作者: 孙晶, E-mail: sunjing@ncut.edu.cn

摘 要: 异质信息网络(HINs)是包含多种类型对象(顶点)和链接(边)的有向图, 能够表达丰富复杂的语义和结构信息. HINs 中的稠密子图查询问题, 即给定一个查询点 q, 在 HINs 中查询包含 q 的稠密子图, 已成为该领域的热点和重点研究问题, 并在活动策划、生物分析和商品推荐等领域具有广泛应用. 但现有方法主要存在以下两个问题: (1) 基于模体团和关系约束查询的稠密子图具有多种类型顶点, 导致其不能解决仅关注某种特定类型顶点的场景; (2) 基于元路径的方法虽然可查询到某种特定类型顶点的稠密子图, 但其忽略了子图中顶点之间基于元路径的连通度. 为此, 首先在 HINs 中提出了基于元路径的边不相交路径的连通度, 即路径连通度; 然后, 基于路径连通度提出了 k-路径连通分量(k-PCC)模型, 该模型要求子图的路径连通度至少为 k; 其次, 基于 k-PCC 模型提出了最大路径连通 Steiner 分量(SMPCC)概念, 其为包含 q 的具有最大路径连通度的 k-PCC; 最后, 提出一种高效的基于图分解的 k-PCC 发现算法, 并在此基础上提出了优化查询 SMPCC 算法. 大量基于真实和合成 HINs 数据的实验结果验证了所提出模型和算法的有效性和高效性.

关键词: 异质信息网络; 稠密子图查询; k-路径连通分量; 最大路径连通 Steiner 分量; 元路径

中图法分类号: TP311

中文引用格式: 李源, 范晓林, 孙晶, 赵会群, 杨森, 王国仁. 异质信息网络中最大路径连通 Steiner 分量查询算法. 软件学报, 2023, 34(2): 655–675. http://www.jos.org.cn/1000-9825/6687.htm

英文引用格式: Li Y, Fan XL, Sun J, Zhao HQ, Yang S, Wang GR. Querying Algorithm for Steiner Maximum Path-connected Components in Heterogeneous Information Networks. Ruan Jian Xue Bao/Journal of Software, 2023, 34(2): 655–675 (in Chinese). http://www.jos.org.cn/1000-9825/6687.htm

Querying Algorithm for Steiner Maximum Path-connected Components in Heterogeneous Information Networks

LI Yuan[1], FAN Xiao-Lin[1], SUN Jing[1], ZHAO Hui-Qun[1], YANG Sen[1], WANG Guo-Ren[2]

[1](School of Information Science and Technology, North China University of Technology, Beijing 100144, China)

[2](School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

Abstract: Heterogeneous information networks (HINs) are directed graphs including multi-typed objects (vertices) and links (edges), which can express rich semantic information and complex structural information. The problem of cohesive subgraph query in HINs, i.e., given a query vertex q, it could be found that the cohesive subgraphs containing q in HINs has become an important problem, and has been widely used in various areas such as event planning, biological analysis, and product recommendation. Yet existing methods mainly have the two deficiencies: (1) cohesive subgraphs based on relational constraint and motif cliques contain multiple types of vertices, which makes it hard to solve the scenario of focusing on a specific type of vertices; (2) although the method based on meta-path can query the cohesive subgraphs with a specific type of vertices, it ignores the meta-path-based connectivity between the vertices in the subgraphs.

* 基金项目: 科技创新 2030——“新一代人工智能”重大项目(2020AAA0108503); 国家自然科学基金(61902004, 61672041, 61772124, 61977001, 61732003); 北京市教委科技项目(KM202010009009)

收稿时间: 2021-08-01; 修改时间: 2021-11-08, 2022-01-17; 采用时间: 2022-04-01; jos 在线出版时间: 2022-07-22

Therefore, the connectivity with novel edge-disjoint paths is firstly proposed based on meta-path in HINs, i.e., path-connectivity. Then, the *k*-path connected component (*k*-PCC) is raised based on path-connectivity, which requires the path-connectivity of subgraph to be at least *k*. Next, the Steiner maximum path-connected component (SMPCC) is further proposed, which is the *k*-PCC containing *q* with the maximum path-connectivity. Finally, an efficient graph decomposition-based *k*-PCC discovery algorithm is designed, and based on this, an optimized SMPCC query algorithm is proposed. A large number of experiments on five real and synthetic HINs prove the effectiveness and efficiency of the proposed approaches.

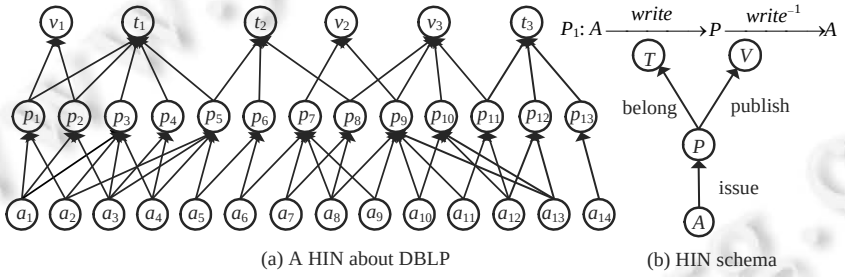
Key words: heterogeneous information networks (HINs); cohesive subgraph query; *k*-path connected component; Steiner maximum-path-connected component; meta-path

异质信息网络(heterogeneous information networks, HINs)是包含多种类型对象(顶点)和链接(边)的有向图, 它不仅完成了对现实世界万物更完整、自然的抽象建模, 而且能够表达丰富、复杂的语义和结构信息[1,2]. HINs 普遍存在于各个领域, 例如作者协作网络、基因疾病网络以及电子商务网络等. 图 1(a)表示 DBLP 网络的一个 HIN 实例, 其详细描述了 4 种不同类型对象(作者(*a*)、论文(*p*)、主题(*t*)、会场(*v*))及其之间的 3 种语义关系, 比如作者 a_1 和 a_2 共同发表论文 p_1, p_1 属于 t_1 主题并出版在 v_1 会场. 本文研究的是 HINs 中的稠密子图查询问题, 即给定一个查询点 *q*, 在 HINs 中查询包含 *q* 的稠密子图. 而该问题已成为图数据挖掘领域的热点和重点研究问题, 并在许多领域有广泛应用.

(1) 活动策划[3,4]. 例如: 为组织一场专题研讨会, 学者可在 DBLP 网络中进行稠密子图查询, 并依据返回结果进行策划;

(2) 生物分析[5−7]. 在基因的 HINs 中, 通过发现包含某基因的稠密子图, 可揭示这些基因之间的隐含关系, 从而进行分析;

(3) 商品推荐[8,9]. 比如: 在电子商务网络中为向消费者推荐商品, 可查询到该网络中的稠密子图, 从而进行精准推荐.



(a) A HIN about DBLP (b) HIN schema

图 1 DBLP 的 HIN 和 HIN 模式的实例及元路径 *P* 的实例

现有 HINs 中, 稠密子图查询的方法可根据子图中顶点类型个数主要分为两类.

- 第 1 类是查询到的稠密子图中包含多种类型顶点[10,11]. 例如: Hu 等人[10]通过建模模体团, 从而发现 HINs 中的稠密子图. 但是由于其定义过于严格, Jian 等人[11]通过限制 HINs 中顶点之间的连接个数, 提出了关系约束的稠密子图查询方法. 以图 1(a)中的 HIN 为例, 给定一个查询点 a_1, 约束关系为作者至少发表 3 篇论文, 论文至少有 2 个合著作者, 则该方法查询到的稠密子图的顶点集为{a_1,a_2,a_3, a_4,p_1,p_2,p_3,p_4,p_5}. 可以看到, 该类方法查询到的稠密子图具有两种类型的顶点, 但这也导致了该类方法不能解决只关注某种特定类型顶点的场景的问题. 若作者 a_1 想要组织一场研讨会, 邀请相关学者参加, 则该类方法就无法适用;
- 第 2 类方法是 Fang 等人[12,13]提出的一种基于元路径的方法, 该方法要求稠密子图中的顶点与 *q* 同类型, 并被给定元路径 *P* 的路径实例所连接. 元路径已被广泛研究, 它被定义为两个顶点类型之间顶点类型结合边类型的一种序列[14−16]. 例如, 图 1(b)中定义: 在作者(*A*)和论文(*P*)上的序列 P_1 为一个具体元路径, 它可解释为两个作者为合著关系. 虽然该方法可查询到具有某种特定类型顶点的稠密子图,

但它仅仅要求子图中的顶点被给定元路径的路径实例所连接, 忽略了顶点之间基于元路径的连通度, 因此使得查询到的子图不够稠密. 以图 1(a)中的 HIN 为例, 给定一个查询点 a_1 和元路径 P_1=(*APA*), 则最终得到的稠密子图的顶点集为{a_1,a_2,a_3,a_4,a_5}. 但从图 1(a)中可以看到: 作者 a_5 仅仅发表了 2 篇论文, 而剩下的作者至少发表了 3 篇论文, 因此, {a_1,a_2,a_3,a_4}对应的子图比{a_1,a_2,a_3,a_4,a_5}对应的子图更稠密.

针对以上问题, 本文首次在 HINs 中提出了基于元路径的边不相交路径的连通度, 即路径连通度这一概念; 进而, 基于路径连通度, 首次提出了 *k*-PCC 模型, 该模型要求子图的路径连通度至少为 *k*. 具体地,

(1) 边不相交路径. 给定元路径 *P*, 如果 *P* 的所有路径实例对应的边是不同的, 那么这些路径实例被称为边不相交路径. 例如: 给定元路径 P_1, 顶点 a_1 和 a_2 之间关于 P_1 的边不相交路径为($a_1p_1a_2$)和($a_1p_3a_2$).

(2) 路径连通度. 受启发于同质网络中连通度的概念[17], 本文提出了 HINs 中两不同顶点 *u* 和 *v* 之间的路径连通度定义, 其为使 *u* 和 *v* 不 *P*-邻接的删除最小边不相交路径的个数, 其中, *P*-邻接是指两顶点之间由 *P* 的路径实例所连接. 例如: 给定元路径 P_1, 使图 1(a)中顶点 a_1 和 a_2 不 P_1-邻接的删除最少的边不相交路径为($a_1p_1a_2$)和($a_1p_3a_2$), 故 a_1 和 a_2 之间的路径连通度为 2.

(3) 无向加权图. 基于以上定义, 可将 HIN 构建为一个无向加权图 G_P=(V_T,E_T,*w*), 其中, 顶点集 V_T 为与 *q* 同类型的所有顶点, 边集 E_T 和对应的权重 *w* 是根据路径连通度所计算的. 以图 1(a)中的 HIN 为例, 令 *q*=a_1, *P*=P_1, 图 2 展示了构建的对应无向加权图 G_{P_1}, 则给定一个无向加权图 G_P 和正整数 *k*, *k*-PCC 定义为图 G_P 中的诱导子图 *g*, 当且仅当满足以下条件: (i) *g* 的路径连通度不小于 *k*; (ii) 不存在 *g* 的母图满足条件(i). 其中, 使用 *k*-PCC 定义 HINs 中的稠密子图主要有以下两个优点: (i) *k*-PCC 不仅保证了顶点之间是否有元路径连接, 还考虑了顶点之间基于元路径的连通度, 因此更加稠密; (ii) 在设计无向加权图中的 *k*-PCC 发现算法时, 可利用高效的图分解方法, 便于扩展到大图计算中. 最后, 基于 *k*-PCC 模型提出了最大路径连通 Steiner 分量(Steiner maximum-path-connected component, SMPCC)概念, 其为包含 *q* 的具有最大路径连通度的 *k*-PCC. 同样地, 以图 1(a)中的 HIN 为例: 给定一个查询点 a_1 和一条元路径 P_1=(*APA*), 则查询到的 SMPCC 为{a_1,a_2,a_3,a_4}, 其比上述两类方法得到的结果更加稠密且准确. 因此, HINs 中的稠密子图查询问题可通过查询 SMPCC 来解决.

为了查询 HINs 中的 SMPCC, 本文需要解决以下 3 个问题.

(1) 如何高效地构建无向加权图. 首先, 构建无向加权图需要计算所有与 *q* 同类型的顶点作为其顶点集, 然而为了提高效率, 这里仅计算(*P*,θ)-连通于 *q* 的顶点作为其顶点集, 其中, (*P*,θ)-连通表示某一顶点经θ跳 *P* 的路径实例到达另一顶点; 然后, 再在其顶点集中通过深度优先搜索算法计算两点之间的路径连通度从而确定边集, 最终完成无向加权图的构建.

(2) 如何有效地在无向加权图中计算不同 *k* 值的所有 *k*-PCCs. 借鉴于无向无权图中 *k*-边连通分量的计算方法[18], 本文提出了基于割[19]理论的图分解方法. 具体地, 该方法可将一个不是 *k*-PCC 的无向加权图通过图分解的方式分解为如图 3 所示的图分解树. 其中, 每个 *k*-PCC 由叶子节点表示, 而树中的每个中间节点都是图分解过程中产生的非 *k*-PCC 子图. 另外, 树中的兄弟节点具有一个特性: 除其中一个兄弟节点外, 它们中的每一个都与父图剩余部分所连接边的权重和小于 *k*. 以图 2 所示的无向加权连通图 G_{P_1} 为例: 给定 *k*=3, 可将图 G_{P_1} 分解为 3 个不连通的 3-PCC 子图 g_1、g_2 和 g_3, 其中, g_1 与 G_{P_1} 剩余子图所连接的边权重为 1<3; 接着, 基于图分解的方法设计了给定 *k* 值的优化 *k*-PCCs 发现算法, 其时间复杂度为 O(*h*·*r*·|E_T|), 其中, *h* 和 *r* 均为较小的整数.

(3) 如何在 *k* 值未知的情况下, 从无向加权图中发现 SMPCC. 因为 SMPCC 为包含 *q* 的具有最大路径连通度的 *k*-PCC, 显然, 无向加权图中 *k*-PCC 的路径连通度不可能超过图中所有边的权重和, 所以可以从图中所有边的权重和开始依次递减计算对应 *k* 值的 *k*-PCCs, 同时检查发现的 *k*-PCCs 是否包含 *q*: 若包含, 则立即停止查找并返回该 *k*-PCC. 但是该方法存在一个问题: 图中所有边的权重之和会

远远大于 SMPCC 的路径连通度，导致其耗费大量时间. 为了解决该问题，本文提出了一个 SMPCC 路径连通度的上界定理，依据该定理，设计出了优化查询 SMPCC 算法，同时提出了大规模 HINs 中 SMPCC 查询算法.

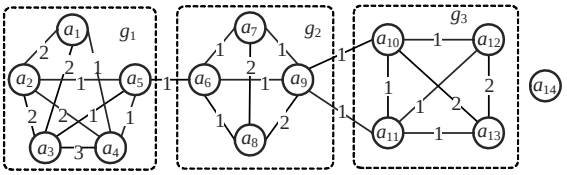


图 2 用 P_1 构建的无向加权图 G_{P_1}

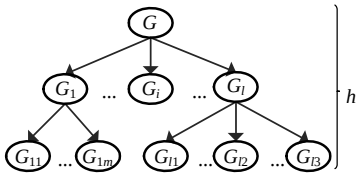


图 3 图分解树

本文的主要贡献包括:

(1) 提出了 HINs 中的 k-PCC 模型，该模型不仅可保证子图的高稠密性，还能确保子图可被高效地发现；进而基于 k-PCC 模型定义了 SMPCC 概念，其为包含 q 的具有最大路径连通度的 k-PCC;

(2) 为了发现 HINs 中的 k-PCCs，设计了基于元路径的无向加权图构建算法 GP-BUILD，并在无向加权图上提出了基于图分解技术的带两个优化策略的 k-PCCs 发现算法 KPCC-FIND;

(3) 为了高效地查询 SMPCC，提出了一个上界定理，并依据该定理设计了优化查询 SMPCC 算法 SMPCC-OPT 和适用于大规模网络的 SMPCC-BEST 算法，从而完成了 SMPCC 的高效查询;

(4) 在不同模式的真实和合成 HINs 数据集上进行了大量实验，以此验证了本文所提出模型和算法的有效性和高效性.

本文第 1 节对 HINs 和元路径的相关概念进行阐述，并提出本文研究的问题定义. 第 2 节设计基于元路径的无向加权图构建算法. 第 3 节首先给出图分解的相关理论知识，然后提出基于图分解的 k-PCCs 发现算法. 在第 4 节提出一个上界定理，并设计优化查询 SMPCC 算法. 第 5 节通过在真实和合成 HINs 上的实验，评估算法的有效性和高效性. 第 6 节总结现有稠密子图查询的研究工作. 第 7 节总结全文，并探讨未来研究工作.

1 基本概念及问题定义

本节将给出 HINs 和元路径的一些基本概念及其符号表示，介绍 k-PCC 模型及其相关定义，并阐述本文研究问题的具体定义. 表 1 简要总结了本文常用的符号及其含义.

表 1 符号及其描述

符号	描述	符号	描述
$G=(V,E,w)$	带顶点集 V、边集 E 及对应边权重 w 的无向加权图	Γ_{uv}	一组顶点 u 和 v 间的边不相交路径
$w(u,v)$	边(u,v)的权重	$\lambda(u,v)$	顶点 u 和 v 间的路径连通度
$G[X]$	顶点子集 X 在图 G 中的诱导子图	$f(S,T)$	割(S,T)的值
$G_H=(V,E,\Phi,\Psi)$	带顶点集 V 及其映射 Φ和边集 E 及其映射 Ψ的 HIN	G_P	用 P 构建的无向加权图
$M_H=(\mathcal{A},\mathcal{R})$	带顶点类型集$\mathcal{A}$和边类型集$\mathcal{R}$的 HIN 模式	h	图分解树的高度
q	一个查询点	r	Mas-GP-OPT 的迭代次数
P	M_H 上的一个对称元路径	k	一个正整数 $k(k\geqslant 2)$
p	P 的路径实例	k-PCC	k-路径连通分量
l	P 的长度，即 P 中边的个数	SMPCC	包含 q 有最大路径连通度的 k-PCC
θ	跳数，即顶点到达另一顶点通过的路径个数	$\sum w$	SMPCC 路径连通度的上界

1.1 基本概念

给定一个无向加权图 $G=(V,E,w)$，其中，V 表示顶点集，$|V|$表示顶点个数；E 表示边集，$|E|$表示边个数，边集 E 中的每条边用(u,v)表示，即$(u,v)\in E$；$w(u,v)$表示边集 E 中对应边(u,v)的权重. 假定一个顶点子集 $X\subseteq V$，则 X 的诱导子图 $G[X]=(X,E(X),w(E(X)))$是图 G 的子图，其中，X 为子图 $G[X]$的顶点集；$E(X)$为子图 $G[X]$的边集，并且 $E(X)\subseteq E$；$w(E(X))$是 $E(X)$中每条边的权重，即 $G[X]=(X,\{(u,v)\in E|u,v\in X\},\{w(u,v)|\{(u,v)\in E|u,v\in X\}\})$.

定义 1(HIN). HIN 为具有顶点类型映射 $\Phi:V\to\mathcal{A}$和边类型映射 $\Psi:E\to\mathcal{R}$的有向无权重图 $G_H=(V,E,\Phi,\Psi)$, 这里的$\mathcal{A}$和$\mathcal{R}$需要满足$|\mathcal{A}|>1$ 或$|\mathcal{R}|>1$. 其中, 每个顶点 $v\in V$ 属于顶点类型集$\mathcal{A}:\Phi(v)\in\mathcal{A}$中的一个具体顶点类型, 每条边 $e\in E$ 属于某一具体边类型 $\Psi(e)\in\mathcal{R}$.

定义 2(HIN 模式). 给定带有顶点类型映射 $\Phi:V\to\mathcal{A}$和边类型映射 $\Psi:E\to\mathcal{R}$的 HIN $G_H=(V,E,\Phi,\Psi)$, 则其 HIN 模式为以 G_H 中的顶点类型集$\mathcal{A}$为顶点集, 边类型集$\mathcal{R}$为边集的一个有向无权重图 $M_H=(\mathcal{A},\mathcal{R})$.

图 1(a)描述了以 DBLP 部分数据构建的一个 HIN 实例, 图 1(b)则是图 1(a)的 HIN 模式. 该 HIN 模式具有 4 种类型的顶点: 作者(A)、论文(P)、会场(V)和主题(T), 以及 3 种类型的关系. 另外, 对于不同顶点类型之间的关系, 如果存在从顶点类型 A 到 P 的关系 R, 那么必然存在从顶点类型 P 到 A 的逆关系 R^{-1}. 比如, 作者 a_1 和论文 p_1 之间的关系可表示为发表与被发表两种关系.

定义 3(元路径). 给定 $M_H=(\mathcal{A},\mathcal{R})$, 则元路径 P 定义为 M_H 上的路径, 表示为 $A_1\xrightarrow{R_1}A_2\xrightarrow{R_2}\cdots\xrightarrow{R_l}A_{l+1}$, 其中, $A_i\in\mathcal{A}$, $R_i\in\mathcal{R}$ ($1\leqslant i\leqslant l$). l 为元路径 P 的长度, 即元路径 P 中的边个数.

为了简便, 如果相同顶点类型之间没有多种边类型, 则元路径也可以用顶点类型来表示, 即 $P=(A_1A_2\cdots A_{l+1})$. 另外, 顶点 a_1 和 a_{l+1} 之间的具体路径 $p=(a_1a_2\cdots a_{l+1})$为元路径 P 的路径实例, 其中, 路径实例 p 满足以下条件: $\forall i$, 都有 $\Phi(a_i)=A_i$, 并且每条边 $e_i=(a_i,a_{i+1})$满足 $\Psi(e_i)=R_i$, 最终将其记为 $p\in P$. 以图 1(a)中的 HIN 为例, 路径$(a_1p_1a_2)$是 P_1 的一个路径实例. 注意: 这里的小写字母 a_i 表示 HIN 中的顶点, 大写字母 A_i 表示 HIN 模式中的顶点类型; 边信息同上.

如果元路径 P'是 P 在 HIN 模式 M_H 上的逆路径, 则称 P'为 P 的逆元路径, 并记为 P^{-1}. 如果 P 和 P^{-1} 相同, 则称它们为对称元路径. 相似地, p^{-1} 为 p 的逆路径实例. 图 1 中的 $P_1=(APA)$及其逆元路径 P_1^{-1} 仍为 P_1, 故 P_1 为对称元路径.

在 HIN 中, 如果顶点 u 和 v 被元路径 P 的路径实例 p 所连接, 则称它们为 P-邻接; 而如果存在从 u 到 v 的顶点链, 使得链中的任意顶点都与其邻居顶点是 P-邻接, 那么它们被称为 P-连通, 其中, 若经过 θ跳路径实例, 则称为(P,θ)-连通. 例如: 在图 1(a)中, 顶点 a_1 和 a_2 被 P_1 的路径实例$(a_1p_1a_2)$所连接, 故 a_1 和 a_2 为 P_1-邻接; 而顶点 a_1 和 a_3 之间存在顶点链$\{a_1a_2a_3\}$, 其中, 该链中的任意顶点都与其邻居顶点是 P_1-邻接的, 故 a_1 和 a_3 为 P_1-连通; 且 a_1 经过 2 个路径实例到达 a_1, 故 a_1 和 a_3 也称为$(P_1,2)$-连通.

1.2 问题定义

本文的目的是从 HINs 中查询包含 q 的稠密子图, 且子图中所有顶点类型都与 q 的类型($\Phi(q)$或称为目标类型)相同. (1) 为了连接具有目标类型的顶点, 本文使用了开始顶点类型和结尾顶点类型均与目标类型相同的对称元路径, 需要注意的是, 本文使用的元路径都是对称的; (2) 对于子图的稠密性, 本文提出了基于元路径的边不相交路径的连通度, 即路径连通度, 并在此基础上提出了 k-PCC 模型, 其中, 路径连通度约束边不相交是因为它会加强子图的稠密性, 减少弱参与稠密子图中的顶点. 例如: 通过 Fang 等人[12,13]基于元路径的方法发现的稠密子图为$\{a_1,a_2,a_3,a_4,a_5\}$, 但是从图 1(a)中可以看出, 虽然 a_5 在该子图中有 3 个合著者, 但是 a_5 仅仅发表了一篇 p_5 论文, 而剩余的 4 位作者均发表了至少 3 篇论文, 所以他们之间的关联性更高. 导致该现象的原因是边 $a_5\to p_5$ 共享在了 3 条路径实例中, 所以约束其边不相交则避免了这种情况. 以下的元路径 P 均是连接两个目标类型的元路径, 集合 V_T 是基于 P 找到的具有目标类型的顶点集, 同时也是图 G_P 中的顶点集, $G_P=(V_T,E_T,w)$是基于元路径 P 构建的无向加权图.

定义 4(边不相交路径). 给定元路径 P 的路径实例集合Γ, 对于任意两个路径实例 $p_1,p_2\in\Gamma$, 如果它们当中任意第 i($1\leqslant i\leqslant l$)条边都是不同的, 则称Γ为一组边不相交路径集. 特别地, Γ_{uv} 表示顶点 u 和 v 之间的边不相交路径集.

定义 5(路径连通度). 集合 V_T 中任意两不同顶点 u 和 v 的路径连通度$\lambda(u,v)$是使 u 和 v 不 P-邻接的删除最少边不相交路径的个数. 同时, 图 G_P 的路径连通度为图 G_P 中任意两不同顶点之间的最小路径连通度, 即:

$$\lambda(G_P)=\min_{u,v\in VT}\lambda(u,v).$$

定义 6(*k*-PCC). 给定无向加权图 G_P、顶点集 $X(X\subseteq V_T)$和正整数 $k(k\geqslant 2)$, 那么 X 的诱导子图 $G_P[X]$为 k-PCC, 如果满足以下条件: (1) $\lambda(G_P[X])\geqslant k$; (2) $G_P[X]$的任意母图的路径连通度都小于 k.

给定查询点 $q\in V_T$, 那么将 G_P 中包含 q 的诱导子图记为 G_P^q, 其中的所有顶点类型都为目标类型, 且被元路径 P 的边不相交路径所连接. 接下来, 基于 k-PCC 模型给出 SMPCC 的定义.

定义 7(SMPCC). 给定 G_P 中的诱导子图 G_P^q, 若 G_P^q 的路径连通度 $\lambda(G_P^q)$ 是最大的, 则称 G_P^q 为最大路径连通 Steiner 分量, 其为包含 q 的具有最大路径连通度的 k-PCC. 令 $spc(q)$为 q 的任意 SMPCC 的路径连通度, 则也称其为 q 的 Steiner 路径连通度.

问题 1. 给定 HIN G_H、查询点 q、元路径 P、跳数θ和正整数 k, 从 G_H 中发现所有的 k-PCCs.

例如: 在图 1(a)的 HIN 中, 给定 $q=a_1$, $P=P_1$, $\theta=5$, $k=3$, 则发现的所有 3-PCCs 为图 2 中的 3 个子图 g_1、g_2 和 g_3.

问题 2. 给定 HIN G_H、查询点 q、元路径 P 和跳数θ, 从 G_H 中查询最大路径连通 Steiner 分量, 即 q 的 SMPCC.

以图 1(a)中的 HIN 为例: 给定 $q=a_1$, $P=P_1$, $\theta=5$. 首先构建如图 2 所示的无向加权图 G_{P_1}, 然后从 G_{P_1} 中计算出 a_1 的 SMPCC, 即$\{a_1,a_2,a_3,a_4\}$. 明显地, a_1 的 SMPCC 为包含 a_1 的具有最大路径连通度的 5-PCC.

2 基于元路径的无向加权图构建算法

本节主要研究依据 HINs 构建基于元路径的无向加权图算法. 具体地, 给定 HIN G_H、查询点 q 和元路径 P, 依据 G_H 构建对应的无向加权图 $G_P=(V_T,E_T,w)$. 由于本文最终是在 HINs 中查询与 q 同顶点类型的稠密子图, 所以提出了依据 HINs 并基于对称元路径的无向加权图构建算法 GP-BUILD. 该算法的基本思想是: 先计算图 G_P 的顶点集, 然后根据路径连通度的概念计算其边集以及对应边的权重信息. 接下来, 首先详细分析 GP-BUILD 算法的基本思想, 然后给出具体的算法描述, 最后分析该算法的时间复杂度.

GP-BUILD 算法的基本思想: 首先, 可计算所有具有目标类型的顶点作为图 G_P 的顶点集 V_T, 然而这是很耗时且不必要的. 因为不是所有具有目标类型的顶点都 P-连通于 q, 同时, 在所有 P-连通于 q 的目标类型顶点中, 若有经过许多跳路径实例才到达的顶点, 则在查询 SMPCC 时基本不会影响它的有效性, 但是会极大地增加时间负担. 所以为了解决这些问题, 本文提出仅计算(P,θ)-连通于 q 的具有目标类型的顶点作为最终的顶点集 V_T. 然后, 根据新颖的路径连通度的概念, 在 HINs 上通过深度优先搜索算法计算顶点集 V_T 中任意两点之间的路径连通度, 从而确定最终的边集 E_T 以及对应边的权重 w.

算法 1 给出了 GP-BUILD 算法的具体描述.

- 首先, 该算法初始化最终的顶点集 V_T 为空, 并将查询点 q 添加至集合 Q, 且初始化 *skip* 记录当前的跳数, 以方便计算(P,θ)-连通于 q 的顶点(第 1 行).
- 然后, 从 Q 开始计算(P,θ)-连通于 q 的具有目标类型的顶点. 具体地, 从 q 开始先计算 P-邻接于 q 的具有目标类型的顶点集 N, 然后迭代地计算 P-邻接于 N 中所有顶点的具有目标类型的顶点集, 直到最终找到(P,θ)-连通于 q 的所有具有目标类型的顶点集 V_T(第 2–17 行).
- 其次, 计算顶点集 V_T 中任意两不同顶点之间的路径连通度, 从而确定边集 E_T 及权重 w. 详细来说, 先通过深度优先搜索算法计算两不同顶点之间是否存在元路径 P 的边不相交路径, 其中, 边不相交是从图 G_H 中删除当前已找到的路径实例中的每一条边后再去找下一条路径实例来保证的, 直到该两顶点之间不再存在 P 的边不相交路径为止(第 18–25 行). 若计算的该两点之间的路径个数大于 0, 则增加该两点之间的边并分配其权重为路径个数(第 26 行、第 27 行).
- 最终, 返回构建的无向加权图 G_P(第 30 行).

算法 1. 构建无向加权图(GP-BUILD).

输入: HIN G_H、查询点 q、元路径 P、跳数θ;

输出: 无向加权图 G_P.

1. $V_T=\varnothing$, $Q=\{q\}$, $skip=0$; //V_T: (P,θ)-连通于 q 的具有目标类型的顶点集
2. **WHILE** $|Q|>0$
3. $skip$++;
4. **FOR** $i\in[1,\ldots,l]$ //l 为元路径 P 的长度
5. $N=\varnothing$; //N: 与 Q 中顶点 P-邻接的顶点
6. **FOR** $u\in Q$
7. **FOR** $v\in$neighbors of u
8. **IF** type of (u,v) is i-th edge of meta-path P
9. **IF** (u,v) not signed i
10. Add v to N, and sign i to (u,v)
11. **END FOR**
12. $Q=N$;
13. **END FOR**
14. $Q=Q\backslash V_T$, $V_T=V_T\cup Q$;
15. **END FOR**
16. **IF** $skip=\theta$ **BREAK** //仅计算(P,θ)-连通于 q 的具有目标类型的顶点集
17. **END WHILE**
18. **FOR** $u\in V_T$ //开始计算 V_T 中任意两点间的路径连通度
19. **FOR** $v\in V_T\backslash u$
20. $p_{uv}=0$; //p_{uv}: 顶点 u 和 v 间路径个数
21. **DO** //循环寻找 u 和 v 之间的路径个数
22. p=find an edge-disjoint path instance from u to v; //发现一条从 u 到 v 的边不相交路径实例
23. p_{uv}++;
24. Remove all edgs in p from G_H; //从 G_H 中删除 p 中的所有边
25. **WHILE** p is not empty;
26. **IF** $p_{uv}>0$
27. Add an edge between different vertices u and v, and allocate the weight w as p_{uv};
28. **END FOR**
29. **END FOR** //确定最终的边集 E_T 及权重 w
30. **RETURN** $G_P=(V_T,E_T,w)$; //返回无向加权图 G_P

以图 1(a)中的 HIN 为例: 给定 $q=a_1$, $P=P_1$ 和$\theta=5$, 构建该 HIN 对应的无向加权图 G_{P_1}. 使用 GP-BUILD 算法的执行过程如下.

- 首先计算出图 G_{P_1} 的顶点集 $V_T=\{a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8,a_9,a_{10},a_{11},a_{12},a_{13}\}$. 注意, 这里的 V_T 中并不包括 a_{14}, 这是因为该点并不$(P_1,5)$-连通于 a_1;
- 然后计算 V_T 中任意两不同顶点之间的边不相交路径个数, 确定边集 E_T 及其权重 w. 若顶点 a_1 和 a_2 之间的边不相交路径个数为 2, 则边 $w(a_1,a_2)=2$;
- 最后, 构建的无向加权图 G_{P_1} 如图 2 所示.

GP-BUILD 算法的最坏时间复杂度为 $O((|V_T|^2\cdot w_{\max}+1)\cdot\sum_{i=1}^{l}n_i\cdot d_{i,i+1})$. 首先, 该算法计算图 G_P 的顶点集 V_T 最坏要花费时间 $O(\sum_{i=1}^{l}n_i\cdot d_{i,i+1})$, 其中, n_i 表示具有元路径 P 中第 i 顶点类型的顶点个数, $d_{i,i+1}$ 表示与具有元路径 P 中第 i 顶点类型的顶点连通的具有 P 中第$(i+1)$顶点类型的顶点的最大个数.

为了方便, 本文用 $O(\sum_{i=1}^{l}n_i\cdot d_{i,i+1})$ 表示计算任意两不同顶点间路径的最坏时间, 则计算边集 E_T 及权重 w

花费 $O(|V_T|^2 \cdot w_{\max} \cdot \sum_{i=1}^{l} n_i \cdot d_{i,i+1})$ 的时间，其中，$w_{\max}$ 表示图 G_P 中的最大权重．最后，算法 1 的总时间复杂度为 $O((|V_T|^2 \cdot w_{\max}+1) \cdot \sum_{i=1}^{l} n_i \cdot d_{i,i+1})$.

3 基于图分解的 *k*-PCCs 发现算法

在上节构建的无向加权图 G_P 上，给定正整数 k($k \geqslant 2$)，本节主要利用图分解的方法从图 G_P 中发现所有的 *k*-PCCs．通过分析无向无权图中的 *k*-边连通分量发现算法，本节首先给出基于图分解的 *k*-PCCs 发现框架 KPCC-FIND，然后详细分析图分解的方法，最后设计优化的图分解算法 DEC-OPT.

3.1 基于图分解的*k*-PCCs发现框架

本节在给出基于图分解的 *k*-PCCs 发现框架之前，首先阐述一些关于 *k*-PCCs 的有趣定理.

定理 1．给定图 G_P 和正整数 k，一个 *k*-PCC 是极大的.

证明：假设 *k*-PCC g 不是极大的，则意味着可增加图 G_P 中一个或多个顶点到 g，其仍为 *k*-PCC．显然，这与 *k*-PCC 定义中的条件(2)其任意母图不再是 *k*-PCC 不符．证毕． □

定理 2．给定图 G_P 和正整数 k，对于任意两个 *k*-PCCs g_1 和 g_2，若 $g_1 \cap g_2 \neq \varnothing$，则 $g_1=g_2$.

证明：假设 $g_1 \neq g_2$，则通过定理 1 可知，$g_1 \cup g_2$ 为更大的 *k*-PCC．这与 g_1 和 g_2 是极大的不符．证毕． □

KPCC-FIND 框架的基本思想是，迭代地将一个不是 *k*-PCC 的图分解为一个或多个 *k*-PCC 子图．具体地，首先，给定一个无向加权连通图 G_P，只要有它的连通子图不是 *k*-PCC，就尝试将它们分解为更小的子图集．当所有的连通子图都已是 *k*-PCC 时，该框架停止计算．在此过程中产生的所有连通子图(包括中间子图和最终的 *k*-PCCs)可组织为如图 3 所示的分解树，树中的每个椭圆代表一个连通子图．其中，根为输入图 G_P，每个连通子图的孩子为通过图分解产生的连通子图集．最终，图 G_P 的所有 *k*-PCCs 为分解树中的叶子节点．KPCC-FIND 框架在算法 2 给出，它通过在分解树上进行广度优先搜索得到所有的 *k*-PCCs．其中，图分解的 DEC 算法在后文第 3.2 节和第 3.3 节详细加以讨论.

算法 2．基于图分解的 *k*-PCCs 发现框架(KPCC-FIND).

输入：无向加权图 G_P、正整数 k;

输出：图 G_P 的所有 *k*-PCCs.

1. $Q_g=\{G_P\}$; //初始化一个包含 G_P 的队列 Q_g
2. **FOR** every subgraph g in Q_g
3. $\vartheta_k(g)=DEC(g,k)$; //将子图 g 分解为 *k*-PCCs
4. **IF** $\vartheta_k(g)$ composes of just one subgraph //若 $\vartheta_k(g)$中仅有一个子图
5. **OUTPUT** $\vartheta_k(g)$, and it is one of the *k*-PCCs;
6. **ELSE**
7. **PUSH** all of subgraphs of $\vartheta_k(g)$ to Q_g;
8. **END FOR**

KPCC-FIND 算法的时间复杂度为 $O(h \cdot Tc)$，其中，h 表示分解树的高度，Tc 表示用图分解算法分解图 G_P 的时间复杂度，在后文第 3.3 节给出详细分析.

3.2 图分解方法

本节首先给出图分解的相关概念及图分解的过程，然后依据图分解的方法设计图分解算法 DEC.

定义 8(割(*S*,*T*))．给定无向加权图 $G_P=(V_T,E_T,w)$，割 $C=(S,T)$将 V_T 划分为两个非空、不相交的子集 S 和 T，即 $S \cup T=V_T$, $S \cap T=\varnothing$.

同时，割也可用在不相交子集中顶点之间的边集合表示，即$\{(u,v) \in E_T | u \in S, v \in T\}$．割的值定义为割中所有边的权重和，即 $f(C)=f(S,T)=\sum\{w(u,v) | (u,v) \in E_T | u \in S, v \in T\}$.

定义 9(*s*-*t* 割). 如果顶点 s 和 t 分别在不相交子集 S 和 T 中, 则割 $C=(S,T)$也称为 s-t 割; 且如果该 s-t 割的值不大于任何其他 s-t 割的值, 则称该割为最小 s-t 割.

令$\lambda(s,t)$表示图 G_P 中最小 s-t 割的值, 则该$\lambda(s,t)$也是顶点 s 和 t 之间的路径连通度. 如果$\lambda(s,t)\geqslant k$, 则称两顶点 s 和 t 为 k-路径连通.

定义 10(最小割). 图 G_P 的全局最小割为图 G_P 中所有割中具有最小割值的割.

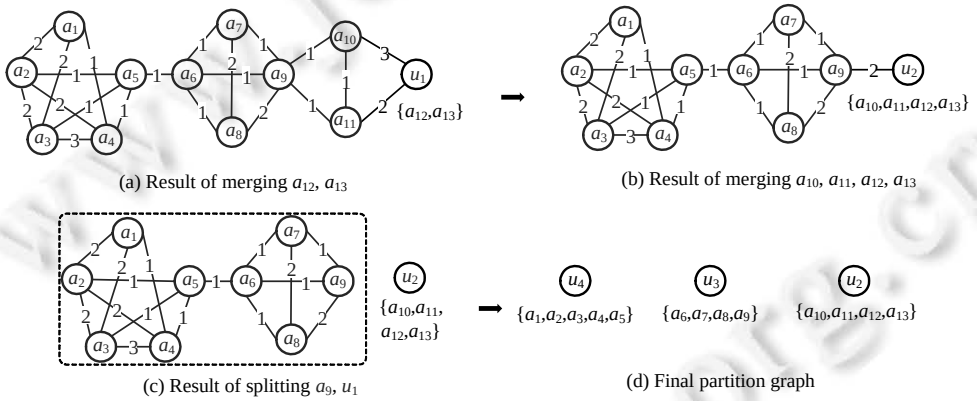
令$\lambda(G_P)$表示图 G_P 的全局最小割的值, 则该$\lambda(G_P)$也是图 G_P 的路径连通度, 即$\lambda(G_P)=\min_{s,t\in VT,s\neq t}\lambda(s,t)$. 如果$\lambda(G_P)\geqslant k$, 则称图 G_P 为 k-PCC. 例如: 割 $C_1=\{(a_5,a_6)\}$和割 $C_2=\{(a_9,a_{10}),(a_9,a_{11})\}$均为以图 2 所示图的割, 其中, C_1 为全局最小割, C_2 为最小 a_6-a_{13} 割.

定义 11(分区图(partition graph, PG)). 图 $G_P=(V_T,E_T,w)$的分区图 $PG=(G_P,D)$是通过给每个顶点 $u\in V_T$ 增加一个域 D 的元素集而得到的, 其中, $\bigcup_{u\in V_T}D(u)=D$, $D(u)\cap D(v)=\varnothing$, $\forall u\neq v$, 这里的 $D(u)$表示与顶点 $u\in V_T$ 相关的元素集. 也就是说, 与 V_T 中顶点相关的元素集组成了 D 的一个分区.

一个无向加权图 $G_P=(V_T,E_T,w)$是一个特殊的 $D=V_T$ 的分区图, 其中, $D(u)=\{u\}$, $\forall u\in V_T$. 因此, k-PCCs 发现算法的输入是一个分区图 $PG=(G_P=(V_T,E_T,w),D(=V_T))$, 其中, $D(u)=\{u\}$, $\forall u\in V_T$.

定义 12(合并算子). 合并算子$\beta_{u,v}(PG)$是将 PG 中的两个顶点 u 和 v 合并为一个超顶点 u_{uv}. 具体地, 该算子首先将 u_{uv} 添加到 $D(u_{uv})=D(u)\cup D(v)$的 PG 中; 然后, 对于每个 $x\in\{x|(u,x)\in E_T,(v,x)\in E_T\}$, 增加 u_{uv} 和 x 之间的边(u_{uv},x), 并赋权重为 $w(u,x)+w(v,x)$; 最后, 从 PG 中删除顶点 u 和 v 以及它们的相关边.

例如: 图 4(a)为合并算子 $\beta_{a_{12},a_{13}}(PG)$ 的结果, 其中, PG 为图 2 所示的分区图. 在合并过程中, 首先将 a_{12}, a_{13} 合并为 u_1; 然后增加 u_1 和 x 之间的边, 其中, x 包含 a_{10} 和 a_{11}, 则增加边(a_{10},u_1)并赋权重为 $w(a_{12},a_{10})+w(a_{13},a_{10})=1+2=3$ 和边(a_{11},u_1), 并赋权重为 2; 最后, 从 PG 中删除顶点 a_{12}、a_{13} 以及相关的边.



(a) Result of merging a_{12}, a_{13} (b) Result of merging a_{10}, a_{11}, a_{12}, a_{13}

(c) Result of splitting a_9, u_1 (d) Final partition graph

图 4 图分解为分区图的过程

定义 13(分割算子). 分割算子$\gamma_{u,v}(PG)$是从 PG 中删除割 C 中的所有边, 其中, $C=(S,T)$为分区图 PG 的一个割.

例如: 图 4(c)为分割算子 $\gamma_{a_9,u_2}(PG)$ 的结果, 其中, PG 为图 4(b)所示的分区图, a_9-u_2 割为其 PG 的一个割.

给定任意一个分区图 $PG=(G=(V,E,w),D)$, 通过执行合并算子或分割算子得到的结果图仍为分区图. 为了简化预处理, 本文余下部分将用$\beta_{s,t}(PG)$和$\gamma_{S,T}(PG)$分别表示执行对应算子后得到的分区图.

DEC 算法的总体思路为: 在分区图 $PG=(G_P=(V_T,E_T,w),D(=V_T))$上反复执行合并和分割算子, 直到得到一个特殊的分区图, 即 $E_T^r=\varnothing$ 的 $PG^r=(G_P^r=(V_T^r,E_T^r,w),D(=V_T))$, 其中, 每个 $D(u),u\in V_T^r$ 均会诱导出图 G_P 的一个连通子图. 令 $\vartheta_k(PG^r)$表示 PG^r 所代表的连通子图的集合, 即 $\vartheta_k(PG^r)=\{G_P[D(u)]\,|\,u\in V_T^r\}$, 则 DEC 算法返回 $\vartheta_k(PG^r)$作为图分解以后的 k-PCCs. 以图 2 所示的连通图为例: 如果对它进行 k=3 的分解, 则最终的分区图为图 4(d)的形式, 即 $\vartheta_3(G_{P_1})=\{g_1,g_2,g_3\}$. DEC 算法如算法 3 所示: 给定输入图 G_P 和正整数 k, 首先构建分区图

PG(第 1 行); 然后, 通过分割算子(第 5 行)和合并算子(第 7 行)迭代地更新分区图, 直到其边集为空(第 3–9 行); 最终返回所有的连通子图(第 10 行). 至于算子的选择, 其取决于 Mas-GP 算法计算出的顶点之间的最小割的值(第 3 行).

Mas-GP 算法是在 Mas 算法[19]的基础上提出的. Mas 算法可计算无向无权图的全局最小割, 其基本思想为: 先计算(|V|−1)对顶点的最小割, 再将具有最小割值的割记为全局最小割. 因本文是计算无向加权图中的全局最小割, 故提出 Mas-GP 算法. 具体地, 首先计算图 G_P 中所有顶点的一个排序并记为 L, 令 t 为 L 中的最后一个顶点, s 为 L 中 t 前面的一个顶点; 然后, 存在一个性质是图 G_P 中 t 的邻接边即为最小 s-t 割[19]. 其中, L 的创建如下: 首先, 初始化一个包含 V_T 中任意一个顶点的 L; 然后, 只要有顶点没有包括在 L 中, 就将与 L 连接最紧密的顶点 u 添加到 L 的尾部, 即 $u=\arg\max_{v\in V_T\setminus L_y}(L,v)$, 这里的 $y(L,v)$为顶点 v 与 L 中顶点之间边的权重和. 如果令 s 和 t 为最近添加到 L 中的两个顶点, 则($L\setminus\{t\},\{t\}$)为一个最小 s-t 割; 而如果用斐波那契堆来找与 L 连接最紧密的顶点, 那么 Mas-GP 算法的时间复杂度为 O(|E_T|+|V_T|·log|V_T|).

算法 3. 图分解(DEC).

1. 构建图 G_P 的分区图 PG, 即 PG^0=((G^0=G_P),(D=V_T)), i=0;
2. **WHILE** PG^i's edge set is not empty
3. (s,t,S,T)=*Mas-GP*(PG^i,k); //计算 PG^i 中的最小割
4. **IF** $f(S,T)<k$
5. $PG^{i+1}=\gamma_{s,t}(PG^i)$;
6. **ELSE**
7. $PG^{i+1}=\beta_{s,t}(PG^i)$;
8. i=i+1;
9. **END WHILE**
10. **RETURN** $\vartheta_k(PG^i)$;

例如, 图 4 展示了以图 2 所示的连通图作为输入并给定 k=3 的图分解过程. 具体地, 假设首先根据最小割的计算确定第 1 次迭代是在顶点 a_{12} 和 a_{13} 之间进行, 又因其割的值为 4>3, 所以 a_{12} 和 a_{13} 被合并且得到如图 4(a)所示的分区图. 然后, 图 4(b)为合并 a_{10}、a_{11} 和 u_1 之后的分区图. 接着, 下一次迭代计算出 a_9 和 u_2 之间的割为 C={(a_9,u_2)}, 并且该割的值为 2<3, 故选择分割算子删除此割中的所有边, 便得到如图 4(c)所示的矩形虚线包围的子图和孤立点 u_2. 继续迭代执行合并算子和分割算子, 最终得到如图 4(d)所示的 3 个 3-PCC 子图.

DEC 算法的最坏时间复杂度为 O(|V_T|·|E_T|+|V_T|^2·log|V_T|). 因为 DEC 算法最多需要执行(|V_T|−1)次 Mas-GP 算法的迭代, 所以该算法的最坏时间复杂度为 O(|V_T|·|E_T|+|V_T|^2·log|V_T|).

3.3 DEC-OPT算法

本节主要研究无向加权图中优化的图分解算法 DEC-OPT. 首先介绍一种有效的数据结构[18], 并将其应用到无向加权图中的图分解算法中; 然后, 在此基础上提出两个优化策略, 进一步提高图分解算法的效率; 最后提出无向加权图中优化的图分解算法 DEC-OPT.

(1) 无向加权图中的数据结构

DEC 算法时间复杂度中的|V_T|·log|V_T|是由于 Mas-GP 算法使用斐波那契堆查找与 L 连接最紧密的顶点所导致的, 具体原因是, Mas-GP 算法使用斐波那契堆需要两个操作: 一是花费 O(1)时间的更新键值(update key, UK), 二是花费 O(log|V_T|)时间的获取最大值(extract max, EM). 但是, 斐波那契堆比较复杂, 并且在真实大图中效率比较低, 于是介绍一种有效的数据结构, 并将其应用到无向加权图中的图分解算法中.

- 数据结构: 令 $key(v)=y(L,v)$, 其表示 Mas-GP 算法执行过程中顶点 v 的键. 令 $c_{\max}$ 表示不同顶点对之间所有最小割中的最大割值, 则有 $key(v)\leqslant c_{\max}$, $\forall v\in V_T$, 且 $c_{\max}\leqslant|E_T|$. 然后, 可用一个带有头表的双链表实现 UK 操作和 EM 操作. 对于双链表, 可用 3|V_T|的空间存储数据, 即对于每个顶点 $u\in|V_T|$, $key(u)$存储键、$pre(u)$和 $next(u)$分别存储双链表中顶点 u 的前驱顶点和后继顶点, 其中, 每个双链表连接着

所有具有相同键值的顶点. 对于头表 H, $H(x)$存储双链表中键值等于 x 的第 1 个顶点. 因为最大的键值可能为 $c_{\max}$, 所以 H 的大小最大为 $c_{\max}$. 另外, 定义一个初值为 0 的 po 记录数据结构中所有顶点当前可能的最大键值, 并且 po 只在 UK 操作后更新;

- EM 操作: 为了获取下一个键值最大的顶点, 首先减少 po 直到 $H(po)\neq nil$, 然后记录 $H(po)$指向的第 1 个顶点, 并将其从双链表中删除;
- UK 操作: 为了将顶点 v 的键值从 x 更新到 y, 首先将 v 从 $H(x)$指向的双链表中删除; 然后, 将 v 插入到双链表 $H(y)$中; 最后, 如果 $y>po$, 则更新 $po=y$.

在新的数据结构上, UK 操作的最坏时间复杂度为 O(1), 这是由双链表的特性来保证的; EM 操作的最坏时间复杂度为 $O(c_{max})\leqslant O(|E_T|)$, 这是因为它需要减少 po 的值直到 $H(po)\neq nil$, 而且 po 可能最大为 $c_{\max}$, 唯一的非空双链表可能为 $H(1)$. 最终, Mas-GP 算法计算任意一对顶点之间的最小割需要花费 $O(|E_T|)$的时间.

(2) DEC-OPT 算法

将新的数据结构和经典的更早合并和共享链表两个优化策略[18]应用到无向加权图中的图分解算法中, 则以图 2 所示的连通图并给定 k=3 的图分解过程见表 2. 但此时存在一个问题: 在初始化 L 时就已将 V_T 中的一个顶点加入 L, 这样会导致在执行合并和分割算子时, 使该点无法与其他顶点进行更早的合并和分割, 从而使迭代次数增加. 为此, 本文提出第 1 个优化策略: 初始化 L 为空且不提前将 V_T 中的任意一个顶点加入 L, 并定义任意一个顶点到空 L 的键值为 0, 即 $key(v)=y(L,v)=0$, $\forall v\in V_T$, $L=\varnothing$. 接着, 提出第 2 个削减顶点的优化策略: 在一个 k-PCC 中, 每个顶点的所有邻接边权重之和应不小于 k, 所以可删除输入图中顶点邻接边权重和小于 k 的顶点. 最后, 将所有优化策略应用到无向加权图的图分解算法中, 并设计如算法 4 所示的 DEC-OPT 算法.

表 2 优化前的执行

迭代	列表 L
1	$a_1,\{a_2,a_3,a_4\},a_5,a_6,\{a_7,a_8\},a_9,a_{10},\{a_{11},a_{13}\},a_{12}$
2	$a_1,\{a_2,a_3,a_4,a_5\},a_6,\{a_7,a_8,a_9\},\{a_{10},a_{11},a_{13}\},a_{12}$
3	$a_1,\{a_2,a_3,a_4,a_5\},\{a_6,a_7,a_8,a_9\},\{a_{10},a_{11},a_{13},a_{12}\}$
4	$\{a_1,a_2,a_3,a_4,a_5\}$

算法 4. 图分解优化(DEC-OPT).

输入: 无向加权图 G_P、正整数 k;

输出: 如果$\lambda(G_P)<k$, 则返回 G_P 的子图; 否则, 返回 G_P.

```
1.  构建 G_P 的分区图 PG, 即 PG^0=((G^0=G_P),(D=V_T)), i=0;
2.  WHILE PG^i's edge set is not empty
3.     PG^{i+1}=Mas-GP-OPT(PG^i,k);
4.     i=i+1;
5.  END WHILE
6.  RETURN ϑ_k(PG^i);
7.  Procedure Mas-GP-OPT(PG^i,k)
8.  L=∅;   //初始化空集 L
9.  Initialize the data structure;   //初始化数据结构
10. WHILE L≠|V_T|
11    u=EM
12.    Add u to L and remove u from the data structure;   //添加顶点 u 到 L 并从数据结构中删除 u
13.    Q.add(u);   //初始化队列 Q, 并将顶点 u 加入
14.    WHILE Q≠∅
15.       v=Q.pop(·);
16.       FOR each (v,t)∈E_T with t∉L
```

17. **IF** $key(t)\geqslant k$
18. $Q.add(t)$, and remove t from the data structure;
19. **ELSE**
20. UK for t;
21. **END FOR**
22. Merge u and v, if $u\neq v$;
23. **END WHILE**
24. **END WHILE**
25. **WHILE** $|L|>1$ and the value of this cut of the final two vertices in L is smaller than k //执行分割操作
26. Split this cut;
27. Remove the final vertex from L;
28. **END WHILE**

算法 4 给出了 DEC-OPT 算法的具体描述. 在程序 Mas-GP-OPT 中, 首先初始化 L 为空而不是初始化有 V_T 中任意一个顶点 u 的 L, 这样就不会导致 u 不会与其他顶点提前合并或分割; 然后, 队列 Q 中的顶点迭代地进行合并操作, 直到 $L=V_T$(第 10–23 行); 最后, 如果 L 中最后两个顶点的最小割值小于 k, 则迭代地执行分割操作(第 25–28 行). 同样地, 以图 2 所示的连通图为例并给定 k=3, 优化的图分解过程见表 3. 可以看到: 优化后的迭代次数少于优化之前的迭代次数, 提高了图分解算法的效率. DEC-OPT 算法的时间复杂度为 $O(r\cdot|E_T|)$, 其中, $r(<<|V_T|)$[18]为 Mas-GP-OPT 的迭代次数, 即算法 4 中第 6 行中 i 的值.

表 3 DEC-OPT 的执行

迭代	列表 L
1	$a_1,\{a_2,a_3,a_4\},a_5,a_6,\{a_7,a_8\},a_9,a_{10},\{a_{11},a_{13}\},a_{12}$
2	$\{a_1,a_2,a_3,a_4\},a_5,a_6,\{a_7,a_8,a_9\},\{a_{10},a_{11},a_{13}\},a_{12}$
3	$\{a_1,a_2,a_3,a_4,a_5\},\{a_6,a_7,a_8,a_9\},\{a_{10},a_{11},a_{13},a_{12}\}$

4 SMPCC 查询算法

基于上一节提出的基于图分解的 k-PCCs 发现算法, 本节首先提出优化的 SMPCC 查询算法. 然后给出 HINs 中查询 SMPCC 的总时间复杂度分析, 最后深入讨论大规模 HINs 中 SMPCC 的高效查询问题.

4.1 优化的SMPCC查询算法

本小节提出 SMPCC 查询算法. 算法的思想: 由定义 7 可知, SMPCC 为图 G_P 中包含 q 的具有最大路径连通度的 k-PCC. 显然, q 的任意 SMPCC 的路径连通度 $spc(q)$不可能大于图 G_P 所有边的权重和. 所以最基本的想法是: 先从图 G_P 的所有边的权重和开始, 依次递减计算不同 k 值的 k-PCCs, 同时检查其 k-PCC 是否包含 q. 一旦存在一个包含 q 的 k-PCC, 就立刻返回该子图并停止计算. 通过这样的思想, 可保证该子图为包含 q 的具有最大路径连通度的 k-PCC, 从而找到 SMPCC. 进一步分析, 因为 k-PCC 中 k 的值受约束于 k-PCC 中顶点邻接边权重和的最小值, 所以 q 的任意 SMPCC 的 $spc(q)$不可能大于 q 的所有邻接边的权重和. 因此, 本节提出了定理 3 并设计了优化的 SMPCC 查询算法 SMPCC-OPT, 如算法 5 所示.

定理 3. 给定图 G_P 和查询点 q, q 的任意 SMPCC 的路径连通度 $spc(q)$的上界为 q 的所有邻接边的权重和为 $\sum w$.

证明: 因为 q 的任意 SMPCC 为包含 q 的具有最大路径连通度的 k-PCC, 所以该 SMPCC 肯定包含 q. 假设 $spc(q)$的上界大于 $\sum w$, 则意味着 q 除了当前的邻接边外, 还存在其他邻接边. 显然, 这与事实不符, 故上界为 $\sum w$. 证毕. □

算法 5. 优化 SMPCC 查询算法(SMPCC-OPT).

输入: 无向加权图 G_P、查询点 q;

输出: SMPCC.

1. K=∅; //K: 存储计算出的 k-PCCs
2. **FOR** $k=\sum w$ to 1
3. K=*KPCC-FIND*(G_P,k); //发现对应 k 的 k-PCCs
4. **IF** there is a subgraph g containing q in K //K 中存在包含 q 的子图
5. **RETURN** g as the SMPCC;
6. **END FOR**

算法 5 首先初始化 K 为空, 然后, k 从 $\sum w$ 开始依次递减计算对应 k 值的 k-PCCs 存储到 K, 同时检查 K 中是否有包含 q 的 k-PCC: 若有, 则将其作为 SMPCC 返回并停止计算; 否则继续查找. KPCC-FIND 算法的时间复杂度为 $O(h\cdot r\cdot|E_T|)$, 所以 SMPCC-OPT 算法的总时间复杂度为 $O(\sum w\cdot h\cdot r\cdot|E_T|)$, 其中, $\sum w$、h 和 r 均为较小的整数.

4.2 大规模SMPCC查询算法

经过以上分析, 在 HINs 中查询 SMPCC 首先需要计算无向加权图 $G_P=(V_T,E_T,w)$, 其最坏耗时为 $O((|V_T|^2\cdot w_{\max}+1)\cdot\sum_{i=1}^{l}n_i\cdot d_{i,i+1})$; 然后, 再在无向加权图中执行时间复杂度 $O(\sum w\cdot h\cdot r\cdot|E_T|)$ 的 SMPCC-OPT 算法. 那么最终在 HINs 中查询 SMPCC 的总时间复杂度 $T=O((|V_T|^2\cdot w_{\max}+1)\cdot\sum_{i=1}^{l}n_i\cdot d_{i,i+1}+\sum w\cdot h\cdot r\cdot|E_T|)$, 其中, $w_{\max}$ 表示图 G_P 中的最大权重, n_i 表示具有元路径 P 中第 i 顶点类型的顶点数, $d_{i,i+1}$ 表示与具有元路径 P 中第 i 顶点类型的顶点连通的具有 P 中第$(i+1)$顶点类型的顶点的最大个数.

定理 4. HINs 中查询 SMPCC 问题在多项式时间复杂度内可解.

证明: 首先, 在计算无向加权图的顶点集时, 仅计算了(P,θ)-连通于 q 的顶点, 因此$|V_T|<<n_1$; 其次, 通过深度优先搜索算法计算两点间是否有路径可达时, 计算的是与当前顶点匹配的元路径中下一顶点类型的顶点, 而并非匹配元路径中下一顶点类型的所有顶点, 故 $d_{i,i+1}<<n_{i+1}(1\leqslant i\leqslant l)$; 最终:

$$T<<O((n_1^2\cdot w_{\max}+1)\cdot\sum_{i=1}^{l}n_i\cdot n_{i+1}+\sum w\cdot h\cdot r\cdot|E_T|)=O(n_1^2\cdot w_{\max}\cdot\sum_{i=1}^{l}n_i\cdot n_{i+1}+\sum_{i=1}^{l}n_i\cdot n_{i+1}+\sum w\cdot h\cdot r\cdot|E_T|);$$

又因为 $w_{\max}$、$\sum w$、h、r 均为常数, 所以随着 $n_i(1\leqslant i\leqslant l)$的增长, $T<<O(w_{\max}\cdot n_1^2\cdot\sum_{i=1}^{l}n_i\cdot n_{i+1})$. 显然, 该问题在多项式时间复杂度内可解. 证毕. □

针对如何有效地在大规模 HINs 中查询 SMPCC 问题, 通过分析其时间复杂度可知, 它的高时间复杂度主要是由无向加权图的计算导致的. 为此, 本文第 2 节已通过仅计算(P,θ)-连通于 q 的顶点作为其顶点集, 将其时间复杂度中的 n_1 优化为$|V_T|$, 并记其无向加权图的构建算法为 GP-BUILD. 为了进一步提高效率且适应于大规模 HINs, 本文限制顶点集个数, 并在通过计算两顶点间的路径连通度确定边集时提出: 当两点间的路径连通度达到 $\sum w$ 时就不再计算. 这是因为由定理 3 得知, 任意 q 的 SMPCC 的路径连通度不超过 $\sum w$. 同时, 这里将通过优化顶点集和边集来构建无向加权图的算法记为 GP-WBUILD. 由于 GP-WBUILD 算法中计算的顶点集是(P,θ)-连通于 q 的顶点, 且边集中路径连通度的计算不会超过 $\sum w$, 另外, 边集中任意两点间是否由路径实例连接也只是计算θ跳路径可达, 所以 GP-WBUILD 的时间复杂度为 $O((|V_T|^2\cdot\sum w+1)\cdot\theta\cdot\sum_{i=1}^{l}c_i\cdot d_{i,i+1})$, 其中, c_i 表示θ跳路径中具有元路径 P 中第 i 顶点类型的最大顶点数. 那么使用 GP-WBUILD 构建无向加权图从而查询 SMPCC 优化算法的时间复杂度为 $O((|V_T|^2\cdot\sum w+1)\cdot\theta\cdot\sum_{i=1}^{l}c_i\cdot d_{i,i+1}+\sum w\cdot h\cdot r\cdot|E_T|)$, 其中,

$$\sum w<<w_{\max},\ \theta\cdot\sum_{i=1}^{l}c_i\cdot d_{i,i+1}<<\sum_{i=1}^{l}n_i\cdot n_{i+1}.$$

5 实验与评估

为了充分评估算法在不同 HIN 模式数据上的表现, 本文选取两种不同 HIN 模式共 5 个数据集作为输入, 然后给出 k-PCC 模型的有效性分析和优化的图分解算法及大规模 SMPCC 查询算法的效率分析.

5.1 数据集和实验设置

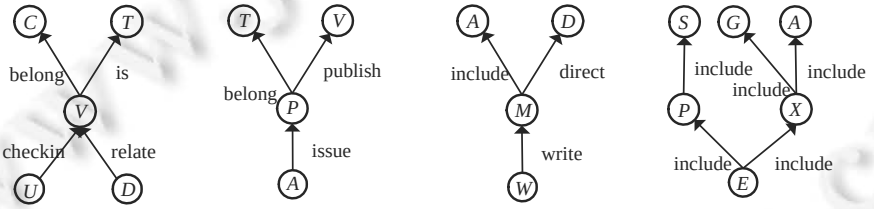
(1) 数据集

本文共采用了 4 个真实和 1 个合成的 HIN 数据集, 分别为 Foursquare、DBLP、IMDB、DBpedia 和 SynDBLP, 其中, 关于它们的顶点数、边数、顶点类型数、边类型数和元路径数的统计信息见表 4. 具体地, Foursquare、DBLP 和 IMDB 有相同的星形 HIN 模式, 其中, Foursquare 为包含 5 种顶点类型(用户(*U*)、城市(*C*)、日期(*D*)、场所(*V*)和场所类型(*T*))的签到记录; DBLP 为文献发表记录, 包括作者(*A*)、论文(*P*)、主题(*T*)和会场(*V*)这 4 种顶点类型; IMDB 为电影记录, 顶点类型有电影(*M*)、导演(*D*)、演员(*A*)和作家(W); DBpedia 具有丰富的 HIN 模式, 其包含维基百科中各个领域的数据; 合成数据集 SynDBLP 是根据真实 DBLP 的星形 HIN 模式随机生成的, 所以其顶点类型和边类型与 DBLP 的相同.

表 4 HIN 数据集

HIN	顶点数	边数	顶点类型数	边类型数	元路径数
Foursquare	43 199	405 476	5	4	20
DBLP	682 819	1 951 209	4	3	12
IMDB	2 467 806	7 597 591	4	3	12
DBpedia	5 900 558	17 961 887	413	637	1 000
SynDBLP	12 000 000	30 250 199	4	3	12

对于每个数据集中的元路径, 本文是根据每个数据集的 HIN 模式给出的. 首先, 图 5 展示了 4 个真实数据集的 HIN 模式, 其中, 由于 DBpedia 的 HIN 模式过于复杂, 这里只给出其部分模式; 而合成数据集 SynDBLP 是根据 DBLP 的 HIN 模式生成的, 所以其 HIN 模式和元路径都与 DBLP 的相同. 然后, 根据其 HIN 模式, 每个数据集中的元路径如下例所示: Foursquare 中的元路径 *P*=(*UVU*), DBLP 中的 *P*=(*VPAPV*), IMDB 中的 *P*=(*AMA*)以及 DBpedia 中的 *P*=(*EXGXE*)等.



(a) Foursquare (b) DBLP (c) IMDB (d) DBpedia

图 5 4 个真实数据集的 HIN 模式

(2) 查询设置

对于每个数据集上的查询, 首先任意选择 10 条元路径, 而根据已存在的工作[12−16], 所选择元路径的长度最长为 4. 然后, 在每条元路径下至少给定 10 个查询点以组成 10 次查询, 这样在每个数据集中至少有 100 次查询. 接着, 在每个数据集上执行这 100 个查询, 而查询的平均结果展示在后文第 5.2 节和第 5.3 节的图表中. 这样做的好处是减少了每次查询的偶然性.

(3) 对比算法

- FastBCore[12]: 在 HINs 中查询给定 *k* 的基于普通路径的(*k*,*P*)-核算法;
- BatchECore[12]: 在 HINs 中查询给定 *k* 的基于边不相交路径的(*k*,*P*)-核算法;
- DEC-BASE: 无向加权图中不加优化策略的图分解算法;
- DEC-OPT: 无向加权图中使用新数据结构并加上所有优化策略的图分解算法;
- SMPCC-BASE: 在 HINs 中基于 GP-BUILD 和 DEC-OPT 算法的 SMPCC 查询的基线算法;
- SMPCC-OPT: 在 HINs 中基于 GP-BUILD 和 DEC-OPT 算法的 SMPCC 查询的优化算法;
- SMPCC-BEST: 在 HINs 中基于 GP-WBUILD 和 DEC-OPT 算法的 SMPCC 查询的优化算法.

其中, FastBCore 和 BatchECore 这两个算法与本文的研究目的相同, 均是基于元路径从 HINs 中查询包含

q 的稠密子图, 且子图中所有顶点类型都与 q 的类型相同. 所不同的是, 上述两个算法是从最小度角度来考虑子图稠密性的, 而本文的 SMPCC-OPT 算法是从路径连通度来考量的, 所以选取了这两个算法作为对比算法来评估本文所提出模型和算法的有效性. 在算法效率方面, 通过比较 DEC-BASE 和 DEC-OPT 这两个算法以分析基于图分解方法中所提出的优化策略; 再通过比较 SMPCC-BASE、SMPCC-OPT 和 SMPCC-BEST 这 3 个算法来评估大规模 HINs 中的 SMPCC 查询方法.

(4) 实验环境

Windows 7 专业版 64 位操作系统, Intel(R) Core(TM) i5-6500 的 CPU, 主频 3.20 GHz, 16 G 内存; 开发平台为 Visual Studio 2019, 开发语言为 C++.

5.2 算法有效性分析

本节主要评估在 HINs 中提出的 k-PCC 模型的稠密性. 因本文是在 HINs 中查询稠密子图, 所以选择了以下 4 种度量方法来比较结果子图的稠密性.

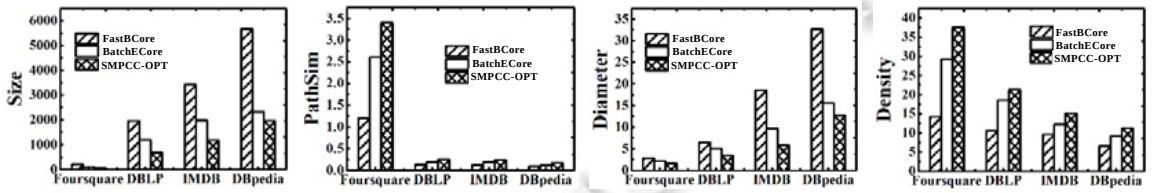
(1) Size: 结果子图中的顶点个数, 即大小;

(2) PathSim[14]: 结果子图基于元路径的相似度;

(3) Diameter[12]: 结果子图的直径;

(4) Density[12]: 结果子图的密度.

图 6 从 4 种角度评估了 FastBCore、BatchECore 和 SMPCC-OPT 这 3 个算法所查询结果子图的稠密性, 其中, FastBCore 和 BatchECore 中所使用的(k,P)-核模型中的 k 值设置与 SMPCC-OPT 查询出的路径连通度相同. 这是因为 SMPCC 是包含 q 的具有最大路径连通度的 k-PCC, 显然, 该最大路径连通度 k 是唯一的. 而 FastBCore 和 BatchECore 这两个算法仅仅只能查询给定 k 值的子图而并没有考虑最大路径连通度, 设置相同的 k 值(即最大路径连通度)便于评估本文所提出模型和算法的有效性; 同时, 修改算法也只查找(P,θ)-连通于 q 的顶点. 首先, 从图 6(a)中可以看出: 在 4 个真实数据集中, SMPCC-OPT 查询的结果子图的顶点数最少. 这是因为 k-PCC 模型增加了顶点之间边不相交路径个数的考虑, 削减了小于 $spc(q)$值的顶点. 然后, 从图 6(b)给出的相似度来看: 在小数据集 Foursquare 中, 3 个算法计算出的结果子图的相似度相对都比较高; 而在另外 3 个大数据集中, 由于它们的稀疏性, 所以计算出的相似度都远小于 Foursquare 中的相似度. 但从整体上看, SMPCC-OPT 查询结果的相似度都略高于 FastBCore 和 BatchECore 算法. 其次, 图 6(c)展示了结果子图的直径, 可以看到, 4 个数据集中 SMPCC-OPT 查询的结果子图最小. 这说明 k-PCC 模型优于其他模型. 最后, 从图 6(d)展示的结果子图的密度来看, 所查询的 SMPCC 密度最高. 这也说明 k-PCC 模型更加稠密. 综上, k-PCC 模型具有更高的稠密性, 从而验证了该模型的有效性.

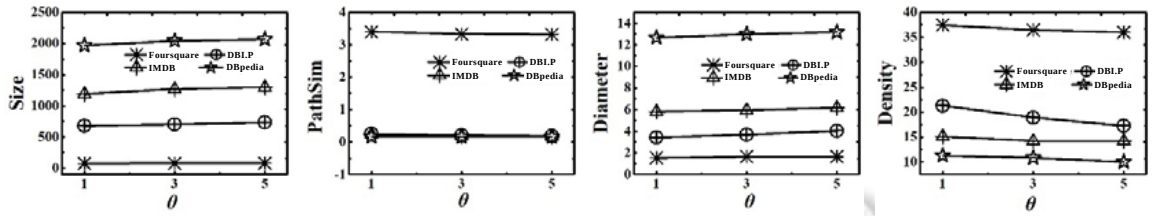


(a) Size (b) PathSim (c) Diameter (d) Density

图 6 k-PCC 的有效性

图 7 评估了 SMPCC-OPT 算法中不同θ值对 SMPCC 的影响. 因在构建无向加权图时计算的是(P,θ)-连通于 q 的所有顶点作为其顶点集, 所以参数θ会对 SMPCC 有一定的影响, 其中, 参数θ决定着 HINs 中某一顶点可通过θ跳路径实例到达另一顶点, θ值越小, 则说明这两点之间的关联性越强. 从图 7(a)给出的结果子图的大小可以看出: 随着θ值的增加, 结果子图的大小呈缓慢增长趋势最后趋于平缓. 其中, 增长趋势是因为较小的θ值会忽略关联性相对强的一些顶点, 而随着θ值的增加, 会将这些顶点考虑在内, 从而使结果子图的大小增加; 最后趋势平缓是因为较大的θ值会将关联性弱的一些顶点构建在无向加权图中, 使得在查询 SMPCC 时不

会将其发现. 从图 7(b)–图 7(d)计算的 3 个度量方法中可以看出: 随着θ值的增加, 结果子图的相似度和密度在下降, 直径在增加, 这同样是因为较大的θ值会将关联性弱的一些顶点考虑在内. 综上可见: SMPCC 的稠密性会随θ值的增加而增加, 但是超过一定值后就基本不影响 SMPCC 的有效性.



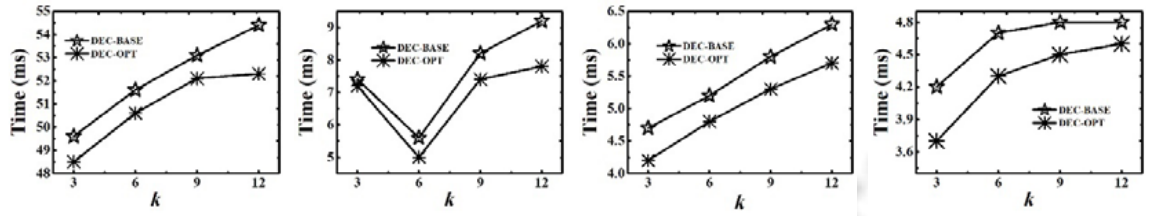
(a) Size (b) PathSim (c) Diameter (d) Density

图 7 不同θ值对 SMPCC 的影响

5.3 算法性能分析

本节主要验证所提出优化策略的高效性. 首先比较 DEC-BASE 和 DEC-OPT 这两个算法, 评估图分解的优化策略, 然后比较 SMPCC-BASE、SMPCC-OPT 和 SMPCC-BEST 这 3 个算法评测在大规模 HINs 中优化查询 SMPCC 方法.

图 8 给出了不同 k 值下 DEC-BASE 和 DEC-OPT 这两个算法在 4 个数据集上的运行时间. 从图 8 中首先可以看出: 不论 k 值是多少, DEC-OPT 算法的运行时间始终比 DEC-BASE 算法的要快. 这主要是因为第 3.3 节提出的两个优化策略通过初始化 L 为空, 使得顶点能够提早合并或分割从而减少迭代的次数; 同时, 由每个顶点的所有邻接边的权重和来削减顶点从而加快运行时间. 然后还可以看到: 随着k值的不断增大, 除了图 8(b)中 k=6 的情况, 两个算法的运行时间在不断加快. 原因主要是: 当 k 值增大时, 需要计算 k-PCC 中顶点之间更大的路径连通度, 从而需要更多的时间. 而图 8(b)中的情况主要是因为依据 IMDB 数据集构建的无向加权图在图分解的过程中顶点能够提早合并为 6-PCC, 从而使 Mas-GP-OPT 执行的次数降低, 时间减少.

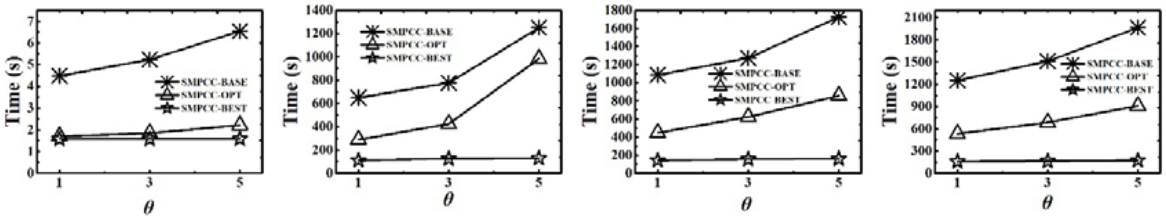


(a) Foursquare (b) IMDB (c) DBpedia (d) SynDBLP

图 8 不同 k 值下, DEC-BASE 和 DEC-OPT 算法的运行时间

图 9 展示了不同θ值下, SMPCC-BASE、SMPCC-OPT 和 SMPCC-BEST 这 3 个算法在 4 个数据集上的运行时间. 从结果中可以看出, SMPCC-OPT 算法一直优于 SMPCC-BASE 算法. 分析原因是: SMPCC-BASE 算法是从无向加权图中所有边的权重和开始查找 SMPCC 的, 而 SMPCC-OPT 是从其查询点 q 的所有邻接边的权重和开始的, 显然, 所有边的权重和远大于查询点 q 的所有邻接边的权重和, 所以 SMPCC-OPT 算法运行效率比较高, 同时也验证了查询 SMPCC 的优化策略. 针对适合大规模 HINs 上的 SMPCC-BEST 算法, 可以从图 9 中看出, SMPCC-BEST 算法在很大程度上比 SMPCC-OPT 和 SMPCC-BASE 两个算法的效率都高. 这是因为 SMPCC-BEST 算法在构建无向加权图时进行了双重优化, 不仅通过仅计算(P,θ)-连通于 q 的顶点来优化顶点集, 而且在确定边集计算路径连通度时, 通过应用定理 3, 最多只用计算到$\sum w$, 而不必计算两点间大于$\sum w$的路径连通度, 从而减少了大量时间的耗费. 横向来看: 随着θ值的增加, SMPCC-OPT 和 SMPCC-BASE 两个算法的运行时间都明显增加. 这是因为θ值的增加会使得无向加权图中任意两点之间计算的路径个数增加, 从而消耗时间. 但是可以看到: 随着θ值的增加, SMPCC-BEST 算法的运行时间略微增加但基本不受

θ值的影响. 这是因为在构建无向加权图时, 计算两点间的路径连通度至多计算到$\sum w$, 从而不需要花费大量时间计算随θ值增加而增大的路径连通度.

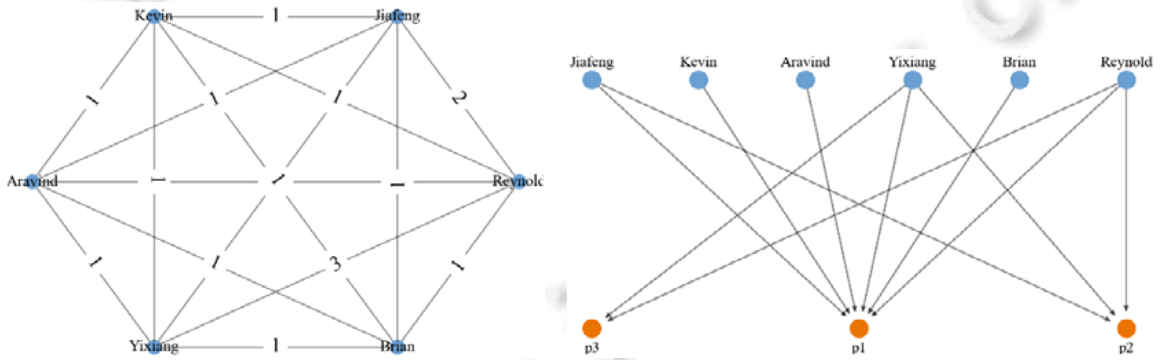


(a) Foursquare (b) IMDB (c) DBpeida (d) SynDBLP

图 9 不同θ值下, SMPCC-BASE、SMPCC-OPT 和 SMPCC-BEST 算法的运行时间

5.4 案例分析

与现有的研究[12,13]一致, 本文通过使用 2019–2020 年 3 个研究领域——数据库、人工智能和网络与信息安全的主要会议的论文发表情况构造一个 SmallDBLP 数据集, 其详细包括 28 055 位作者、11 207 篇论文、20 个会议和 4 839 个主题, 然后在 SmallDBLP 数据集中分析了真实的案例. 查询设置: q=Yixiang Fang(图数据领域的学者), P=(APA), θ=3, 则最终查询出的 SMPCC 如图 10(a)所示, 其为包含 Yixiang Fang 具有最大路径连通度的 5-PCC. 详细地, 来自香港大学的 Reynold Cheng 教授是 Yixiang Fang 的博士生导师, 而 Jiafeng Hu 和 Yixiang Fang 为同门, 皆是 Reynold Cheng 的博士生; 同时, Kevin Chen-Chuan Chang、Aravind Sankar 和 Brian Y. H. Lam 均为图数据领域的研究学者. 图 10(b)展示了图 10(a)对应的 HIN, 其中, p_1 为论文 Exploring Communities in Large Profiled Graphs, p_2 为论文 Discovering Maximal Motif Cliques in Large Heterogeneous Information Networks, p_3 为论文 Effective and Efficient Community Search Over Large Directed Graphs. 然后, 查询出的 6 位作者通过这 3 篇论文被紧密联系起来. 最后, 在对比算法 FastBCore 上设置了与 5-PCC 相同的 k 值为 5, 并修改算法也只查找(P,3)-连通于 q 的顶点. 最终进行了实验并得到如表 5 所示的结果. 可以看到: FastBCore 算法因为仅仅考虑了是否有元路径连接而查询到的作者个数比较多, 而本文提出的 SMPCC-OPT 算法查询出的作者个数较少且有合理的应用价值.



(a) SMPCC (5-PCC) (b) 对应图 10(a)的 HIN

图 10 SmallDBLP 中查询的 SMPCC 及对应的 HIN

表 5 在 SmallDBLP 上的案例结果

q=Yixiang Fang, P=(APA), θ=3	
FastBCore	SMPCC-OPT
Yixiang Fang Reynold Cheng Kevin Chen-Chuan Chang Jiafeng Hu Aravind Sankar Brian Y. H. Lam Other 161 authors	Yixiang Fang Reynold Cheng Kevin Chen-Chuan Chang Jiafeng Hu Aravind Sankar Brian Y. H. Lam

6 相关工作

图数据挖掘领域中的稠密子图查询已在同质网络和 HINs 中被广泛研究, 并产生了大量优秀的研究成果. 下面我们对这两种图中现有稠密子图查询的相关工作进行总结.

- 同质网络中的稠密子图查询.

同质网络中的稠密子图查询大都是将稠密子图建模为 k-核、k-捆、k-团、k-边连通分量以及基于密度的子图等模型. 首先, 最常用的是 k-核结构[6–9], 其要求子图内每个顶点的度数均不小于 k. 而随着图模型的发展, Fang 等人分别在时空图[20]和属性图[21–23]中提出了基于 k-核结构的稠密子图查询算法. 紧接着, 一些工作研究了 k-捆结构, 其从边的角度要求子图内的每条边至少被 k–2 个三角形包含[24,25]. 另外, 文献[17,26]解决了远离查询点的顶点包含在结果子图中的问题. 随后, k-团[27]和 k-边连通分量[18,28]被提出, 其中, k-边连通分量要求从其任意删除 k–1 条边后剩下的图仍连通. Stoer 等人[19]提出了基于割的方法; 但是由于其时间复杂度高, Chang 等人[16]于是提出了基于图分解的优化查询 k-边连通分量的算法; 进一步地, 文献[3,29]提出了基于索引查询包含查询点的具有最大连通度的 k-边连通分量, 文献[4]提出了通过扩展-优化框架查询最小的包含查询点的具有最大连通度的 k-边连通分量. 最后, 基于密度的子图模型被提了出来[30,31], 并设计了许多近似查询稠密子图的算法. 但是, 以上模型仅仅是在具有相同类型顶点和边的同质网络中分析的. 然而在 HINs 中, 所有的顶点和边都具有多种类型, 并具有复杂、丰富的语义信息. 所以上述模型均不能被直接应用于 HINs.

- HINs 中稠密子图查询.

对于 HINs 中的稠密子图查询问题, 首先, Wang 等人[32]根据 z-score 定义了 s-度的概念, 然后将 HINs 转化为同质网络并通过 s-度提取核结构 h-structure, 从而发现稠密子图; 由于其缺乏丰富语义, Jian 等人[11]通过限制顶点之间的连接个数提出了关系约束的核结构 r-com. 紧接着, 文献[10,33]通过将模体与团进行整合, 从而建模 HINs 中的稠密子图模型为最大模体团; 另外, Zhao 等人[34]从稠密子图问题的另一个角度推荐设计了基于模体的方法; 进一步地, Phillips 等人[35]和 Zhou 等人[36]分别提出了在 k-分图上发现最大 k-分团子图的模型. 从基于密度的角度看, Galimberti 等人[37,38]提出了通过核分解在多层网络中发现稠密子图的方法. 然而, 上述这些方法查询的子图具有多种类型的顶点, 导致其不能解决只关注某种特定类型顶点的场景. 随后, 文献[39,40]分别利用图谱方法和建模 k-context 结构, 从而发现稠密子图. 但其顶点间缺乏实际语义, 于是, Fang 等人[12,13]提出了一种基于元路径的方法. 该方法仅仅关注子图中某种特定类型的顶点, 即要求稠密子图中的顶点与 q 同类型并被给定元路径 P 的路径实例所连接. 但它在构建无向加权图时只要求子图内的顶点被给定元路径的路径实例所连接, 忽略了顶点之间基于元路径的连通度. 于是, 本文提出了基于元路径边不相交路径连通度的新方法, 并提出了 k-PCC 模型.

7 总结与展望

HINs 中的稠密子图查询问题已成为图数据挖掘领域的热点和重点研究问题, 并在许多领域有广泛应用. 为了在 HINs 中查询更加稠密的具有某种特定类型顶点的子图, 本文首先在 HINs 中提出了基于元路径的边不相交路径的连通度, 即路径连通度; 然后, 基于路径连通度提出了 k-PCC 模型, 该模型要求子图的路径连通度至少为 k; 其次, 基于 k-PCC 模型提出了 SMPCC 概念, 其为包含 q 的具有最大路径连通度的 k-PCC; 最后, 提

出了基于元路径的无向加权图构建算法, 在无向加权图中提出了一种高效的基于图分解的 *k*-PCC 发现算法, 并在此基础上提出了优化查询 SMPCC 算法和适应大规模的 SMPCC 查询算法. 大量基于真实和合成 HINs 数据的实验结果验证了本文所提出模型和算法的有效性和高效性. 未来的研究包括: (1) 可查询最小 SMPCC, 即查询具有最少顶点个数的 SMPCC; (2) 可尝试通过利用图神经网络技术来发现 HINs 中的稠密子图.

References:

[1] Fang YX, Wang K, Lin XM, Zhang WJ. Cohesive subgraph search over big heterogeneous information networks: Applications, challenges, and solutions. In: Proc. of the 2021 Int'l Conf. on Management of Data (SIGMOD). New York: ACM, 2021. 2829−2838.

[2] Shi C, Wang RJ, Wang X. Survey on heterogeneous information networks analysis and applications. Ruan Jian Xue Bao/Journal of Software, 2022, 33(2): 598−621 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6357.htm [doi: 10.13328/j.cnki.jos.006357]

[3] Chang LJ, Lin XM, Qin L, Yu JX, Zhang WJ. Index-based optimal algorithms for computing Steiner components with maximum connectivity. In: Proc. of the 2015 Int'l Conf. on Management of Data (SIGMOD). New York: ACM, 2015. 459−474.

[4] Hu JF, Wu XW, Cheng R, Luo SQ, Fang YX. Querying minimal Steiner maximum-connected subgraphs in large graphs. In: Proc. of the 25th ACM Int'1 on Conf. on Information and Knowledge Management (CIKM). New York: ACM, 2016. 1241−1250.

[5] Meysman P, Saeys Y, Sabaghian E, Bittremieux W, Peer YVD, Goethals B, Laukens K. Mining the enriched subgraphs for specific vertices in a biological graph. IEEE/ACM Trans. on Computational Biology and Bioinformatics, 2016, 16(5): 1496−1507.

[6] Dudley JT, Deshpande T, Butte AJ. Exploiting drug-disease relationships for computational drug repositioning. Briefings in Bioinformatics, 2011, 12(4): 303−311.

[7] Pesantez-Cabrera P, Kalyanaraman A. Efficient detection of communities in biological bipartite networks. IEEE/ACM Trans. on Computational Biology and Bioinformatics, 2017, 16(1): 258−271.

[8] Bonchi F, Khan A, Severini L. Distance-generalized core decomposition. In: Proc. of the 2019 Int'1 Conf. on Management of Data (SIGMOD). New York: ACM, 2019. 1006−1023.

[9] Zhang YK, Yu JX, Zhang Y, Qin L. A fast order-based approach for core maintenance. In: Proc. of the 33rd IEEE Int'1 Conf. on Data Engineering (ICDE). IEEE, 2017. 337−348.

[10] Hu JF, Cheng R, Chang KCC, Sankar A, Fang YX, Lam BYH. Discovering maximal motif cliques in large heterogeneous information networks. In: Proc. of the 35th IEEE Int'1 Conf. on Data Engineering (ICDE). IEEE, 2019. 746−757.

[11] Jian X, Wang Y, Chen L. Effective and efficient relational community detection and search in large dynamic heterogeneous information networks. Proc. of the VLDB Endowment, 2020, 13(10): 1723−1736.

[12] Fang YX, Yang YX, Zhang WJ, Lin XM, Cao X. Effective and efficient community search over large heterogeneous information networks. Proc. of the VLDB Endowment, 2020, 13(6): 854−867.

[13] Yang YX, Fang YX, Lin XM, Zhang WJ. Effective and efficient truss computation over large heterogeneous information networks. In: Proc. of the 36rd IEEE Int'1 Conf. on Data Engineering (ICDE). IEEE, 2020. 901−912.

[14] Sun YZ, Han JW, Yan XF, Yu PS, Wu TY. PathSim: Meta path-based top-*k* similarity search in heterogeneous information networks. Proc. of the VLDB Endowment, 2011, 4(11): 992−1003.

[15] Huang ZP, Zheng YD, Cheng R, Sun YZ, Mamoulis N, Li X. Meta structure: Computing relevance in large heterogeneous information networks. In: Proc. of the 22nd ACM SIGKDD Int'1 Conf. on Knowledge Discovery and Data Mining (SIGKDD). New York: ACM, 2016. 1595−1604.

[16] Zhao H, Yao QM, Li JD, Song YQ, Lee DL. Meta-graph based recommendation fusion over heterogeneous information networks. In: Proc. of the 23rd ACM SIGKDD Int'1 Conf. on Knowledge Discovery and Data Mining (SIGKDD). New York: ACM, 2017. 635−644.

[17] Huang X, Cheng H, Qin L, Tian WT, Yu XJ. Querying *k*-truss community in large and dynamic graphs. In: Proc. of the 2014 ACM SIGMOD Int'1 Conf. on Management Data (SIGMOD). New York: ACM, 2014. 1311−1322.

[18] Chang LJ, Yu JX, Qin L, Lin XM, Liu CF, Liang WF. Efficiently computing *k*-edge connected components via graph decomposition. In: Proc. of the 2013 ACM SIGMOD Int'1 Conf. on Management Data (SIGMOD). New York: ACM, 2013. 205−216.

[19] Stoer M, Wagner F. A simple min-cut algorithm. Journal of the ACM, 1997, 44(4): 585−591.

[20] Fang YX, Cheng R, Li XD, Luo SQ, Hu JF. Effective community search over large spatial graphs. Proc. of the VLDB Endowment, 2017, 10(6): 709−720.

[21] Fang YX, Cheng R, Luo SQ, Hu JF. Effective community search for large attributed graphs. Proc. of the VLDB Endowment, 2016, 9(12): 1233−1244.

[22] Li RH, Qin L, Ye FH, Yu JX, Xiao XK, Xiao N, Zheng ZB. Skyline community search in multi-valued networks. In: Proc. of the 2018 Int'1 Conf. on Management of Data (SIGMOD). New York: ACM, 2018. 457−472.

[23] Wang K, Cao X, Lin XM, Zhang WJ, Qin L. Efficient computing of radius-bounded *k*-cores. In: Proc. of the 34th IEEE Int'1 Conf. on Data Engineering (ICDE). IEEE, 2018. 233−244.

[24] Chen L, Liu CF, Zhou R, Li JX, Yang XC, Wang B. Maximum co-located community search in large scale social networks. Proc. of the VLDB Endowment, 2018, 11(10): 1233−1246.

[25] Zhang YK, Yu JX. Unboundedness and efficiency of truss maintenance in evolving graphs. In: Proc. of the 2019 Int'1 Conf. on Management of Data (SIGMOD). New York: ACM, 2019. 1024−1041.

[26] Huang X, Lakshmanan LVS, Yu JX, Cheng H. Approximate closest community search in networks. Proc. of the VLDB Endowment, 2015, 9(4): 276−287.

[27] Yuan L, Qin L, Zhang WJ, Chang LJ, Yang JY. Index-based densest clique percolation community search in networks. IEEE Trans. on Knowledge and Data Engineering, 2017, 30(5): 922−935.

[28] Zhou R, Liu CF, Yu JX, Liang WF, Chen BC, Li JX. Finding maximal *k*-edge-connected subgraphs from a large graph. In: Proc. of the 15th Int'1 Conf. on Extending Database Technology (EDTB). New York: ACM, 2012. 480−491.

[29] Chen ZY, Chen W, Jia Y, Zhou JF. The KST index based querying algorithm for Steiner maximum connected components. Chinese Journal of Computers, 2020, 43(7): 1215−1229 (in Chinese with English abstract).

[30] Fang YX, Yu KQ, Cheng R, Lakshmanan LVS, Lin XM. Efficient algorithms for densest subgraph discovery. Proc. of the VLDB Endowmentm, 2019, 12(11): 1719−1732.

[31] Li Y, Fan XL, Sun J, Zhao YH, Wang GR. Detecting statistically significant events in large heterogeneous attribute graphs via densest subgraphs. In: Proc. of the Int'1 Conf. on Knowledge Science, Engineering and Management (KSEM). Springer, 2020. 107−120.

[32] Wang RW, Ye FY. Simplifying weighted heterogeneous networks by extracting *h*-structure via *s*-degree. Scientific Reports, 2019, 9(1): 1−8.

[33] Li BX, Cheng R, Hu JF, Fang YX, Ou M, Luo RB, Chang KCC, Lin XM. Mc-explorer: Analyzing and visualizing motif-cliques on large networks. In: Proc. of the 36th IEEE Int'1 Conf. on Data Engineering (ICDE). IEEE, 2020. 1722−1725.

[34] Zhao H, Zhou YQ, Song YQ, Lee DL. Motif enhanced recommendation over heterogeneous information network. In: Proc. of the 28th ACM Int'1 Conf. on Information and Knowledge Management (CIKM). New York: ACM, 2019. 2189−2192.

[35] Phillips CA, Wang K, Baker EJ, Bubier JA, Chesler EJ, Langston MA. On finding and enumerating maximal and maximum *k*-partite cliques in *k*-partite graphs. Algorithms, 2019, 12(1): 23.

[36] Zhou A, Wang Y, Chen L. Finding large diverse communities on networks: The edge maximum k^*-partite clique. Proc. of the VLDB Endowment, 2020, 13(11): 2576−2589.

[37] Galimberti E, Bonchi F, Gulllo F. Core decomposition and densest subgraph in multilayer networks. In: Proc. of the 26th ACM Int'1 Conf. on Information and Knowledge Management (CIKM). New York: ACM, 2017. 1807−1816.

[38] Galimberti E, Bonchi F, Gulllo F, Lanciano T. Core decomposition in multilayer networks: Theory, algorithms, and applications. ACM Trans. on Knowledge Discovery from Data, 2020, 14(1): 1−40.

[39] Liu XM, Shen C, Guan XH, Zhou TD. Digger: Detect similar groups in heterogeneous social networks. ACM Trans. on Knowledge Discovery from Data, 2018, 13(1): 1−27.

[40] Barman D, Bhattacharya S, Sarkar R, Chowdhury N. *k*-context technique: A method for identifying dense subgraphs in a heterogeneous information network. IEEE Trans. on Computational Social Systems, 2019, 6(6): 1190−1205.

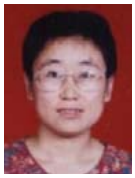
附中文参考文献:

[2] 石川, 王睿嘉, 王啸. 异质信息网络分析与应用综述. 软件学报, 2022, 33(2): 598−621. http://www.jos.org.cn/1000-9825/6357.htm [doi: 10.13328/j.cnki.jos.006357]

[29] 陈子阳, 陈伟, 贾勇, 周军锋. 基于 KST 索引的最大连通 Steiner 分量查询算法. 计算机学报, 2020, 43(7): 1216−1229.

李源(1986－), 男, 博士, 讲师, CCF 专业会员, 主要研究领域为数据挖掘, 数据库, 生物信息学.

范晓林(1995－), 女, 硕士生, CCF 学生会员, 主要研究领域为数据挖掘.



孙晶(1968－), 女, 副教授, CCF 专业会员, 主要研究领域为软件体系结构.

赵会群(1960－), 男, 博士, 教授, CCF 高级会员, 主要研究领域为软件体系结构, 大数据生成, 物联网, 云计算, 体育计算.

杨森(1998－), 男, 硕士生, 主要研究领域为数据挖掘.

王国仁(1966－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为不确定数据管理, 数据密集型计算, 可视媒体数据分析管理, 非结构化数据管理, 分布式查询处理与优化, 生物信息学.