一种大规模双网络中 *k*-连通 Truss 子图发现算法

李 源[1)] 盛 飞[2)] 孙 晶[1)] 赵宇海[3)] 王国仁[4)]

[1)](北方工业大学信息学院 北京 100144)

[2)](北京邮电大学计算机学院 北京 100876)

[3)](东北大学计算机科学与工程学院计算机科学系 沈阳 110169)

[4)](北京理工大学计算机学院 北京 100081)

摘 要 双网络由具有相同顶点集合但不同边集合的物理图和概念图构成，能够反映顶点间不同层面的交互关系. 双网络中稠密子图发现问题旨在发现物理图中连通而概念图中稠密的子图，在协作者网络分析、社区发现和疾病功能团检测等方面具有广泛应用. 但现有稠密子图模型存在以下问题:(1)基于最密集子图模型的稠密子图发现问题本质上是 NP-难的，导致精确的子图发现算法在效率上存在很大问题;(2)基于 *k*-核的模型虽然解决了效率问题，但是发现的稠密子图并不真正“稠密”. 针对以上问题，本文(1)提出了 *k*-连通 truss 子图(*k*-CT)模型. 该模型更加稠密，因此允许子图间存在重叠;(2)为了发现 *k*-连通 truss 子图，提出了一种高效的精确亚线性算法用于发现双网络中所有的 *k*-CT 子图;(3)基于 *k*-CT 子图，提出了最大连通 truss 子图(MCT)概念，对当前 *k*-CT 子图不存在任何非空(*k*+1)-CT 子图;(4)提出了自顶向下、自底向上和二分法三种不同策略的 MCT 子图发现算法. 大量基于真实和合成双网络数据的实验结果证明了本文提出算法的高效性和有效性.

关键词 双网络；稠密子图发现；*k*-连通 truss 子图模型；最大连通 truss 子图模型；*k*-类索引

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2020.01721

A *k*-Connected TrussSubgraph Discovery Algorithm in Large Scale Dual Networks

LI Yuan[1)] SHENG Fei[2)] SUN Jing[1)] ZHAO Yu-Hai[3)] WANG Guo-Ren[4)]

[1)](*School of Information Science and Technology, NorthChina University of Technology, Beijing* 100144)

[2)](*School of Computer Science, Beijing University of Posts and Telecommunications, Beijing* 100876)

[3)](*School of Computer Science and Engineering*(*Department of Computer Science*), *Northeastern University, Shenyang* 110169)

[4)](*School of Computer Science and Technology, Beijing Institute of Technology University, Beijing* 100081)

Abstract Dual network is composed of physical and conceptual graphs with the same set of vertices while different sets of edges, which reflects different interactions between vertices. Cohesive subgraph discovery in dual network, aiming to find the subgraph connected in physical graph and cohesive in conceptual graph with a wide range of applications. e.g., co-author network analysis, community discovery and disease function cluster detection. Yet, existing models mainly have the following two limitations: (1) the cohesive subgraph discovery problem based on densest subgraph model is NP-hard in essence, which leads to low efficiency of accurate discovery algorithm; (2) although the k-core based model solves the efficiency problem, the cohesive subgraphs found are not really “cohesive”. To overcome above limitations,

收稿日期：2019-09-09；在线发布日期：2020-02-07. 本课题得到国家重点研发计划项目(2018YFB1004402)、国家自然科学基金青年项目(61902004)、国家自然科学基金面上项目(61672041, 61772124, 61977001)、国家自然科学基金重点项目(61732003)、北京市教委科技项目(KM202010009009)、北方工业大学科研启动经费资助. **李 源**，博士，讲师，主要研究领域为数据挖掘、数据库、生物信息学. E-mail: liyuan@ncut.edu.cn. **盛 飞**，硕士研究生，主要研究方向为数据挖掘. **孙 晶**(通信作者)，硕士，副教授，主要研究领域为软件体系结构. E-mail:sunjing8248@163.com. **赵宇海**，博士，教授，中国计算机学会高级会员，主要研究领域为数据库、数据挖掘、生物信息. **王国仁**，博士，教授，中国计算机学会高级会员，主要研究领域为数据库.

in this paper, (1) the *k*-connected truss (*k*-CT) model is proposed, which is more cohesive and allows overlapping; (2) an efficient accurate sublinear algorithm is proposed to find all *k*-CTs; (3) the concept of maximum-connected truss (MCT) model is proposed, requiring no non-empty (*k*+1)-CTs in the dual network; (4) three algorithms based on strategies of up-down, bottom-up and binary search are proposed to find the MCTs. Extensive experiments conducted on real and synthetic large dual networks demonstrate the efficiency and effectiveness of our approaches.

Keywords dual network; cohesive subgraph discovery; *k*-connected truss model; maximum-connected truss model; *k*-class index

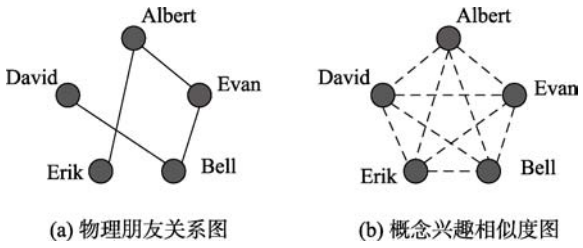
1 引 言

真实世界中，图模型经常被用来表示实体之间的关系，例如社交网络、万维网络、作者协作网络以及生物网络等[1,2]. 给定一个图，集合中的顶点表示实体，集合中的边表示实体间的关系. 在分析大规模网络时，由于网络规模具有巨大性，研究者通常聚焦于网络中很小但是更重要的区域，这对网络性质的研究更有成效也更加可行[3]. 因此，近年来，稠密子图发现逐渐成为图数据管理和分析领域的热点研究问题，并受到研究者们的广泛关注[4-10].

目前为止，许多稠密子图模型已经被提出. 例如，团（clique）或极大团[4]表示一组顶点的子集构成的一个完全子图（即子图内部任意顶点间都存在边）. 但是，团的定义过于严格，以至于许多不同松弛条件的稠密子图定义被相继提出. 其中，*n*-clique[5]把团中任意两点间的距离从 1 放松到 *n*；*k*-plex[6,7]把大小为 *c* 的团中每个节点的度数从（*c*−1）放松到（*c*−*k*）；*n*-clan[9]与 *n*-clique 类似，只是从图直径上对 *n*-clique 加以约束；类团（quasi-clique）[9,10]既可以看作从密度上也可以看作从度上对团进行放松. 但是，计算上述稠密子图都是 NP-难问题，以至于精确算法很难被扩展到规模更大的图中的稠密子图发现问题，而且当前大多数算法都仅局限于单个图中的稠密子图发现.

随着图模型的应用场景越来越复杂，单个简单图已经不能满足表达实体间复杂关系的需要. 在此背景下，双网络模型走入了研究者们的视野. 双网络包含两个具有相同顶点集，但是不同边集的图，其中一个网络表示顶点间物理层面上的交互，体现了顶点间现实中真实存在的联系，称为物理图；另一个网络表示顶点间概念层面的交互，也就是说顶点间具有相似性，但现实中可能不存在联系，该网络称为概念图. 本文聚焦于发现双网络中概念图内稠密，物理图内连通的子图. 该问题在相同研究兴趣小组发现、生物网络内疾病通路识别以及利用社交网络进行精准营销等实际场景中具有广泛应用.

例如，图 1 展示了利用双网络进行商品推荐的案例. 图 1(a)为用户的物理图，边表示用户之间真实存在的朋友关系；图 1(b)为用户的概念图，边表示用户之间购买商品兴趣相似度. 用户间的相似度可以根据用户-商品打分矩阵中，两个用户对相同商品集合的打分相关性度量而得. 值得注意的是，两个具有相似兴趣的用户可能并不存在直接的现实联系. 但是，如果一组具有很高相似性的用户，如图 1(b)所示，在物理图中具有连通性，如图 1(a)所示，那么在这组用户中间就可以进行商品推荐. 原因是该组用户中一旦有人购买了某件商品，组内其他用户很有可能对该商品同样感兴趣，且能够根据物理图中的连通性获得该商品的推荐信息.



(a) 物理朋友关系图 (b) 概念兴趣相似度图

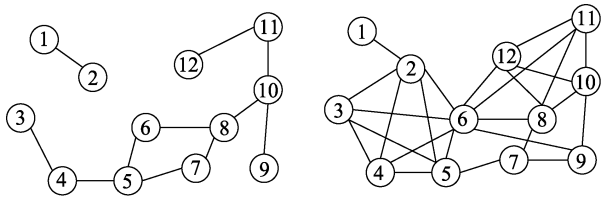
图 1 朋友关系与购买商品兴趣相似度双网络

为了发现双网络中在概念图中稠密而物理图中连通的子图，Wu 等人[11]最先提出了一种 DCS（概念图中最密集而物理图中连通）子图模型. 但是，发现 DCS 子图为 NP-难问题，即使采用近似算法，其发现结果质量不高并且计算效率较低，不能扩展到大图计算. 随后，Cui 等人[12]提出了一种 *k* 连通核（*k*-connected core, *k*-CCO）子图模型，该模型要求子图在概念图中为 *k*-核[13]，而在物理图中连通. *k* 连通核模型解决了计算效率问题，因为 *k*-核能够在 $O(n+m)$的线性时间复杂度内求出，其中 *n* 表示顶点数，*m* 表示边数，但是 *k*-核要求每个顶点至少有 *k* 个邻居的约束并不强，以至于发现的“稠密子图”

并不真的稠密. 很多稠密子图发现算法都先用 *k*-核进行过滤，正如 Seidman[14]所描述，*k*-核为稠密子图存在的温床.

针对现有模型存在的问题，本文提出一种双网络中 *k*-连通 truss（*k*-connected truss，*k*-CT）子图模型. 给定一个双网络 $G(V, E_a, E_b)$和正整数 k，*k*-CT 为双网络中的子图 g，并且满足如下三个条件：(1) 在概念图 G_b 中，g 内任意一条边至少被 *k*-2 个三角形包含；(2) 在概念图 G_b 中，g 内任意两个顶点间都存在一条三角形连通路径，即两个顶点所在的三角形由一系列的三角形连接，其中每两个相邻三角形都共享一条边；(3) 在物理图 G_a 中，g 是连通子图. 使用 *k*-CT 定义双网络中的稠密子图具有如下优点：(1) *k*-truss 不但保持了 *k*-核根据 *k* 值变化的子图间具有层次的特性，而且 *k*-truss 子图定义在结构上更加严格（基于三角形），因此也更稠密. 例如，*k*-truss 一定包含于（*k-1*）-核中，但是反之并不成立；(2) 在单一简单图中的 *k*-truss 发现算法，虽然时间复杂度稍微高于 *k*-核，但是仍然是亚线性多项式时间复杂度，同样可以扩展到大图计算中；(3) *k*-CT 的三角形连通性条件允许不同的 *k*-CT 之间存在交叠，而 *k*-核不支持子图间存在重叠.

接下来，给出 *k*-CT 子图模型在双网络中的示例. 如图 2 所示，图 2(a)为物理图 G_a，图 2(b)为概念图 G_b. 从该图中可以看出顶点集合{3, 4, 5, 6}和{6, 8, 10, 11, 12}分别构成两个 4-CT，而且顶点 6 可以同时属于两个不同的子图. 相反，根据 *k*-CCO 子图模型，顶点集合{3, 4, 5, 6, 7, 8, 9, 10, 11, 12}整体构成一个 3-CCO 子图，几乎覆盖图中所有顶点，未能有效刻画图中的稠密部分.



(a) 物理图 G_a　　(b) 概念图 G_b

图 2　双网络示例

针对 *k*-连通 truss 子图模型，本文重点解决如下两个问题. 给定双网络 G 和正整数 k，第一个问题是发现 G 中所有的 *k*-连通 truss 子图，根据 *k* 值的不同，用户可以得到不同粒度的结果；第二个问题是，在 *k* 值未知的情况下，从双网络中发现最大连通 truss（Maximum-connected truss, MCT）子图，即对于一个 *k*-CT，不存在任何包含该子图的(*k+1*)-CT. 图 2 中发现的两个 4-CT 均为 MCT 子图，因为不被任何 5-CT 包含.

本文的主要贡献点如下：

(1) 提出了双网络中 *k*-CT 子图模型，该模型既能保证发现子图的稠密性，又能保证子图发现的高效性，而且支持子图间存在重叠；

(2) 提出了高效的 KCT-FIND 算法，用于发现双网络中所有的 *k*-CT 子图；

(3) 提出了基于自顶向下、自底向上以及二分法三种不同搜索策略的算法 UB-MCT、BU-MCT 和 BIN-MCT 以及对上述算法的优化策略，从而实现高效的 MCT 子图发现；

(4) 在多个真实和合成数据集上的大量实验证明本文提出算法的高效性和有效性.

本文的剩余部分组织如下，第 2 节阐述和分析了目前单个网络和双网络中稠密子图发现相关工作；第 3 节给出相关概念和所要解决问题的定义；第 4 节详细阐述 *k*-CT 子图发现算法 KCT-FIND；在此基础上，在第 5 节中详细介绍本研究中提出的 MCT 子图发现算法 UB-MCT、BU-MCT 和 BIN- MCT；第 6 节通过在多个真实和合成数据集上的对比实验分析算法的效率和有效性；第 7 节对全文进行总结.

2　相关工作

当前单个简单图中的稠密子图发现算法大致可分为基于团的算法[2]和基于团放松的算法. 团是一种最直观并且最紧密的稠密子图模型，定义为子图中任意两个节点都相邻，图论中被称为完全子图. 发现极大团是 NP 完全问题，Eppstein D 等人[5]提出的 BK 改进算法是当前已知最高效的极大团枚举算法. 在现实应用场景中，由于团的定义过于严格，导致很难发现有效的稠密子图. 为此，诸如 *n*-clique、*n*-clan、*k*-core 等对于团约束更放松的稠密子图定义被接连提出. 接下来分别介绍基于距离、度和三角形约束放松的稠密子图定义和相应的稠密子图模型的特点.

(1) 基于距离放松的稠密子图. *n*-clique 是一个极大子图，要求在整个图中子图中任意两个顶点间的距离不大于 n[6]. 由于发现 *n*-clique 是 NP-难问题，文献[6]分别提出基于分支限界技术的精确算法和近似比为 2 的近似算法来发现所有结果. *n*-clan 为一个子图的直径不大于 n 的 *n*-clique，而 *n*-club 则为直径不大于 n 的极大子图[9].

(2) 基于度放松的稠密子图. *k*-plex 是一个子图中任意顶点在子图中最多有 *k* 个顶点不相邻的极大

子图[7, 8]，所以团可以看成是一个 1-plex. 文献[7]提出了一种 *k*-plex 枚举算法. 当 *k* 为常数时，该算法以多项式延迟时间运行. 虽然该算法相对高效，但是对于大图来讲计算效率依然很低. 为此，文献[8]提出了一种发现规模较大的 *k*-plex 算法，该算法通过将大图分解为更小的块，并且过滤掉规模较小的 *k*-plex，从而加速发现满足规模阈值的 *k*-plex. 与之相似的是，*k*-core 为一个要求子图中每个顶点至少有 *k* 个邻居的极大子图[13,15,16]. 文献[17]最先提出一种与图中边数成线性时间复杂度的核分解算法. 随后，Cheng J 等人[13]提出了一种基于外存的核分解算法，用于发现对于不同 *k* 值的 *k*-核. Khaouid W 等人[18]通过分析不同的 *k*-核分解算法，提出了能够在单机上完成的大图-核分解优化算法. 最近 Bonchi F 等人[19]将 *k*-核的定义一般化，即认为与顶点距离小于等于 *h* 则为该顶点邻居，并提出基于距离泛化的核分解算法. 类团(quasi-clique)则可以看作具有 n 个顶点以及 $\gamma\times\binom{n}{2}$ 条边[10,20].

(3) 基于三角形放松的稠密子图. *DN*-graph[21]为一个连通子图，并且要求任意连通顶点的公共邻居的下界为局部最大. *k*-truss 则是一个极大子图，并要求子图中的每一条边的两个顶点都有至少 *k*-2 个共同邻居，与该边共同形成 *k*-2 个三角形[22]. Cohen J[22]最先提出 *k*-truss 定义并给出多项式时间复杂度算法. 随后 Wang J 等人[3]提出一种改进的捆分解算法，通过对边进行排序加速算法效率. 在图中关系不确定的情况下，Huang X 等人[23]提出了不确定图上的 truss 分解算法. Zheng Z 等人[24]通过设计 KEP-索引实现了在边权重图上的高效 top-*r* 权重 *k*-truss 子图发现. 另外，文献[25]和[26]将 *k*-truss 应用到社交网络的加固中，分别通过固定某些点和边使得网络中的 *k*-truss 子图结构更加稳定.

上述稠密子图模型中，只有 *k*-core 和 *k*-truss 子图能够在多项式时间内求出. 其中核分解问题能够在线性时间复杂度内完成[17]而且高效的全局和局部搜索算法已经被提出用来发现 *k*-core 社区[27-29]；truss 分解问题能够在亚线性时间复杂度内完成[3,22]，而且已经存在关于 *k*-truss 的社区发现问题的研究[30-31]. 然而，发现其他稠密子图则都是 NP-难问题，很难应用到大规模图计算中.

近年来，随着图模型应用场景越来越复杂，双网络逐渐受到研究者的关注[11]. 双网络 $G(V,E_a,E_b)$ 包含两个具有相同顶点集，但是不同边集的图，其中一个图表示顶点间的物理交互关系，另一个图表示顶点间的概念交互关系. Wu 等人[11]提出了双网络中的 DCS 问题，即发现概念图中密集，物理图中连通的子图，但是该问题为 NP-难的. 为此作者提出两种基于不同删除顶点策略的贪婪算法来近似发现最终结果. 由于提出的近似算法没有常数的近似比，所以发现子图的精度无法保证. 随后，Cui 等人[12]提出了 *k* 连通核子图模型，并且提出了高效精确的子图发现算法. 但是 *k*-core 约束较松，所发现子图可能并不稠密且子图间不允许存在重叠.

k-truss 基于三角形划分的定义对图中边被三角形包含的数量加以约束，与 *k*-core 对图中顶点度数的约束相比更加严格. 因此，利用 *k*-truss 更能够表征图中的稠密区域，而且 *k*-truss 允许子图结构间存在重叠. 例如在合作者网络中，存在三角形结构表示作者间拥有非常紧密的合作关系，而不是简单的边连接，而且对于同一作者还可以属于多个研究兴趣小组. 针对 *k*-truss 模型，本文提出了双网络中的 *k*-连通 truss 子图模型并设计了双网络中高效的子图发现算法，可以在多项式时间内精确发现所有 *k*-连通 truss 稠密子图.

3 基本概念及问题定义

本节主要介绍一些基本概念及其符号表达，阐述了 *k*-连通 truss 子图相关定义，并对要解决的主要问题给出具体定义.

3.1 基本概念

给定一个双网络 $G(V,E_a,E_b)$，两个图 $G_a(V,E_a)$ 和 $G_b(V,E_b)$ 分别表示表示物理图和概念图，其中使用 $V=\{v_1,v_2,v_3,\ldots,v_{n-1},v_n\}$ 表示点集，$n=|V|$ 表示顶点个数. 边集 E 中每条边 $e_{i,j}$ 表示顶点 v_i 和 v_j 之间存在边. $m=|E|$ 表示边的个数. 用 $m+n$ 表示图的规模. 给定单个图 G_D 中的顶点 u，使用 $N_{G_D}(u)$ 表示 G_D 中顶点 u 的邻居集合，用 $deg_{G_D}(u)$ 表示 G_D 中顶点 u 的度数，并且 $deg_{G_D}(u)=|N_{G_D}(u)|$. 给定一个顶点集合 $S(S\subseteq V)$，使用 $G_D[S]=\{S,E(S)\}$ 表示顶点集 S 关于图 G_D 的诱导子图，对于任意的顶点 $v_i,v_j\in S$，如果 $(v_i,v_j)\in E$，那么 $(v_i,v_j)\in E(S)$.

将一个长度为 3，顶点分别用 *u, v, w* 表示的环状结构称为三角形结构，记为Δ_{uvw}，满足 $u,v,w\in V$，在三角形结构的基础上定义支持度.

定义 1 (支持度). 图 G_D 中的一条边 $e=(u,v)\in E_{G_D}$ 的支持度用 $sup(G_D,e)$表示，其定义为包含 *e* 的三角形的个数，记为 $sup(G_D,e)=|\{\Delta_{uvw}\,|\,\Delta_{uvw}\in G_D\}|$.

简而言之，边 *e* 的支持度为图 G_D 中包含边 *e*

的三角形的个数. 如果边 e 被一个三角形Δ包含，记为 $e\in\Delta$. 接下来，给出 k-truss 子图的定义.

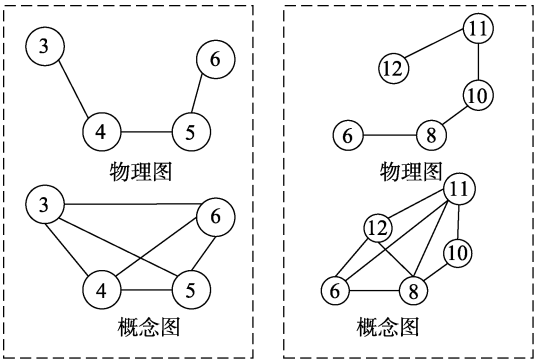


图 3　图 2 中 4-CT 子图示例

定义 2(k-truss 子图). 图 G_D 的 k-truss 子图 ($k\geqslant 2$)，记为 $T_k(G_D)$，定义为 G_D 的极大子图且满足 $\forall e\in E_{T_k}, \sup(T_k(G_D), e)\geqslant k-2$.

一条边 e 在图 G_D 中的 truss 数表示 e 所在的最大非空 k-truss 中的 k 值，记为 $\phi(e)=\max\{k\mid e\in E_{T_k(G_D)}\}$. 给定 $\phi(e)=k$，$e\in E_{T_k(G_D)}$ 但是 $e\notin E_{T_{k+1}(G_D)}$. 同时使用 $k_{max}(G_D)$表示图 G_D 任意边中最大的 truss 数. 根据 truss 数的定义能够得到边集合 k-类的定义.

定义 3 (k-类). 图 G_D 的 k-类，记为 Φ_k，定义为 $\{e\mid e\in E_{G_D}, \phi(e)=k\}$.

由于 k-truss 子图可能不连通[30]，所以还不能直接用其表示稠密子图. 接下来，给出三角形邻接和三角形连通的定义.

定义 4(三角形邻接). 给定图 G 中的两个三角形结构Δ_1 和Δ_2，如果Δ_1 和Δ_2 共享一条相同的边，则称这两个三角形邻接，表示为 $\Delta_1\cap\Delta_2\neq\varnothing$.

定义 5(三角形连通). 给定图 G 中的两个三角形 Δ_s 和Δ_t，Δ_s 和Δ_t 三角形连通，当且仅当存在一系列图 G 中的三角形$\Delta_1\ldots\Delta_n$，其中 $n\geqslant 2$，使得$\Delta_1=\Delta_s$，$\Delta_n=\Delta_t$并且对于$1\leqslant i<n$，$\Delta_i\cap\Delta_{i+1}\neq\varnothing$.

在 k-truss 子图和三角形连通定义的基础上，双网络中的 k-连通 truss（k-connected truss，k-CT）子图正式定义如下：

定义 6（k-CT 子图）. 给定一个双网络 $G(V, E_a, E_b)$和正整数 $k(k\geqslant 2)$，那么 g 称为 k-CT 子图，需要满足如下条件：

(1) g 在概念图中为 k-truss 子图；

(2) g 在概念图中满足三角连通，即 $\forall e_1, e_2\in E_b(g)$，其中 $e_1\in\Delta_1$，$e_2\in\Delta_2$，那么$\Delta_1=\Delta_2$ 或者Δ_1 与 Δ_2 在 g 内三角形连通；

(3) g 在物理图中连通；

(4) g 为满足条件(1)(2)和(3)的极大子图，也就是不存在 $g'\in G$ 使得 $g\subseteq g'$ 并且 g' 满足条件(1)(2) (3).

有时在 k 没有给定的情况下，需要发现双网络 G 中使 k 值最大的非空 k-CT 子图. 根据定义 5，给出最大连通 truss（Maximum-Connected Truss，MCT）子图的定义.

定义 7（MCT 子图）. 给定一个双网络 G，子图 g 为 MCT 子图，当且仅当 g 为一个 k-CT 子图并且 G 中不存在非空的(k+1)-CT.

定义 8（最大 CT 数）. 给定一个双网络 G，G 的最大 CT 数，记为 $kCT_{max}(G)$，是使得 k-CT 子图不为空的最大 k 值.

例如，图 3 展示了图 2 中的两个 4-CT 子图，分别有顶点集合{3, 4, 5, 6}和{6, 8, 10, 11, 12}构成，并且这两个 4-CT 子图同样为 MCT 子图，因为不存在一个非空的 5-CT 子图，所以图 2 中双网络的最大 CT 数值为 4.

3.2　问题定义

问题 1. 给定双网络$G(V, E_a, E_b)$ 和正整数$k(k\geqslant 2)$，从 G 发现所有的 k-CT 子图结构.

随着 k 值增加，结构内部顶点之间的联系愈加紧密，第二个要解决的问题就是发现双网络中的 MCT 子图.

问题 2. 给定双网络 $G(V, E_a, E_b)$，从 G 中发现所有的 MCT 子图结构.

解决上述两个问题主要存在如下两个挑战：(1) k-CT 子图结构需要同时满足物理图和概念图中的约束，比如概念图中的 k-truss 子图在物理图中并不连通，或者在物理图中连通的子图并不满足概念图中 k 值的约束；(2)在双网络中没有给定 k 值的情况下，为了求得所有 MCT 子图，需要能够较快地确定 $kCT_{max}(G)$.

4　双网络中 *k*-CT 子图发现算法

本节主要研究双网络中 k-CT 子图发现问题，给定正整数 $k(k\geqslant 2)$，从双网络中发现所有的 k-CT 子图. 通过分析传统单一网络中的 k-truss 发现算法以及双网络中其它稠密子图算法，提出了双网络下的 k-CT 子图发现算法 KCT-FIND. 该算法的主要思想是从物理图出发高效地计算双网络中 k-CT 子图. 接下来，首先分析 KCT-FIND 算法思想，然后给出算法描述，最后分析 KCT-FIND 算法的复杂性.

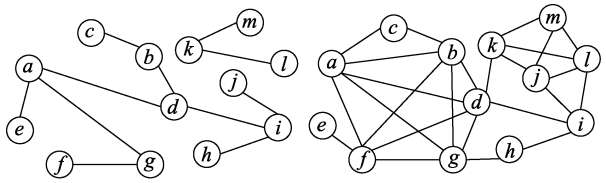
4.1　KCT-FIND算法思想

现有单网络中 k-truss 发现算法虽然能够准确查

找到需要的结果，但在双网络中，*k*-truss 结构的发现存在着困难，原因是传统单网络中的发现算法在双网络中运行存在一定的局限性.

k-CT 子图结构要求在概念图子图中要满足 *k*-truss 结构的定义；在物理图中，要求相应的顶点处于同一连通分量中. 一般的思路是从概念图入手，首先从概念图中寻找满足 *k*-truss 定义的子图，然后在物理图中判断对应的顶点是否在同一个连通分量中，若属于同一连通分量，则将该结构纳入结果集合.

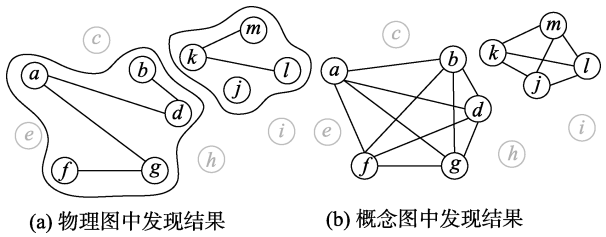
若不属于同一连通分量，在对结果精确度要求不高的情况下，可以通过补点或补边的方式使物理图中相应的顶点连通，如图 4 所示的社交双网络和 *k*=4 作为输入，查找符合 *k*-truss 结构的子图.



(a) 社交双网络的物理图 (b) 社交双网络的概念图

图 4 社交双网络

如图 5 所示，经过计算，查找出两个 4-truss 子图结构，第一个由顶点 *a*、*b*、*f*、*d*、*g* 组成的 4-truss 在物理图中连通，而由顶点 *m*、*k*、*l*、*j* 组成 4-truss 在物理图中分属两个连通分量，此时可以通过添加边(*l*, *j*)使结构满足 *k*-CT 子图的定义，但这样做会对结果的准确性产生影响，因为边(*l*, *j*)在最先输入双网络中并不存在. 若对于结果精确度要求较高，仅从概念图中发现 *k*-truss 的策略并不可行，只有在确保连通性的前提下才能继续后面概念图中的 *k*-truss 发现工作. 通过上述分析发现，需要对概念图和物理图交替进行处理.



(a) 物理图中发现结果 (b) 概念图中发现结果

图 5 社交双网络中发现结果

本节提出 KCT-FIND 算法从物理图入手：(1) 在物理图中查找所有的连通分量. (2) 在得到所有连通分量后，计算每个连通分量内的点集关于概念图的诱导子图中每条边的支持度，并初始化. (3) 遍历概念图中的边，删除支持度不满足条件的边，若有顶点因此孤立，则在物理图和概念图中删除该顶点. 值得注意的是，顶点的删除可能会造成物理图中连通分量变为不连通，也可能会影响其它边的支持度，在删除边时要及时更新邻居边的支持度，在第三步中已经完成了概念图中 *k*-truss 的发现. 对于每个概念图中的 *k*-truss 子图来说，物理图中的节点应该是连通的，由于可能出现删除顶点的情况，物理图中节点的连通性可能遭到了破坏. (4) 检查每个概念图中的 *k*-truss 子图的顶点集合在物理图中的连通性，如果连通则符合 *k*-CT 定义，若出现因删除节点造成的断裂，则可以将 *k*-truss 子图和 *k*-truss 子图顶点集合关于物理图的诱导子图作为输入，不断分解双网络. 接下来给出 KCT-FIND 具体的描述.

4.2 KCT-FIND算法描述

KCT-FIND 算法是第一个双网络中 *k*-CT 子图结构发现算法，根据 4.1 节讨论的从物理图出发，保证连通性的查找策略，本节给出 KCT-FIND 算法的具体描述，如算法 1 所示.

首先 KCT-FIND 算法对结果集合 *T* 初始化为空集（第 1 行）. 其次，该算法在物理图中寻找所有连通分量，此工作可用广度优先算法实现，这里用 *C* 表示连通分量包含的顶点集合，计算物理图中的连通分量的点集在概念图内的诱导子图 $G_b[C]$中所有边的支持度，并做初始化. 如果概念图 $G_b[C]$中所有边均满足支持度要求并且三角连通，则将 $G_a[C]$和 $G_b[C]$加入结果集合 *T* 中（第 2-4 行）. 否则，迭代移除 $G_b[C]$中未满足支持度大于 *k-2* 的边，同时更新边两端顶点的邻居节点的支持度. 若有顶点因此孤立，则将该顶点从物理图和概念图中删除（第 5 行）. 接下来，KCT-FIND 算法对概念图中的每一个三角连通分量，记为 *H*，将 $G_a[H]$和 $G_b[H]$作为新的双网络递归调用 KCT-FIND 算法，不断进行分解直到发现最终结果（第 6-7 行）. 最终，算法返回结果集合 *T*（第 8 行）.

算法 1. KCT-FIND.

输入：$G(V,E_a,E_b)$ 和正整数 *k*

输出：所有 *k*-CT 子图结构 *T*

1. $T \Leftarrow \varnothing$；
2. **FOR** 每个 G_a 中连通分量 $G_a[C]$do：
3. **IF**$G_b[C]$中所有边支持度≥*k*-2 并且 $G_b[C]$三角形连通
4. 将 $G_a[C]$和 $G_b[C]$纳入结果集 *T*；
5. **ELSE** 迭代将 $G_b[C]$中所有支持度小于 *k*-2 的边删除，

同时删除孤立顶点；

6. **FOR**$G_b[C]$中每个三角形连通分量 $G_b[H]$ do：

7. $T \Leftarrow T \cup$KCT-FIND($G_a[H]$, $G_b[H]$, k)；

8 返回 T；

以图 3 中的双网络为例，给定正整数 k=4，求出该双网络中所有 k-CT 子图结构. 使用 KCT-FIND 算法执行效果如下. 按照算法第 2-4 行计算物理图中的所有连通分量，图 4 物理图中存在两个连通分量，分别是由顶点 a、b、c、d、e、f、g、h、i、j 组成的连通分量，和顶点 k、m、l 组成的连通分量. 计算概念图中所有边的支持度，并做初始化. 初始化结果如图 6 所示.

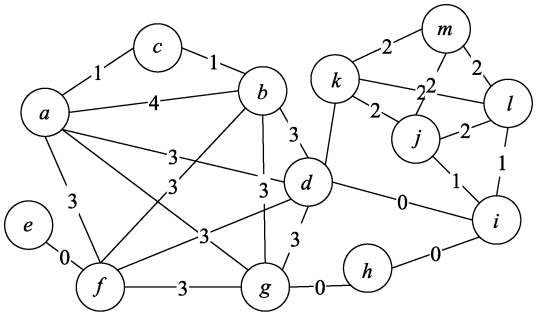
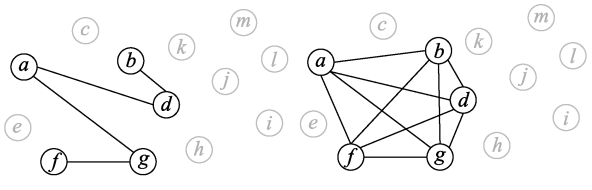


图 6　图 4 中概念图每条边包含三角形数

接下来，根据算法第 5 行，移除概念图中支持度不足 k-2 的边，在图中 5 中分别移除边(a，c)、(b，c)、(e，f)、(g，h)、(h，i)、(i，j)、(l，i)、(d，k)、(d，i)，并且在双网络中移除孤立顶点 c、e、h、i，同时更新边的支持度. 根据算法(第 6-7 行)传入概念图中的诱导子图以及在物理图中的连通分量，由顶点集合{a, b, d, f, g}诱导的双网络子图，通过验证满足 k-CT 子图结构约束，将其加入结果集 T. 由顶点集{m, k, l, j}诱导的双网络子图则在物理图中不连通，存在两个连通分量，其顶点集分别为{k, m, l}和{j}，将该双网络和 k=4 再次调用 KCT-FIND 算法. 最终结果如图 7 所示，结果集 T 中只包含一个 4-CT 子图.



(a) 结果集中的物理图　　(b) 结果集中的概念图

图 7　图 4 双网络中发现的 4-CT 子图

算法 1 的执行过程可以用一个图 8 中给出的深度优先搜索树来表示，该树的每一个节点代表一次 k-CT 调用过程，h 代表树的高度.

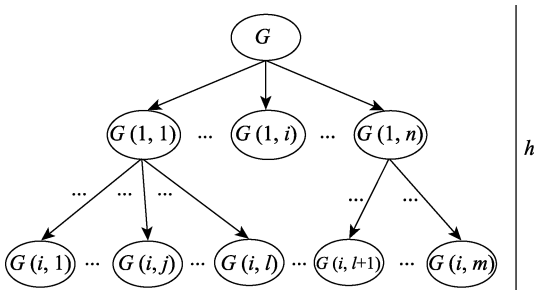


图 8　深度优先搜索树

4.3　KCT-FIND算法复杂度分析

时间复杂度：算法 1 中获取所有连通分量花费 $O(|E_a|)$的时间，检查所有边的支持度和连通性花费 $O(|E_b|^{1.5})$的时间，移除支持度不足的边和孤立顶点花费 $O(|E_b|)$的时间，在一般情况下，$|E_b|>|E_a|$，所以深度优先搜索树中的每个节点的复杂度为 $O(|E_b|^{1.5})$. 树的高度为 h 时，$\sum_{G'(V', E_a', E_b')\in G_l}|E_b'|\leqslant|E_b|$，所以算法 1 的总体时间复杂度为 $O(|E_b|^{1.5}\times h)$.

空间复杂度：算法使用 $O(|V|+|E_a|+|E_b|)$ 空间来存储双网络，$O(|E_a|+|E_b|)$ 空间存放边序列和结果集合. 因此最终空间复杂度为 $O(|V|+|E_a|+|E_b|)$.

5　双网络中 MCT 子图发现算法

上节处理的是 k-CT 子图发现问题，本节试图在双网络中发现最大连通 truss 子图，具体描述见定义 7. MCT 作为双网络中阶数最高的 k-CT 结构，拥有着最高的紧密程度，在双网络中寻找 MCT 结构，因为 k 值较大，寻找到符合条件的结构中每条边的支持度也较大，代表该边两端顶点的公共邻居也较多，同时符合条件的节点数量会随着 k 值的增大而减少，这种高密度的稠密子图是复杂网络研究的关键. 例如，在合作者双网络中寻找 MCT 结构，MCT 在概念图中是最大 k-truss 结构，代表着 MCT 子图内顶点作者之间的研究方向非常近似，同时 MCT 结构在物理图中连通，代表作者之间都有现实世界中的联系，发现 MCT，有助于挖掘出一个研究方向相似、现实之间有联系的研究小组.

令 kCT_{max} 为 MCT 中使 k-CT 结构存在的最大的 k 值，如果 kCT_{max} 已知，则可以通过 k-CT 发现算法计算获得所有的解，即可以获得双网络中的 kCT_{max}-CT 子图. 因此，主要的挑战是计算 kCT_{max}. MCT 计算问题可在多项式时间内解决，本节详细介绍双网络中最大连通 truss(MCT)子图的计算策略，并根据自底向上、自顶向下和二分查找三种计算策

略，提出了 BU-MCT、UB-MCT 和 BIN-MCT 三种 MCT 子图发现算法，并且分析了各个算法的复杂度，最后对上述 MCT 子图发现算法做进一步优化.

5.1 自底向上MCT发现算法BU-MCT

5.1.1 BU-MCT 算法思想

本小节提出一种直接的 MCT 发现算法，如算法 2 所示，第一步，将 k 初始化为 3，因为 k-truss 的定义限制 k 的最小值为 3，第二步，初始化后将目标双网络和 k=3 传入 KCT-FIND 算法，若结果集合 T 为空，说明 3-CT 在目标双网络中不存在并返回 NULL. 第三步，若集合 T 不为空则将 k 加 1，再作为参数传入 KCT-FIND 算法，直到集合 T 为空，并将 k-1 作为 kCT_{max} 返回.

MCT 发现算法中调用 KCT-FIND 算法时需要等待遍历完双网络才能开始下一次调用，而 MCT 发现过程在出现第一个 k-CT 后就可以中断. 对此本节更新了 KCT-FIND 算法，具体执行过程如过程 1 所示. 过程 1 改进了 KCT-FIND 算法，其中一旦发现存在 k-CT 子图后不再进行后续搜索，极大地节省了发现 MCT 子图算法的运行时间.

算法 2. BU-MCT.

输入：$G(V, E_a, E_b)$

输出：所有 MCT 子图结构 T

1. T 初始化为空集；
2. $T \Leftarrow$KCT-FIND-NEW(G，3)；
3. $k \Leftarrow 3$；
4. **WHILE**T 不为空集 do：
5. $k \Leftarrow k+1$；
6. $T \Leftarrow$KCT-FIND-NEW (G，k)；
7. $T \Leftarrow$KCT-FIND (G，k-1)；
8. 返回 T；

过程 1. KCT-FIND-NEW

输入：$G(V, E_a, E_b)$ 和正整数 k

输出：某个 k-CT 子图 T

1. $T \Leftarrow \varnothing$；
2. **FOR** 每个连通分量 $G_a[C]$ in $G_a$do：
3. **IF**$G_a[C]$中所有边支持度$\geqslant k$-2 并且 $G_b[C]$三角连通 do：
4. 将 $G_a[C]$和 $G_b[C]$纳入结果集 T；
5. 返回 T；
6. **ELSE** 迭代将 $G_b[C]$中所有支持度小于 k-2 的边删除，同时删除孤立顶点；
7. **FOR**$G_b[C]$中每个三角连通分量 $G_b[H]$ do：
8. $T \Leftarrow T \cup$KCT-FIND-NEW($G_a[H]$，$G_b[H]$，k)；
9. 返回 T；

5.1.2 BU-MCT 算法描述

根据 5.1.1 节的算法思想，本小节提出了完整的自底向上的 MCT 发现算法，算法完整过程如算法 2 所示. BU-MCT 算法（第 1-3 行）对结果集合 T 和 k 初始化，算法（第 4-6 行）检查当前 k 值在双网络中是否存在 k-CT 结构，若存在 k=k+1，否则跳出循环，算法（第 7 行）确认 kCT_{max}，调用 KCT-FIND（G，k-1），返回结果 T.

5.1.3 BU-MCT 算法复杂度分析

算法 2 中（第 6 行）获取 k-CT 结构最多花费 $O(|E_b|^{1.5} \times h)$的时间，外循环最多循环 kCT_{max} 次，所以时间复杂度最坏情况为 $O(|E_b|^{1.5} \times h \times k_{max})$.

影响该算法时间复杂度的主要分为两个方面，第一个方面是 k-CT 的查找，详细分析见 4.2 节，第二个方面是 kCT_{max} 的大小，因为 k 的初始值是 3，若 kCT_{max} 很大，会导致 k-CT 发现算法被多次调用. 因此，此算法适合应用于小规模数据集或者 kCT_{max} 值较小的数据集当中.

5.2 自顶向下MCT发现算法UB-MCT

5.2.1 UB-MCT 算法思想

在上一节中提出了自底向上的算法 BU-MCT，鉴于在复杂网络中，kCT_{max} 可能会很大，导致该算法的复杂度较高，本节提出了一种自顶向下的算法 UB-MCT. 该算法首先根据定理 5.1，找出 kCT_{max} 可能的最大值，该值可以通过分解双网络，找到概念图中顶点的最大核数[13]从而确定 kCT_{max}. 概念图 G_b 的最大核数用$\beta_{\max}(G_b)$表示，然后将 k 初始化为 $\beta_{\max}(G_b)$+1，接下来将目标双网络和 k 值传入 KCT-FIND-NEW 算法，如果结果集合 T 不为空，说明 $\beta_{\max}(G_b)$-CT-CT 在目标双网络中存在，kCT_{max}=$\beta_{\max}(G_b)$+1；否则，更新 k=k–1，重复调用 KCT-FIND-NEW 算法，直到结果集合 T 不为空.

定理 1. G 中每一个 k-truss 都是一个(k-1)-core 的子图.

证明：令 T 为图 G 的一个 k-truss，T 中一条边为 e=(u，v)，顶点 u 和 v 包含至少 k-2 个共同邻居，因为 e 的存在，k-truss 每个顶点度至少为 k-1，满足(k-1)-core 定义. 证毕.

算法 3. UB-MCT.

输入：$G(V, E_a, E_b)$

输出：所有 MCT 子图 T

1. $T \Leftarrow \varnothing$；
2. core-number(G_b)；//计算 G_b 中顶点的核数
3. $k = max_{u \in V}(\beta(u)) + 1$；

4. **WHILE** T 为空集并且 $k \geqslant 3$ do：
5. T⇐KCT-FIND (G，k)；
6. $k \Leftarrow k-1$；
7. 返回 T；

通过对概念图进行 k-核分解，可以获取每个顶点的核数，并且得到$\beta_{max}(G_b)$值，为计算 kCT_{max} 上界做好准备，core-number 算法的具体过程见文献[13].

5.2.2　UB-MCT 算法描述

根据 5.2.1 节的算法思想，本小节给出了自顶向下 UB-MCT 发现算法完整的执行过程，如算法 3 所示. UB-MCT 发现算法首先将结果集合 T 初始化为空集(第 1 行). 然后，通过调用 core-number 核分解算法对双网络中概念图 G_b 进行核分解，返回每个顶点的核数，并将所有顶点中最大的核数加 1 赋值给 k（第 2-3 行）. 接下来，算法在结果集合为空集的条件下，循环调用 KCT-FIND 算法，并使 $k=k-1$，直到结果集合 T 不为空跳出循环，或者 $k>3$ 时为止（第 4-6 行）. 最后 UB-MCT 算法返回结果集合 T.

5.2.3　UB-MCT 算法复杂度分析

算法 3 中，获取 k-CT 结构最多花费 $O(|E_b|^{1.5} \times h)$ 的运行时间，外循环最多循环次 $max_{u \in V}(\beta(u))$ 次，因此最坏时间复杂度为 $O(|E_b|^{1.5} \times h \times max_{u \in V}(\beta(u)))$.

影响该算法复杂度的主要分为两个方面，第一个方面是 k-CT 的查找，第二个方面是 $max_{u \in V}(\beta(u))$ 和 kCT_{max} 值之间的差距. 在两者相距较近时，时间复杂度可以逼近 $O(|E_b|^{1.5} \times h)$.

5.3　基于二分法的BIN-MCT算法

5.3.1　BIN-MCT 算法思想

前两节提出了两种不同方向的算法，鉴于在复杂网络中，kCT_{max} 的上下限已知，故能够采用二分法思想加速 MCT 子图发现过程. 本节提出了一种基于二分法的算法 BIN-MCT. 首先，根据定理 5.1，找出 kCT_{max} 可能的最大值和最小值，分别记为 max 和 min，max 初始化为 $max_{u \in V}(\beta(u))$，min 初始化 3. 然后，将 $k=min+(max-min)/2$ 和双网络 G 作为参数输入 KCT-FIND-NEW 算法中，如果结果集合 T 为不为空，则说明 k-CT 在目标双网络中存在，将 k 作为 min 和 max 作为参数输入 BIN-MCT 算法，否则将 k 作为 max 和 min 作为参数输入 BIN-MCT，直到结果 $max-min \leqslant 1$ 时，返回 max 值.

算法 4. BIN-MCT.

输入：$G(V, E_a, E_b)$

输出：所有 MCT 子图 T

1. $T \Leftarrow \varnothing$；
2. core-number(G_b)；
3. $max = max_{u \in V}(\beta(u)) + 1$, $min = 3$；
4. k=BIN-Search(G, min, max)；
5. 返回 KCT-FIND(G, k)；

过程 2. BIN-Search(G, min, max)

1. $T \Leftarrow \varnothing$；
2. $k = min + (max - min)/2$；
3. **IF** $max - min \leqslant 1$ do：
4. 返回 max；
5. $T \Leftarrow$ KCT-FIND-NEW(G, k)；
6. **IF** T 为空 do：
7. 返回　BIN-Search(G, min, k)；
8. **ELSE**　do：
9. 返回　BIN-Search(G, k, max)；

5.3.2　BIN-MCT 算法描述

算法 4 给出了基于二分查找的算法 BIN-MCT. 首先，算法对结果集合 T 进行初始化（第 1 行）. 然后，算法对双网络概念图进行核分解，找到 kCT_{max} 的上限，同时下限初始化为 3（第 2-3 行）. 接下来，调用 BIN-Search 函数求得 kCT_{max}，在得到 kCT_{max} 后调用 KCT-FIND 算法发现所有 MCT 子图（第 4-5 行）.

在 BIN-Search 函数中，如果 max-$min \leqslant 1$，则返回 max（第 1-4 行），并且将上下限的平均值作为参数传入 KCT-FIND-NEW 函数；如果结果集为空，将上下限更新为 min 和 k，否则更新为 k 和 max，并递归调用 BIN-Search 函数（第 5-9 行）.

5.3.3　BIN-MCT 算法复杂度分析

算法 4 计算 k-CT 子图最多花费 $O(|E_b|^{1.5} \times h)$ 的运行时间，而且该过程最多被调用 $\log_2 max_{u \in V}(\beta(u))$ 次，因此最坏时间复杂度为 $O(|E_b|^{1.5} \times h \times \log_2 max_{u \in V}(\beta(u)))$. 该算法复杂度主要受以下两方面影响，一是 k-CT 子图查找的时间，二是 kCT_{max} 值相对于上下限的位置.

5.4　算法优化

从前面提出的三种 MCT 子图发现算法中能够看出，BIN-MCT 算法主要从降低调用 KCT-FIND-NEW 函数的次数来提高 MCT 子图发现的效率. 但是，在多次调用函数的过程中依然会产生冗余计算. 例如，在计算是否存在 5-CT 和 6-CT 的过程中，需要重复删除仅属于 3-CT 和 4-CT 的边. 为此，本小节算法优化的主要思想是，在计算是否存在 k-CT 的过程中仅依赖于概念图中的 k-truss 子图 $T_k(G_b)$，而不是整个双网络. 定理 2 给出该思想的理论保证.

定理 2. 给定双网络 $G(V, E_a, E_b)$ 和正整数 k，那么 $V(k-\mathrm{CT}(G)) \subseteq V(T_k(G_b))$.

证明：$T_k(G_b)$表示概念图 G_b 中的 k-truss 子图，

由于 k-CT(G)同时要求在物理图中连通，约束更加严格，因此 k-CT(G)的顶点集合一定被 $T_k(G_b)$的顶点集合所包含，即$V(k\text{-CT}(G)) \subseteq V(T_k(G_b))$．证毕.

从定理 2 能够轻松得出，如果$T_k(G_b)=\varnothing$，那么 k-CT(G)必然为空集. 因此，为了找到双网络中 MCT 子图，可以从概念图中非空 k-truss 子图 $T_k(G_b)$的顶点集合在双网络的诱导子图 $G[V(T_k(G_b))]$中得到正确结果. 具体做法是，首先通过对概念图 G_b 进行 truss 分解，得到该图的 k-类（定义 3）索引. k-类索引的实质是按照每条边的 truss 值对图中边进行划分，根据已知的 k-类，可以容易得到$T_k(G_b)=\mathrm{U}_k^{k_{max}}\Phi_k$. 然后以算法 2 为例，BU-MCT 算法根据定理 2 和 k-类索引可以优化为算法 5，记为 BU-MCT-OPT 算法. 同理，可以得到 UB-MCT-OPT 和 BIN-MCT-OPT 算法.

算法 5. BU-MCT-OPT.

输入：$G(V, E_a, E_b)$

输出：所有 MCT 子图结构 T

1. T 初始化为空集；
2. 对 G_b 进行 truss 分解，求出$\{\Phi_k\}$；
3. $k \Leftarrow 3$；
4. $T_k(G_b)=\mathrm{U}_k^{k_{max}}\Phi_k$
5. $T \Leftarrow$KCT-FIND-NEW($G[V(T_k(G_b))], k$)；
6. **WHILE** T 不为空集 do：
7. $k \Leftarrow k+1$；
8. $T \Leftarrow$KCT-FIND-NEW ($G[V(T_k(G_b)-\Phi_k)], k$) ；
9. $T \Leftarrow$KCT-FIND($G[V(T_k(G_b)+\Phi_{k-1})], k-1$) ；
10. 返回 T；

下面通过一个示例解释算法 5 的运行过程，如图 9 所示. 首先图 9(a)中给出了图 4 中概念图的 k-类索引，由图可知 k_{max}=5. 首先，在$G[V(\mathrm{U}_{k=3}^{k=5}\Phi_k)]$中计算 3-CT. 如果 3-CT 存在，则继续在$G[V(\mathrm{U}_{k=4}^{k=5}\Phi_k)]$中计算 4-CT，直到发现最大的 MCT. 从图 9(b)中可见虚线框内重复计算皆可省去，因此 UB-MCT- OPT 算法相较 UB-MCT 算法能够在更小的子图空间中发现结果. 另外，值得注意的是，由于 $kCT_{max} \leqslant k_{max}$，因此不能够直接在$G[V(\Phi_{k_{\max}})]$中发现 MCT.

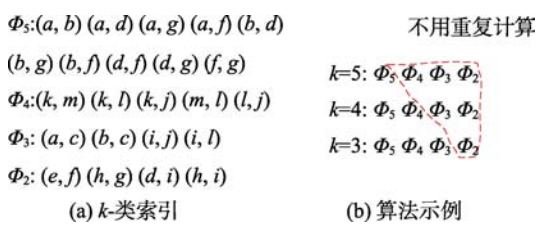


图 9 图 4 双网络 k-类索引和 BU-MCT-OPT 算法示例

6 实验结果与分析

6.1 实验环境配置

本文分别用真实数据集和合成数据集对算法进行了测试，评估的算法包括 k-CT 子图发现算法 KCT-FIND、MCT 子图发现算法 BU-MCT、UB-MCT 和 BIN-MCT 以及其优化算法 BU-MCT-OPT、UB-MCT-OPT 和 BIN-MCT-OPT. 本实验所用的软硬件环境如下：Windows 10 家庭版操作系统，Intel(R) Core(TM) i7-6700HQ 的 CPU，主频 2.6GHz，8G 内存，1T 硬盘. 开发平台为 JetBrains PyCharm 2017，开发语言为 Python 3.7.

6.2 实验数据集

实验一共采用了 8 个数据集，4 个真实数据集和 4 个合成数据集. 其中真实数据集包括：DBLP、Protein、Email 和 Facebook. 接下来介绍使用真实数据集构建的双网络，数据集统计信息如表 1 所示.

表 1 真实数据集

双网络	顶点 $\lvert V\rvert$	物理边 $\lvert E_a\rvert$	概念边 $\lvert E_b\rvert$	kCT_{max}
DBLP	20080	9342	140083	13
Protein	8341	24392	58771	10
Email	1005	16064	23544	101
Facebook	4039	88243	109053	13

(1) DBLP 双网络[12]：从 DBLP 数据集中抽取 SIGMOD、VLDB、ICDE、CIKM、EDBT、DASFAA、KDD、ICDM、SDM、WSDM 和 PAKDD 等 11 个数据库和数据挖掘领域著名国际会议近十年的发表的文章. 每篇文章可能存在多个作者，每个作者作为双网络中的一个顶点，在物理图中，将一篇文章内的所有作者两两建边，代表两者共同发表过论文，有现实联系. 概念图通过使用余弦函数度量两个作者论文关键字的相似程度，若相似度超过阈值则在概念图中两个作者之间建立边，代表两个作者有着相似的研究方向.

(2) Protein 双网络[11]：蛋白质网络就是蛋白质之间的相互作用构成的. 每个蛋白质作为双网络中的顶点. 物理图中蛋白质之间的关系来自于蛋白质交互网络，表示两个蛋白质之间真实存在的物理关系. 概念图中蛋白质之间的关系来自于高血压疾病遗传交互网络，如果两个蛋白质与高血压疾病具有显著的统计关联关系，则这两个蛋白质在概念图中存在边.

(3) Email 双网络①：该双网络是由欧洲大型研究机构的电子邮件数据形成的，顶点代表电子邮件

① http://snap.stanford.edu/data/index.html.

用户，如两位用户之间发送过至少一封邮件，则在物理图中建边. 每位用户属于 42 个部门之一，在概念图中将属于同一部门的任意两个用户之间建边.

(4) Facebook 双网络[32]：通过分析 Facebook 社交数据，双网络顶点代表用户，若两个用户相互关注，则在概念图中建边，若两个用户拥有相同的政治背景则在物理图中建边.

合成数据集使用文献[10]中的算法生成，具体使用图生成器 *GTgraph*(http://www.cse.psu.edu/~kxm85/software/GTgraph)生成物理图和概念图. 合成数据集统计信息如表 2 所示.

表 2　合成数据集

图名称	顶点 $\|V\|$	边 $\|E_a\|$，$\|E_b\|$
SYN1	2×2^{15}	1×10^6
SYN2	4×2^{15}	2×10^6
SYN3	8×2^{15}	4×10^6
SYN4	16×2^{15}	8×10^6

6.3　算法性能分析

本节主要评估本文提出的双网络中 k-CT 子图发现算法和不同 MCT 子图发现算法在真实数据集和合成数据集中的运行效率和扩展性.

6.3.1　k-CT 子图发现算法效率

图 10 展示了 k-CT 子图发现算法 KCT-FIND 在 4 个真实数据集上的运行时间. 对于每个数据集，输入 k 值分别取 $20\%\times kCT_{max}$、$40\%\times kCT_{max}$、$60\%\times kCT_{max}$、$80\%\times kCT_{max}$ 以及 kCT_{max}，其中 kCT_{max} 为数据集的最大 CT 数，如定义 8 所示. 从图 10 中可以看出随着 k 值的不断增大，KCT-FIND 算法的运行时间不断变快. 分析原因主要是因为当 k 值增大时，要求每条边被不同三角形包含的个数约束增加，从而导致满足条件边的数量大大降低，结果子图的规模也变小. 另外，由于 k 值增大引起的约束变强，KCT-FIND 算法迭代的次数减少，这也导致算法能够更快终止.

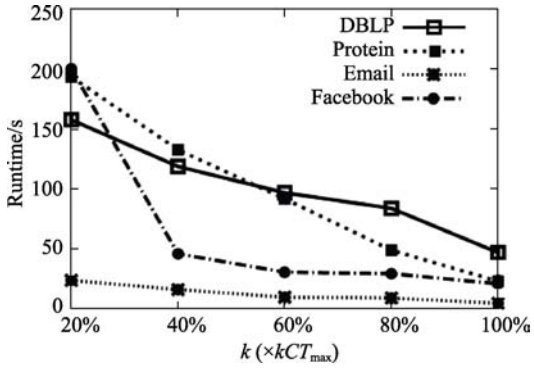


图 10　KCT-FIND 算法在不同数据集中运行时间

6.3.2　MCT 子图发现算法效率

本小节主要测试了 BU-MCT、UB-MCT 和 BIN-MCT 算法以及其对应优化算法 BU-MCT-OPT、UB-MCT-OPT 和 BIN-MCT-OPT 在发现 MCT 子图时的效率. 从图 11(a)中可以看出，BIN-MCT 算法几乎在所有的真实数据集中运行速度最快，其次是 UB-MCT 算法，运行速度最慢的是 BU-MCT 算法. 但是，这里有一个特例，在 Email 双网络中 UB-MCT 算法的运行速度要快于 BIN-MCT 算法，原因是 UB-MCT 算法采用自顶向下策略，如果 k 值上界与 kCT_{max} 值相距非常近，UB-MCT 算法会在非常短的时间内结束，在这种情况下会优于 BIN-MCT 算法. 从图 11(b)展示的优化算法运行时间同样可以观察到类似结果. 另外，从图 11(b)中还可以看出，优化后算法的运行时间要大大快于其对应的原始算法，原因是在使用了 k-类索引后，可以从当前概念图中的 k-truss 子图求出诱导子图，而不是每次都从整个双网络中搜索结果，大大降低了算法的搜索空间. 例如，BU-MCT-OPT 算法在 DBLP 数据集上比优化前算法 BU-MCT 快将近 1 个数量级.

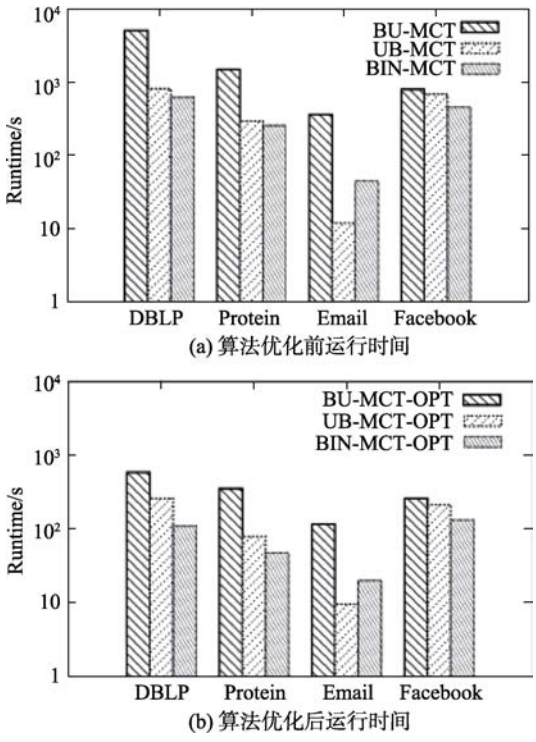


图 11　在不同真实数据集中 MCT 算法运行时间

6.3.3　MCT 子图发现算法扩展性

本小节主要评估 BU-MCT、UB-MCT 以及 BIN-MCT 算法的可扩展性. 在不同的真实双网络中，分别随机采样概念图中边 $|E_b|$ 的 20%、40%、60%、80%和 100%组成新的双网络，然后测试在不同网络上的运行时间.

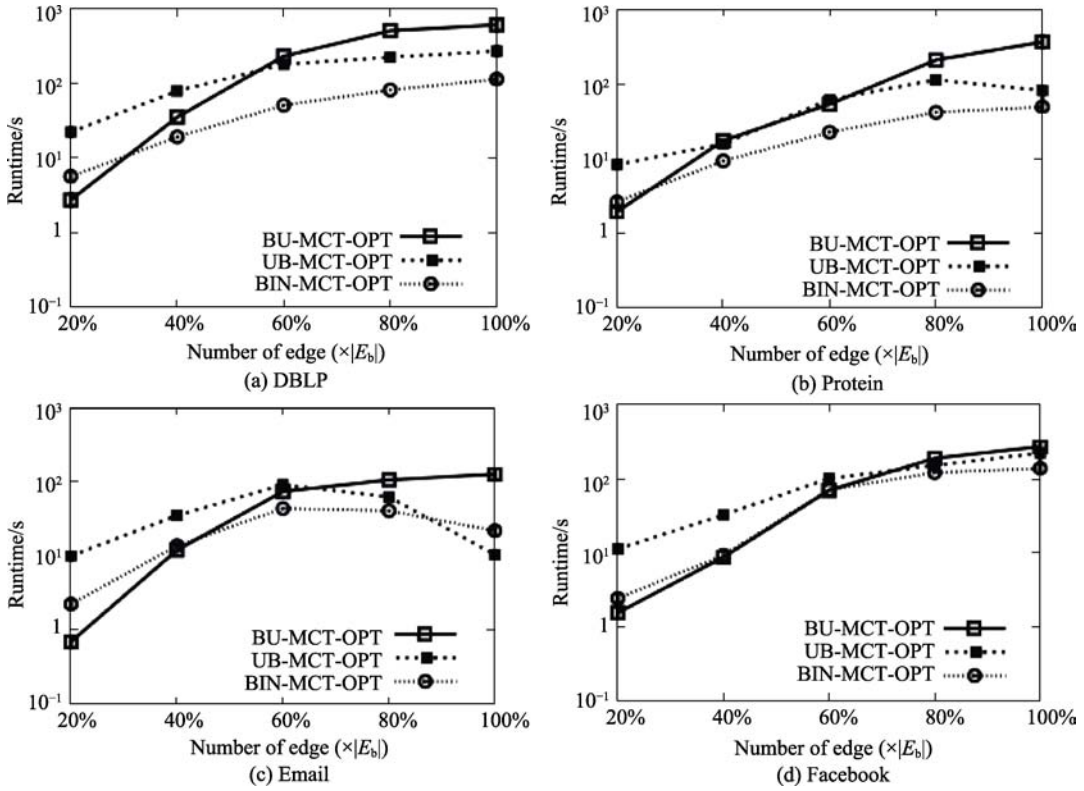


图 12 不同真实数据集上的扩展性

图 12 给出了 BU-MCT-OPT、UB-MCT-OPT 和 BIN-MCT-OPT 算法在不同双网络中的运行时间. 从结果中可以看出，当概念图中边数较少时，BU-MCT-OPT 算法运行最快，原因是这时图中 kCT_{max} 值非常小. 然后随着边数的不断增加，BU-MCT-OPT 算法的运行时间增长很快，并且逐渐超过 UB-MCT-OPT 和 BIN-MCT-OPT 算法. 当采样边数量的百分比变得更大时，BIN-MCT-OPT 算法运行速度优于其他两种算法，而且该算法随边数增加运行时间增加的速率不大，表现非常稳定，具良好可扩展性.

为了分析 BU-MCT-OPT、UB-MCT-OPT 和 BIN-MCT-OPT 算法在更大规模双网络上的可扩展性，这里采用表 2 中列出的合成数据集进行测试.

图 13 给出了在合成数据集上不同算法的运行效率. BIN-MCT-OPT 算法在不同顶点数量的双网络中的运行速度均为最快，其次是 UB-MCT-OPT 算法，但是随着顶点数量的增加，BU-MCT-OPT 算法运行时间增长的更慢，并且最终几乎接近 UB-MCT 算法. 这种现象反映了虽然网络中边的数量也在增加，但是网络实际边数与完全图中边数的比值更低，说明图变得更加稀疏，因此 kCT_{max} 值与其上界的距离更大，造成 UB-MCT 算法运行时间增长速率更大. 另外，从图中还能够看出，优化后的算法在一万秒的时间内能够处理顶点规模几十万，边规模几百万量级的双网络.

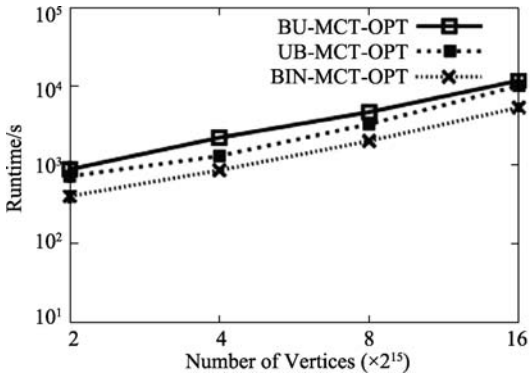


图 13 合成数据集上 MCT 算法运行时间

为了进一步分析 MCT 子图发现算法的可扩展性，本小节将 BIN-MCT-OPT 算法与文献[12]中发现最大连通核子图 MCCO 的算法 BIN-MCCO 的运行效率进行比较.

从图 14 显示的结果可以看出，BIN-MCCO 算法的运行时间要略优于 BIN-MCT-OPT 算法，这是因为发现 k_{max}-核子图的时间复杂度要低于发现 k_{max}-truss 子图的时间复杂度. 但是，BIN-MCT-OPT 算法的运行时间为亚线性算法，在大图计算中也可以接受. 另外，从下节中有效性分析可以看出 MCT 子图要比 MCCO 更加紧密，有效性也更高.

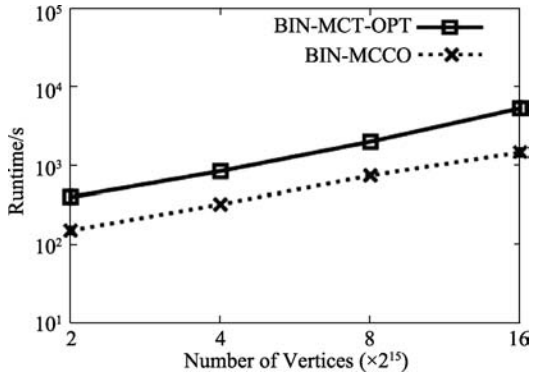


图 14　BIN-MCT-OPT 与 BIN-MCCO 运行时间比较

6.4　算法有效性分析

从 *k*-CT 子图定义可以看出使用 *k*-truss 模型定义的双网络中子图相较于 *k*-核模型定义的子图更加稠密. 本节使用经典的图密度和直径两种度量方法来比较本文提出的 *k*-CT 子图模型和 *k*-CCO 子图[12]的稠密性. 图密度定义为图中出现边的数量与图中可能存在的所有边数量的比值，而图直径的定义为图中任意两个顶点间最短距离中的最大值[32]. 给定概念图中的诱导子图 $G_b[S]$，则 $G_b[S]$ 的密度和直径分别表示为 $dens(G_b[S]) = \dfrac{2|E_b(S)|}{|S|(|S|-1)}$ 和 $diam(G_b[S]) = \max_{u,v\in S} dist(u,v)$，其中 $dist(u, v)$ 表示顶点之间的最短距离. 从上述定义可以看出子图密度越大，或是直径越小，都说明子图越稠密.

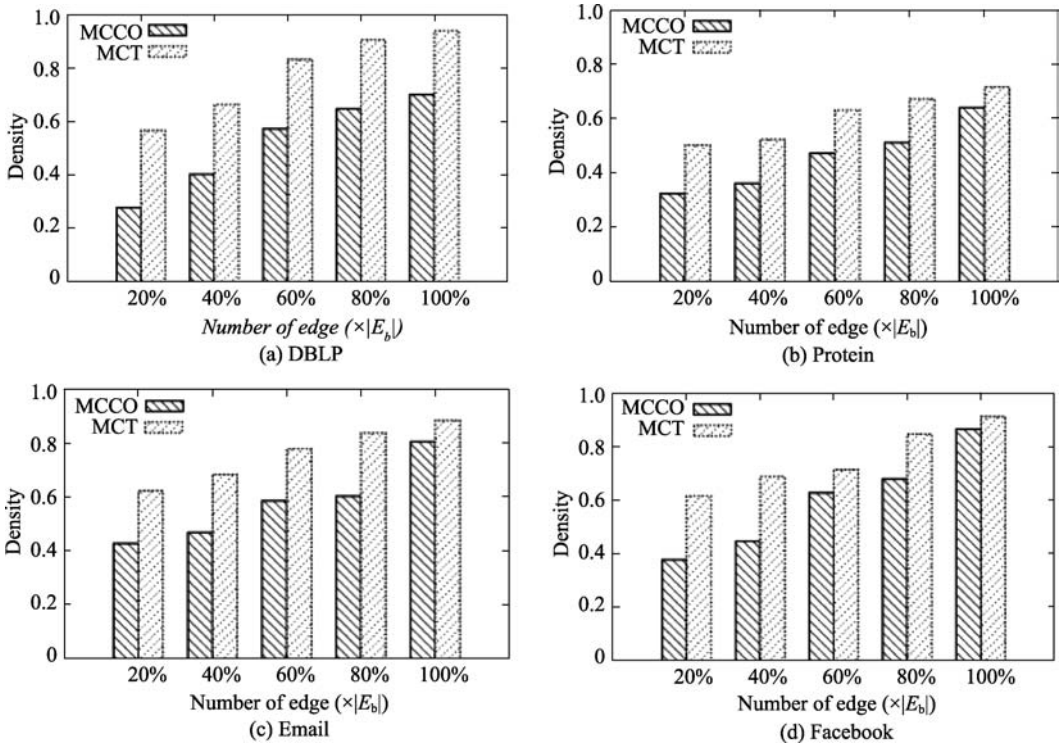


图 15　不同真实数据集上的密度

图 15 分别给出了 MCCO 子图和 MCT 子图在不同真实双网络中子图密度大小的比较. 这里将 *k*-CCO 子图中 *k* 值最大的子图记为 MCCO 子图. 在不同的真实双网络中，分别随机采样概念图中边 $|E_b|$ 的 20%、40%、60%、80%和 100%组成新的双网络，然后计算在不同网络上发现子图的密度. 从实验结果可以看出，MCT 子图在所有双网络的概念图中的密度都要大于 MCCO 子图，原因是 MCT 子图的定义要比 MCCO 子图约束要更强，因此更加稠密.

图 16 分别给出了 MCCO 子图和 MCT 子图在不同真实双网络中子图直径大小的比较. 在不同的真实双网络中，采用与密度实验相同的方法对概念图边进行采样，组成新的双网络，然后计算在不同网络上发现子图的直径. 从实验结果可以看出，从不同双网络中发现的 MCCO 和 MCT 子图，MCT 子图的直径都要小于 MCCO 子图的直径，尤其在比较稀疏的双网络中，结果更加明显. 原因是 MCT 子图要求子图中任意两条边之间都是三角形连通，因此也更加紧密. 综上可见，*k*-CT 子图具有更高的稠密程度，也证明了该子图模型的有效性.

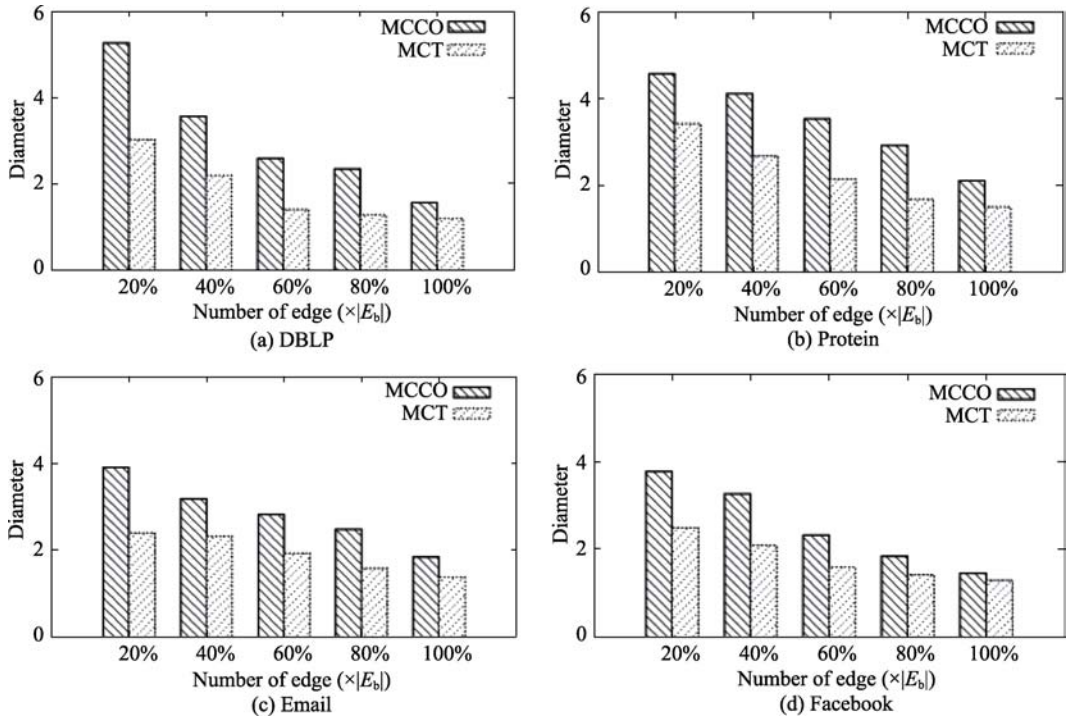


图 16 不同真实数据集上的直径

6.5 案例分析

首先,图 17 给出了 DBLP 双网络中发现的真实 *k*-CT 子图（8-CT）. 概念图中，包括美国亚利桑那州立大学 Huan Liu 教授在内的 8 名学者都是在社交媒体挖掘和机器学习领域非常知名的学者. 图中可以看出，这些学者具有相同的研究兴趣，因此在概念图中稠密；从物理图中能够看出，这 8 名学者分别来自两个研究团队，其中 Alok N. Choudhary、Ankit Agrawal 和 Wei-keng Liao 来自于美国西北大学，而 Jiliang Tang、SuhasRanganath、Suhang Wang 和 Xia Hu 都来自于 Huan Liu 教授的研究团队. 而且这两个团队可以通过 Huan Liu 教授而彼此认识. 如果需要举行一个小型的关于社交媒体挖掘的研讨会，该双网络中的学者是非常合适的.

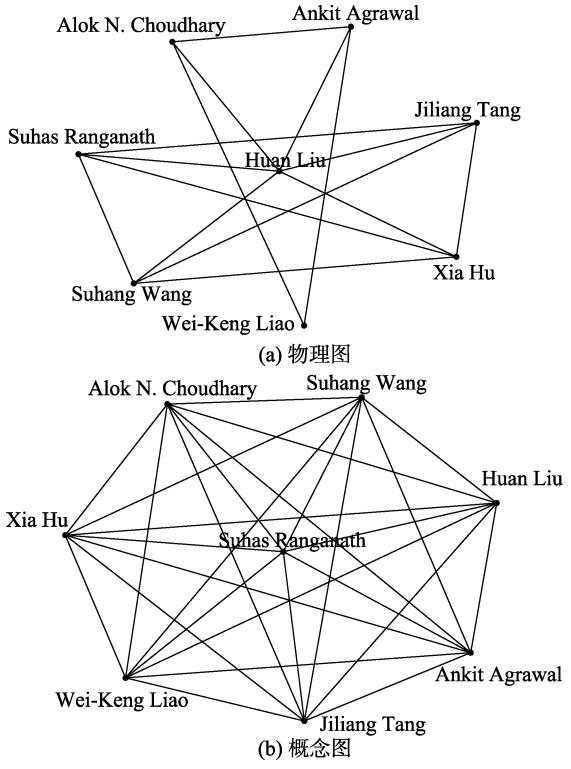
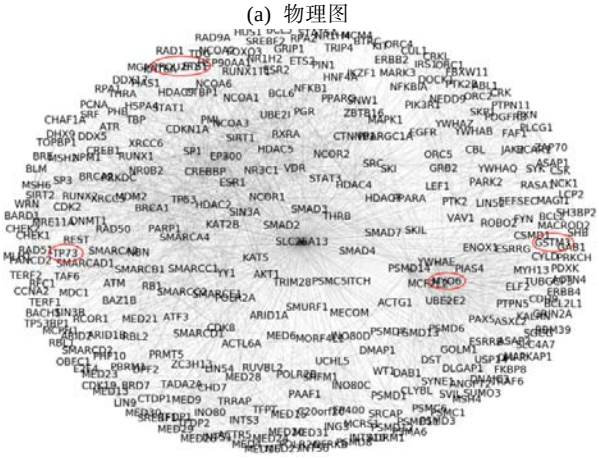


图 17 DBLP 双网络中发现的 *k*-CT 子图

接下来，图 18 给出了从蛋白质双网络中发现的 *k*-CT 子图(7-CT). 在该图中，概念图表示与高血压疾病密切相关的遗传交互网络，物理图表示两个基因在蛋白质交互网络中实际存在的关联. 在结果显示的 MCT 子图中，发现了 4 个当前已知的与高血压疾病相关的基因，分别为 MYO6、GSTM3、POU2F1 以及 TP73，在图中已圈出. 例如，MYO6 编码一种肌动蛋白的分子运动细胞，参与细胞内内小泡和细胞器的运输，显示与高血压有关[33]；Tp73 编码肿瘤蛋白 p53，它对不同的细胞压力做出反应，调节诱导细胞周期停滞、凋亡、衰老、DNA 修复和代谢改变的靶基因，在文献[34]中证明与高血压有关. 通过在蛋白质双网络中发现 *k*-CT 子图，能够有效帮助致病基因检测和疾病通路发现.

(a) 物理图



(b) 概念图

图 18　蛋白质双网络中发现的 k-CT 子图

7　总结与展望

稠密子图发现是双网络分析中的重要问题并且受到广泛关注. 针对现有稠密子图模型发现效率低和稠密度差的问题, 本文提出一种 k-连通 truss 子图模型, 该模型不仅更加稠密而且允许子图间的重叠, 同时能够在亚线性时间内发现子图. 首先, 本文提出了高效的 KCT-FIND 算法来发现指定 k 值 k-CT 子图. 接下来, 分别提出了三种不同策略的算法来发现双网络中最大连通 truss(MCT)子图, 并进行了相应优化. 最后, 大量在真实和合成数据集上的实验验证了本文提出模型和算法的高效性和有效性. 未来还需要设计更加通用的算法来满足各种稠密子图模型的发现需要.

参　考　文　献

[1] Cheng Y, Yuan Y, Chen L, et al.Event-participant and incremental planning over event-based social networks. IEEE Transactions on Knowledge and Data Engineering.doi: 10.1109/TKDE.2019.2931906

[2] Cheng Y, Chen L, Yuan Y, et al. Rule-based graph repairing: semantic and efficient repairing methods//Proceedings of the 34th IEEE International Conference on DataEngineering. Paris, France, 2018: 773-784

[3] Wang J, Cheng J. Truss decomposition in massive networks. Proceedings of the VLDB Endowment, 2012, 5(9): 812-823

[4] Pattillo J, Youssef N, Butenko S. On clique relaxation models in network analysis. European Journal of Operational Research, 2013, 226(1): 9-18

[5] Eppstein D, Löffler M, Strash D. Listing all maximal cliques in sparse graphs in near-optimal time//Proceedings of International Symposium on Algorithms and Computation. Berlin, Germany, 2010: 403-414

[6] Kargar M, An A. Keyword search in graphs: Finding r-cliques. Proceedings of the VLDB Endowment, 2011, 4(10): 681-692

[7] Berlowitz D, Cohen S, Kimelfeld B. Efficient enumeration of maximal k-plexes//Proceedings of the 2015 ACM International Conference on Management of Data. Melbourne, Australia, 2015: 431-444

[8] Conte A, Firmani D, Mordente C, et al. Fast enumeration of large k-plexes//Proceedings of the 23rd ACM International Conference on Knowledge Discovery and Data Mining. Halifax, Canada, 2017: 115-124

[9] MokkenR J. Cliques, clubs and clans. Quality & Quantity, 1979, 13(2): 161-173

[10] Zeng Z, Wang J, Zhou L, et al. Coherent closed quasi-clique discovery from large dense graph databases//Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining. Philadelphia, USA, 2006: 797-802

[11] Wu Y, Jin R, Zhu X, et al. Finding dense and connected subgraphs in dual networks//Proceedings of the 31st IEEE International Conference on Data Engineering. Seoul, South Korea, 2015: 915-926

[12] Cui L, Yue L, Wen D, et al. K-connected cores computation in large dual networks. Data Science and Engineering, 2018, 3(4): 293-306

[13] Cheng J, Ke Y, Chu S, et al. Efficient core decomposition in massive networks//Proceedings of the 27th IEEE International Conference on Data Engineering. Hannover, Germany, 2011: 51-62

[14] Seidman S B. Network structure and minimum degree. Social networks, 1983, 5(3): 269-287

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

[16] Sarıyüce A E, Gedik B, Jacques-Silva G, et al. Incremental k-core decomposition: algorithms and evaluation. The International Journal on Very Large Data Bases, 2016, 25(3): 425-447

[17] Batagelj V, Zaversnik M. An O(m) algorithm for cores decomposition of networks. arXiv preprint cs/0310049, 2003

[18] Khaouid W, Barsky M, Srinivasan V, et al. K-core decomposition of large networks on a single PC. Proceedings of the VLDB Endowment, 2015, 9(1): 13-23

[19] Bonchi F, Khan A, Severini L. Distance-generalized core decomposition//Proceedings of the 2019 ACM International Conference on Management of Data. Amsterdam, Netherlands, 2019: 1006-1023

[20] Pei J, Jiang D, Zhang A. On mining cross-graph

quasi-cliques//Proceedings of the 11th ACM International Conference on Knowledge Discovery and Data Mining. Chicago, USA, 2005: 228-238
[21] Wang N, Zhang J, Tan K L, et al. On triangulation-based dense neighborhood graph discovery. Proceedings of the VLDB Endowment, 2010, 4(2): 58-68
[22] Cohen J. Trusses: Cohesive subgraphs for social network analysis. National Security Agency Technical Report, USA, 2008, 16: 3-29
[23] Huang X, Lu W, Lakshmanan L V S. Truss decomposition of probabilistic graphs: Semantics and algorithms//Proceedings of the 2016 ACM International Conference on Management of Data. San Francisco, USA, 2016: 77-90
[24] Zheng Z, Ye F, Li R H, et al. Finding weighted k-truss communities in large networks. Information Sciences, 2017, 417: 344-360
[25] Zhang F, Zhang Y, Qin L, et al. Efficiently reinforcing social networks over user engagement and tie strength//Proceedings of the 34th IEEE International Conference on Data Engineering. Paris, France, 2018: 557-568
[26] Zhu W, Zhang M, Chen C, et al. Critical Edge Identification: A K-Truss Based Model. arXiv preprint arXiv:1906.12335, 2019
[27] Sozio M, Gionis A. The community-search problem and how to plan a successful cocktail party//Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining. Washington, USA, 2010: 939-948
[28] Barbieri N, Bonchi F, Galimberti E, et al. Efficient and effective community search. Data mining and knowledge discovery, 2015, 29(5): 1406-1433
[29] Cui W, Xiao Y, Wang H, et al. Local search of communities in large graphs//Proceedings of the 2014 ACM International Conference on Management of Data. Snowbird, USA, 2014: 991-1002
[30] Huang X, Cheng H, Qin L, et al. Querying k-truss community in large and dynamic graphs//Proceedings of the 2014 ACM International Conference on Management of Data. Snowbird, USA, 2014: 1311-1322
[31] Akbas E, Zhao P. Truss-based community search: a truss-equivalence based indexing approach. Proceedings of the VLDB Endowment, 2017, 10(11): 1298-1309
[32] Li Y, Zhao Y, Wang G, et al. Effective k-vertex connected component detection in large-scale networks//Proceedings of the 22ndInternational Conference on Database Systems for Advanced Applications. Suzhou, China, 2017: 404-421
[33] Slavin T P, Feng T, Schnell A, et al. Two-marker association tests yield new disease associations for coronary artery disease and hypertension. Human Genetics, 2011, 130(6): 725-733
[34] Watanabe Y, Yoshida M, Yamanishi K, et al. Genetic analysis of genes causing hypertension and stroke in spontaneously hypertensive rats: gene expression profiles in the kidneys. International Journal of Molecular Medicine, 2015, 36(3): 712-724

LI Yuan, Ph.D., lecture. His major research interests include data mining, database and bioinformatics.

SHENG Fei, M. S. candidate. His major research interest is data mining.

SUN Jing, master, associate professor. Her major research interest is software architecture.

ZHAO Yu-Hai, Ph.D., professor. His major research interests include database, datamining and bioinformatics.

WANG Guo-Ren, Ph.D., professor. His major research interests focus on database.

Background

Cohesive subgraph mining is one of the key research area in the field of big graph analysis. Mining cohesive subgraphs from big graph is very helpful to discover important areas in the graphs and has been widely used in many fields. However, with the application scenarios of graph models becoming more and more complex, a single graph can no longer meet the needs of expressing complex relationships among entities. Thus, dual network model has aroused the extensive interests of researchers. Dual networks are composed of concept network and physical network. The problem of discovering cohesive subgraphs in conceptual network, while keeping connected in physical network is very meaningful. However, finding the existing DCS (densest in concept graph and connected in physical graph) subgraph is NP-hard. Even if the approximation algorithm is used, the quality of the results is not high and the computation is inefficient, so it cannot be extended to the calculation of big graphs. Also, although the recent *k*-CCO (k-connected core) model solves the efficiency problem, it is not cohesive enough.

This paper proposes the *k*-connected truss (*k*-CT) model, which is more cohesive and allow overlapping. Based on *k*-CT,the concept of maximum-connected truss (MCT) is further proposed. Then, three algorithms based on different strategies are proposed to find MCTs, respectively. Experimental results verify that the proposed algorithms are both efficient and effective.

This work is supported by the National Key Research and Development Program of China (2018 YFB1004402), the National Natural Science Foundation of China (61902004, 61672041, 61772124, 61732003, 61977001), Project of Beijing Municipal Education Commission (KM202010009009) and the start-up fund of North China University of Technology.