面向有向图的k-plex稠密子图挖掘算法

侯景乐, 李振军, 代强强, 李荣华, 王国仁

引用本文

侯景乐, 李振军, 代强强, 李荣华, 王国仁. 面向有向图的k-plex稠密子图挖掘算法[J]. 计算机科学, 2026, 53(3): 166-172.

HOU Jingle, LI Zhengjun, DAI Qiangqiang, LI Ronghua, WANG Guoren. Research on Maximal Directed k-plex Enumeration Problem [J]. Computer Science, 2026, 53(3): 166-172.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.250400086

面向有向图的 k-plex 稠密子图挖掘算法

侯景乐[1]　李振军[2]　代强强[1]　李荣华[1]　王国仁[1]
1 北京理工大学计算机学院　北京 100081
2 深圳城市职业学院　广东 深圳 518038
(1779870177@qq.com)

摘　要　有向图的有向边可以表示关系的指向或数据的传递,在稠密子图挖掘中引入并拓展一些无向图的经典稠密子图模型对图挖掘工作有着重要帮助。为此,结合有向图的特点与 k-plex 的定义,称有向图中任意一个顶点的非出边邻居和非入边邻居均不超过 k 的子图结构为有向 k-plex。已有工作给出了在无向图中枚举极大 k-plex 的输出敏感算法,然而它们无法直接应用于有向图。为了解决这一问题,提出了一种基于图分解的递归枚举算法。为了更进一步优化运行效率,引入了基于支撑点的剪枝策略,还提供了基于有向 k-plex 上界的优化算法来终止一些无效的搜索分支。在真实图数据上进行实验,结果表明,图分解算法与剪枝优化均取得了良好的效果,所提算法在处理真实图数据时具有很强的实用性,能在 2 h 内完成对 KONECT 数据集中数百组真实世界有向图的处理。

关键词:稠密子图;有向 k-plex;图分解;支撑点剪枝;上界预估剪枝
中图分类号　TP391

Research on Maximal Directed k-plex Enumeration Problem

HOU Jingle[1],LI Zhengjun[2],DAI Qiangqiang[1],LI Ronghua[1] and WANG Guoren[1]
1 School of Computer Science & Technology,Beijing Institute of Technology,Beijing 100081,China
2 Information and Communication College of Shenzhen City Polytechnic,Shenzhen,Guangdong 518038,China

Abstract　The directed edge of a directed graph can represent the direction of a relationship or the transfer of data. It is of great help to introduce and expand some classical dense subgraph models of undirected graphs in dense subgraph mining. Therefore, combining the characteristics of digraphs with the definition of k-plex, the subgraph structure in which the nonoutgoing neighbors and nonincoming neighbors of any vertex in a digraph do not exceed k is called a directed k-plex. Output sensitive algorithms for enumerating maximal k-plex in undirected graphs have been proposed, but they cannot be applied directly to directed graphs. To solve this problem, a recursive enumeration algorithm based on graph decomposition is proposed for maximal directed k-plex enumeration problem. In order to further optimize the efficiency of the algorithm, a pruning strategy based on support points is introduced, and an optimization algorithm based on the upper bound of directed k-plex is provided to terminate some invalid search branches. Experimental results on real graph data show that both the graph decomposition algorithm and pruning optimization have achieved good results. The proposed algorithm has strong practicability in processing real graph data, and can complete the processing of hundreds of groups of real world digraphs in KONECT data set within 2h.

Keywords　Dense subgraph, Directed k-plex, Graph decomposition, Support point pruning, Upper bound estimated pruning

1　引言

稠密子图挖掘是图研究中常见的方法之一。现实世界的图数据通常可以被建模为有向图的形式,如社交网络[1-2]、电子邮件通信网络[3]、网站链接网络[4]等。有向图中的有向边体现了信息的方向性和流动性,使得在有向图中挖掘稠密子图能更好地解决寻找紧密社团、分析欺诈行为等问题[5-6]。

由于早期研究的局限性,现有针对稠密子图结构的研究大多仅专注于无向图,忽视了图中边的方向会丢失顶点之间的不对称关系,进而导致挖掘结果不准确[7]。近年来,陆续有研究工作开始将目光转向有向图研究。目前,k-核、k-团等稠密子图模型已经被陆续扩展至有向图,它们的出现使得从有向图中挖掘出更准确的稠密子图成为图分析中新的热点研究方向[8-10]。k-核、k-团作为限制条件相对严苛的稠密子图模型,在现实世界的图数据中会因为数据噪音、数据误差等因素无法获取到足够准确的稠密区域[11-12],当它们被拓展到有向

到稿日期:2025-04-18　返修日期:2025-09-10
基金项目:广东省普通高校特色创新类项目(2024KTSCX260)
This work was supported by the Characteristic Innovation Project of Ordinary Universities in Guangdong Province(2024KTSCX260).
通信作者:李振军(lizhenjun@szsit.edu.cn)

图中时，该问题同样存在。在无向图研究中，研究人员针对这一问题提出了松弛团模型的概念。松弛团模型往往在顶点度数、路径长度、连通性等方面做出妥协，这使得它们能更好地从现实世界的图中获取稠密子图[13]。松弛团模型的代表之一是 k-plex，因此，为了在有向图中更好地搜索稠密子图，将 k-plex 的概念拓展至有向图至关重要。

本文重点研究有向图中的有向 k-plex。由于 k-plex 模型目前还缺乏在有向图中的定义，因此首先结合无向图 k-plex 的定义将 k-plex 模型拓展到有向图中。之后，重点研究了 k-plex 模型的经典枚举问题，即极大 k-plex 枚举问题。针对极大有向 k-plex 枚举问题，本文受传统的 D2K 算法思想的启示，提出了一种基于图分解的递归搜索算法。为了进一步提升算法的运行效率，本文还为算法引入了基于支撑点的剪枝策略和基于有向 k-plex 上界的剪枝优化算法。最后，对算法处理真实世界图数据的表现进行了实验。实验结果表明，本文提出的优化算法均起到了实质性的效果，能很快处理有向图中的极大有向 k-plex 枚举问题，当 $k=2$ 且 $q=10$ 时，算法能在 1 s 内解决 KONECT 数据集中超过 40 组实例，能在 2 h 内解决 KONECT 数据集中超过 100 组实例。

2 相关工作

k-plex 模型是由 Seidman 等[14]于 1978 年提出的经典的稠密子图模型。k-plex 要求顶点集中的任意一个顶点的非邻居数不超过 k，换言之，对于一个顶点数为 n 的 k-plex，它的任意一个顶点的度都大于等于 $n-k$。

1973 年，Bron 等提出了著名的 BK 算法，它通过维护结果集、候选集和排除集 3 个顶点集合来确定当前搜索状态，还利用一种名为 pivot 的方法缩小搜索树规模，来求解 $k=1$ 时的特殊极大 k-plex，即极大团[15]。BK 算法虽然是一种极大团枚举算法，但它提出的这种递归枚举思想为后续的极大 k-plex 枚举算法提供了重要的支撑，是后续所有相关枚举算法的基础。2007 年，Wu 等基于 BK 算法提出了一种并行枚举图中所有极大 k-plex 的算法[16]，但由于 k-plex 是一种松弛团模型，因此实际得到的大多数 k-plex 规模很小，且存在不连通的现象，这显然不符合 k-plex 作为一种稠密子图模型的意义。2017 年，Conte 等提出了一种全新的枚举极大 k-plex 的算法，这种算法要求枚举出的 k-plex 顶点数必须大于指定阈值[17]。利用该约束，算法去除了大量不连通的无效极大 k-plex，有效提高了枚举效率。2018 年，Conte 等进一步将这种对 k-plex 约束大小的思想和 BK 算法结合，并给出了一种被称作 D2K 的算法[18]。D2K 算法利用图分解技术进一步提升了 k-plex 枚举的效率，还提供了一种基于 size 的剪枝方法，这些贡献使得在大规模无向图上枚举极大 k-plex 不再像过去一样困难。

从极大 k-plex 的研究历程中不难发现，在无向图中搜索极大 k-plex 已经拥有众多解决方案，但针对有向 k-plex 问题的研究却十分匮乏，现有针对有向 k-plex 的研究更多的集中于 $k=1$ 时的特殊有向 k-plex。2021 年，Conte 等首次对有向图中的团模型给出了定义，并着重探讨了如何从有向图中搜索满足强连通性的有向团[19]。他们提出了一种输出敏感算法来枚举满足强连通性的有向团，并给出了图中有向团的数量上限。2023 年，Chen 等对有向团枚举做出了更进一步的优化，他们提出了一种多分枝递归枚举算法，这种算法维护了一个有向团和一个无向团，并逐步向有向团中添加顶点来获取极大有向团，这也是目前效率最高的极大有向团枚举算法[20]。考虑到朴素的有向图 k-plex 不应当仅局限于 $k=1$ 这一种情况，定义有向 k-plex 并解决极大有向 k-plex 枚举问题就显得十分必要，这也正是本文要解决的问题。

3 相关定义

本文研究的主要对象为简单有向图。对于任意一个简单有向图 $\vec{G}=(V,\vec{E})$，V 为有向图 $\vec{G}$ 的顶点集，$\vec{E}$ 为有向图的边集。在边集 $\vec{E}$ 中，每对顶点 $\langle u,v\rangle\in\vec{E}$ 表示图中存在一条由顶点 u 指向顶点 v 的有向边，其中顶点 u 被称为顶点 v 的入边邻居，顶点 v 被称为顶点 u 的出边邻居。$N_v(\vec{G})$ 表示顶点 v 的所有邻居集合，$N_v^+(\vec{G})$ 表示顶点 v 的所有出边邻居集合，$N_v^-(\vec{G})$ 表示顶点 v 的所有入边邻居集合。p 跳邻居是 k-plex 中的一种重要概念，它代表了与顶点距离为 p 的所有顶点所组成的集合，在本文中用 $N_v^p(\vec{G})$ 表示。对应地，顶点 v 通过 p 跳能到达的顶点，本文用 $N_v^{p+}(\vec{G})$ 表示；通过 p 跳能到达顶点 v 的顶点，本文用 $N_v^{p-}(\vec{G})$ 表示。

在无向图中，k-plex 的定义是顶点集中任意一个顶点的非邻居数不超过 k 的无向子图。通过 k-plex 的定义不难发现，k-plex 是一种对邻居关系做出松弛的伪团模型，考虑到在过去稠密子图扩展研究中，Guo 等对有向伪团给出的定义是有向图中的每个顶点的出度、入度均超过一定比例的有向子图[10]，Conte 等为有向团给出的定义是满足强连通性且底图为团的有向子图[19]。本文结合它们的拓展思想与 k-plex 自身的定义，给出了有向图中有向 k-plex 的定义。

定义 1（有向 k-plex） 对于有向图 $\vec{G}=(V,\vec{E})$ 的一个子图 $\vec{H}$，若其中的任意一个顶点的非入边邻居和非出边邻居均不超过 k，则称该子图 $\vec{H}$ 为图 $\vec{G}$ 的一个有向 k-plex。

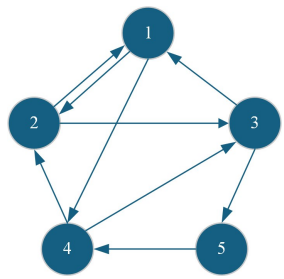

图 1 有向图

Fig. 1 Directed graph

例如，在图 1 所示的有向图中，由顶点{3,4,5}构成的子图的每个顶点有且仅有一个入邻居和一个出邻居，它的每个顶点的非入邻居和非出邻居均为 2，因此可以称{3,4,5}这个子图为图{1,2,3,4,5}的一个有向 2-plex。

同无向图 k-plex 一样，有向 k-plex 作为一种松弛模型，若不对其顶点规模加以限制，算法将花费大量的时间寻找一些规模小且不具有实际意义的非连通 k-plex。为了避免这一点，在

有向 k-plex 枚举问题中，本文也引入了特定的顶点边界。

定理 1(强连通性) 对于有向 k-plex $\vec{H}$，若 $\vec{H}$ 的顶点数 $|\vec{H}|\geqslant 2k-1$，则 $\vec{H}$ 中的任意两个顶点 u 和 v 之间存在一条从 u 到 v 的路径，且该路径长度不超过 2。

证明 对于有向 k-plex $\vec{H}$ 中任意两个顶点 u 和 v，它们之间存在两种关系，即 u 为 v 的入邻居或非入邻居。

1)若 u 为 v 的入邻居，则此时 u 到 v 之间存在一条长度为 1 的路径，定理 1 成立。

2)若 u 为 v 的非入边邻居，则根据有向 k-plex 的定义，此时 $\vec{H}$ 中最多有 k 个非顶点不是 u 的出边，k 个顶点不是 v 的入边，除去 u 和 v 自身，此时 $\vec{H}$ 内部至少存在 $|\vec{H}|-2k+2$ 个顶点既是 u 的出边，又是 v 的入边。通过这样的顶点 w，$\vec{H}$ 中存在一条从 u 到 w 再到 v 的路径，由于 $|\vec{H}|\geqslant 2k-1$，因此 $\vec{H}$ 中至少存在一个这样的顶点 w，使得 $\vec{H}$ 中任意两个顶点距离不超过 2，定理得证。

定理 1 给出的顶点边界使得有向 k-plex 能通过简单的限制顶点数就确保得到的稠密子图满足强连通性。在有向图稠密子图挖掘中，基于求解强连通分量的挖掘涉及到程序分析[21]、安全检测[22]等多个领域，结合有向 k-plex 作为一种松弛团模型的特点，满足顶点规模大于 $2k-1$ 的有向 k-plex 将能探索相关领域中更加复杂的稠密区域。结合顶点边界，本文研究的主要问题如下：

问题(极大有向 k-plex 枚举问题) 对于给定有向图，找出图中所有顶点数大于 q 的极大有向 k-plex。其中，$q\geqslant 2k-1$。

4 极大有向 k-plex 枚举

4.1 基于图分解的搜索算法

在正式设计算法之前，首先讨论有向 k-plex 的拓展性。

定理 2(继承性) 对于任意有向 k-plex $\vec{H}$，若 $\vec{S}$ 是它的子图，则 $\vec{S}$ 也是一个有向 k-plex。

证明 对于有向 k-plex $\vec{H}$，它的任意一个顶点的非出边邻居和非入边邻居均不超过 k 个。$\vec{S}$ 可以看作是由 $\vec{H}$ 删除顶点所得到的新的子图，而删除顶点并不会增加顶点的非邻居数，因此 $\vec{S}$ 也是一个有向 k-plex。

定理 2 证明了有向 k-plex 满足继承性，这意味着算法可以通过向一个已有的有向 k-plex 不断添加新顶点来尝试获得一个更大的有向 k-plex，当无法继续添加顶点时，算法就获得了一个极大有向 k-plex。对于有向图 $\vec{G}=(V,\vec{E})$，V 中的任意一个顶点 u 均是一个大小为 1 的有向 k-plex，而根据定理 1，包含顶点 u 的有向 k-plex 中的所有顶点与 u 的距离一定不超过 2，因此，对于一个包含顶点 u 的极大有向 k-plex，它一定处于子图 $u\cup N_u^+\cup N_u^-\cup N_u^{2+}\cup N_u^{2-}$ 中。图分解的算法的核心思想就是在这样一个子图中搜索极大有向 k-plex。

定理 3(图修剪) 给定一个有向 k-plex $\vec{H}$，对于 $\vec{H}$ 中的任意两个顶点 u 和顶点 v，若顶点 u 和顶点 v 不相邻，则它们的出边邻居的交集满足条件 $|N_u^+(\vec{H})\cap N_v^+(\vec{H})|\geqslant|\vec{H}|-2k+2$。

证明 当顶点 u 和 v 不相邻时，在集合 $\vec{H}-\{u,v\}$ 中，顶点 u 和 v 均至少存在 $|\vec{H}|-k$ 条出边邻居，此时，$\vec{H}-\{u,v\}$ 的大小至少为二者的出边邻居之和再减去重复部分，即 $|\vec{H}|-2\geqslant 2|\vec{H}|-2k-|N_u^+(\vec{H})\cap N_u^-(\vec{H})|$。化简可得 $N_u^+(\vec{H})\cap N_v^+(\vec{H})\geqslant|\vec{H}|-2k+2$。

表 1 图修剪关系

Table 1 Relationship of graph pruning

顶点关系	邻居	$N_v^+(\vec{H})$	$N_v^-(\vec{H})$
u、v 不相邻	$N_u^+(\vec{H})$	$\geqslant\|\vec{H}\|-2k+2$	$\geqslant\|\vec{H}\|-2k+2$
	$N_u^-(\vec{H})$	$\geqslant\|\vec{H}\|-2k+2$	$\geqslant\|\vec{H}\|-2k+2$
u、v 互为出入邻居	$N_u^+(\vec{H})$	$\geqslant\|\vec{H}\|-2k$	$\geqslant\|\vec{H}\|-2k$
	$N_u^-(\vec{H})$	$\geqslant\|\vec{H}\|-2k$	$\geqslant\|\vec{H}\|-2k$
u 是 v 的入边邻居	$N_u^+(\vec{H})$	$\geqslant\|\vec{H}\|-2k+1$	$\geqslant\|\vec{H}\|-2k$
	$N_u^-(\vec{H})$	$\geqslant\|\vec{H}\|-2k+2$	$\geqslant\|\vec{H}\|-2k+1$
u 是 v 的出边邻居	$N_u^+(\vec{H})$	$\geqslant\|\vec{H}\|-2k+1$	$\geqslant\|\vec{H}\|-2k+2$
	$N_u^-(\vec{H})$	$\geqslant\|\vec{H}\|-2k$	$\geqslant\|\vec{H}\|-2k+1$

定理 3 给出了一种图修剪的方法。由于算法需要在子图 $u\cup N_u^+\cup N_u^-\cup N_u^{2+}\cup N_u^{2-}$ 中搜索包含顶点 u 的极大有向 k-plex，因此子图 $u\cup N_u^+\cup N_u^-\cup N_u^{2+}\cup N_u^{2-}$ 中无法与顶点 u 满足定理 3 的顶点必然不在有向 k-plex 中，故可以被提前排除。定理 3 中仅给出了顶点 u 和 v 在一种情况下的邻居关系，它们的完整关系被记录在表 1 中，证明方式与定理 3 类似，此处不再赘述。

算法 1 基于图分解的搜索算法

```
输入：有向图 G=(V,E)，极大有向 k-plex 下界 q，k
输出：顶点数不小于 q 的极大有向 k-plex
1. 对底图 G 做简并排序获得顶点排序{v1,v2,…,vn}
2.    for u∈{v1,v2,…,vn} do
3.       C←{v|Nu+ ∪ Nu- ∪ Nu2+ ∪ Nu2-,v>u}
4.       X←{v|Nu+ ∪ Nu- ∪ Nu2+ ∪ Nu2-,v<u}
5.       使用表 1 将 C 和 X 中与 u 无法共存的顶点删除
6.       Enum(C,X,{u},q,k)
7.    End
8.    Enum(C,X,S,q,k)
9.    if C=∅ 且 X=∅ 且|S|≥q
10.      Output S
11.      Return
12.   for u∈C do
13.   C1←{v|v∈C 且 S∪{u,v}是一个有向 k-plex}
14.   X1←{v|v∈X 且 S∪{u,v}是一个有向 k-plex}
15.      Enum(C1,X1,S∪{u},q,k)
16.      C←C\{u},X←X∪{u}
17. End
```

利用定理 3 的图修剪，算法能在一个更小的范围内搜索极大有向 k-plex。本文采用 BK 算法的经典思想在单个子图中搜索极大有向 k-plex。在搜索过程中，算法维护 3 个经典集合来代表当前搜索状态，即结果集、候选集和排除集。对于顶点 u 和它经过修剪的子图顶点集合 V'，算法初始时的结果集合为 $\{u\}$，候选集为 V' 中还未搜索过的顶点集合，排除集为 V' 中已经搜索过的顶点集合。递归枚举算法将依次尝试将候选集中的顶点添加到结果集中，以获得一个更大的有向 k-

plex。当无法继续添加顶点时，若排除集为空，则证明当前获得的有向 k-plex 是一个极大且未被搜索过的有向 k-plex。此时，算法将输出该极大有向 k-plex，并尝试开启下一轮的搜索。

简并序作为稠密子图枚举中常见的一种排序方法，最大特点是可以使得各个顶点在排序大于它们的顶点集合中的邻居变得很少，通常远小于图的最大度。递归搜索算法指数型时间复杂度的算法，若使用简并排序来依次枚举各个顶点对应的有向 k-plex，能很好地降低算法的时间复杂度。本文在极大有向 k-plex 的枚举算法中应用了这种经典的排序算法，如算法 1 第 1 行所示。在算法 1 中，本文给出了基于图分解的递归搜索算法的具体步骤，其中递归算法的思想基于定理 2(继承性)，而算法 1 中第 3－5 行则基于定理 3(图修剪)。

4.2 基于支撑点的剪枝优化

作为一种基于图分解的递归搜索算法，算法 1 能枚举出有向图的所有极大有向 k-plex。但考虑到现实世界的图数据通常十分庞大，对算法进行剪枝优化来提高效率是十分必要的。因此，本文提出了一种基于支撑点的剪枝优化方法。

基于支撑点的剪枝优化原理是，在当前结果集为 $\vec{H}$ 的一轮递归中，若存在候选点 v 能与支撑点 p 同时加入有向 k-plex 构成一个更大的有向 k-plex，则在当前递归分支中，算法仅需枚举支撑点 p 与不满足该条件的候选点。对于被排除的候选点 v 而言，它递归产生的极大有向 k-plex 存在两种可能，即包含 p 或不包含 p。对于包含 p 的极大有向 k-plex，p 产生的递归分支可以覆盖它。对于不包含 p 的极大有向 k-plex，其中必定还包含除 p，v，$\vec{H}$ 中的其他顶点。此时，极大有向 k-plex 可以被该顶点找出。通过以上推理可以发现，利用支撑点提高算法效率的关键在于找出并排除能与支撑点共同加入当前结果集的候选点。

定理 4(支撑点)　在递归枚举过程中，若顶点 p 为支撑点，那么对于 p 的任意一个出入边邻居 v，若 $\vec{H}\backslash(N_p^+(\vec{H})\cup N_v^+(\vec{H}))$ 和 $\vec{H}\backslash(N_p^-(\vec{H})\cup N_v^-(\vec{H}))$ 均为空集，则对于当前有向 k-plex $\vec{H}$，$\vec{H}\cup\{v,p\}$ 能形成一个更大的有向 k-plex。

证明　顶点 p 和顶点 v 均在候选集中，因此它们都可以单独与 $\vec{H}$ 构成一个新的有向 k-plex。p 和 v 互为出入邻居，因此制约 p 和 v 同时加入 $\vec{H}$ 的因素在于 $\vec{H}$ 中的顶点是否满足有向 k-plex 约束。对于 $\vec{H}$ 中的任意一点 w，w 在 $\vec{H}\cup p$ 和 $\vec{H}\cup v$ 中的非出边邻居均不超过 k 个。若 p 是 w 的出边，则当 p 加入 $\vec{H}\cup v$ 中时，w 的非出边邻居均不超过 k 个。同理，若 v 是 w 的出边，则当 v 加入 $\vec{H}\cup p$ 中时，w 的非出边邻居也不超过 k 个，即 w 以 p 或 v 的任意一点为出边时，w 可以满足 p 和 v 同时加入 $\vec{H}$ 的出边邻居约束条件。因此，若条件 $\vec{H}\backslash(N_p^+(\vec{H})\cup N_v^+(\vec{H}))$ 为空满足，则 $\vec{H}$ 中的所有顶点均满足 p 和 v 同时加入 $\vec{H}$ 的出边条件，当集合 $\vec{H}\backslash(N_p^+(\vec{H})\cup N_v^+(\vec{H}))$ 和 $\vec{H}\backslash(N_p^-(\vec{H})\cup N_v^-(\vec{H}))$ 均为空集时，p 和 v 可同时加入 $\vec{H}$。

算法 2　基于支撑点的剪枝优化

输入：有向图 $\vec{G}=(V,\vec{E})$，极大有向 k-plex 下界 q，k

输出：大小超过 q 的极大有向 k-plex

1. 选取在 C 中出入边最多的顶点为支撑点 p
2. for u∈{v_1，v_2，…，v_n}do
3.　if u∈$N_p^+(C)$且 u∈$N_p^-(C)$
4.　　if $S\backslash(N_p^+(S)\cup N_u^+(S))=\emptyset$ 且 $S\backslash(N_p^-(S)\cup N_u^-(S))=\emptyset$
5.　　Continue
6. End

算法 2 给出了利用支撑点优化算法的具体过程。考虑到能被跳过的候选集顶点必定是支撑点的出入邻居，为了尽可能多地跳过候选点，本文选取在候选集中出入邻居最多的顶点作为候选点。算法 2 的步骤将取代算法 1 中第 13 行中的内容，进而提高算法效率。

4.3 基于有向 k-plex 上界的剪枝优化

在极大有向 k-plex 枚举过程中，所有有向 k-plex 都被要求必须大于一个 $q\geqslant 2k-1$ 的下界来保证得到有向 k-plex 满足强连通性。因此，若算法能提前预估当前搜索分支所能找到的最大有向 k-plex 的顶点规模，就能提前结束一些无效分支，提高算法效率。

定理 5(搜索分支对应的最大有向 k-plex)　在算法 1 递归枚举过程中，若顶点 u 加入了有向 k-plex $\vec{H}$，则当前分支能找到的最大有向 k-plex 的顶点数不超过 $|\vec{H}|+k-\ell$，其中 $\ell=\min(|\overline{N_u^+}(\vec{H})|-|\overline{N_u^+}(C)|,|\overline{N_u^-}(\vec{H})|-|\overline{N_u^-}(C)|)$。

证明　顶点 u 加入 $\vec{H}$ 后，在仅考虑顶点 u 的约束条件下，它的非出边邻居最多还能添加 $k-|\overline{N_u^+}(\vec{H})|$ 个，非入边邻居最多能添加 $k-|\overline{N_u^-}(\vec{H})|$ 个。由于添加的顶点会破坏其他顶点的 k-plex 约束，因此实际产生的有向 k-plex 只会更小，定理得证。

定理 5 阐述了一种仅关注顶点 u 的有向 k-plex 上界预估方法。在定理 5 中，算法认为 u 的出边邻居、入边邻居均能在不产生任何影响的情况下加入 $\vec{H}$，但这显然是不可能的。$\vec{H}$ 内部所有顶点还能接受最多 $\sum_{v\in\vec{H}}\overline{N_v^+}(\vec{H})$ 个非出边邻居、$\sum_{v\in\vec{H}}\overline{N_v^-}(\vec{H})$ 个非入边邻居，而 u 的任意一个邻居 w 加入 $\vec{H}$ 都将为 $\vec{H}$ 带来 $\overline{N_u^-}(\vec{H})$ 个非出边邻居和 $\overline{N_u^+}(\vec{H})$ 个非入边邻居。利用该性质，算法可以缩减加入 $\vec{H}$ 的顶点数。此外，有向 k-plex 的约束是针对 k-plex 中的每一个顶点的，因此在每次新顶点加入时，算法可以判断它的加入是否与 $\vec{H}$ 中已有顶点的出度、入度限制冲突，进而缩减加入 $\vec{H}$ 的顶点数。为了简化预测过程，在本文中算法将只利用出度或入度来预测有向 k-plex 的上界。判断使用出度或入度的标准为定理 5 中 $|\overline{N_u^+}(\vec{H})|-|\overline{N_u^+}(C)|$ 与 $|\overline{N_u^-}(\vec{H})|-|\overline{N_u^-}(C)|$ 的大小关系。算法 3 展示了仅利用出度预测有向 k-plex 上界的算法。

算法 3　基于有向 k-plex 上界的剪枝优化

输入：顶点 u，有向 k-plex $\vec{H}$，候选集 C

输出：顶点 u 加入 $\vec{H}$ 后的有向 k-plex 上界

1. for i=0 to k－1 do

```
2.    B[i]=∅
3. End
4. for v∈N_u^+(C) do
5.    B[N_v^-(H)].push(v)
6. End
7. for v∈H do
8.    s[v]=k−|N_v^+(H)|
9. s←Σ_{v∈H} k−|N_v^+(H)|
10. ub←|H|+k−|N_u^+(H)|
11. End
12. for i=0 to k−1 do
13.   if s≥i then
14.     for v∈B[i] do
15.       w←min{w|w∈N_v^-(H)}
16.       if s[w]≥0 then
17.         ub←ub+1   s←s−i
18.         s[w]←s[w]−1
19.     End
20. End
```

算法 3 首先计算了顶点 u 加入所有非出边邻居后产生的 k-plex 上限，之后尝试向 k-plex 中添加顶点 u 的出边邻居。按照这些邻居对 $\vec{H}$ 中出边邻居的影响依次将它们添加到 $\vec{H}$ 中，并着重判断它们是否破坏了 $\vec{H}$ 中某一顶点在有向 k-plex 中的约束条件。算法 3 的意义在于，在算法 1 每次进行 Enum 前，算法 3 可以运行并判断得到的 ub 是否超过 q。若 $ub<q$，则证明当前搜索分支已经无法搜索到满足条件的极大有向 k-plex，此时当前分支可以提前终止，算法效率将得到提升。

4.4 算法分析

定理 6(时间复杂度) 由算法 1—算法 3 共同构建的极大有向 k-plex 求解算法的时间复杂度为 $O(|E|+|V|\partial^2 d^2 2^{\partial d})$，其中，$\partial$ 为有向图 $\vec{G}$ 的简并序，d 为有向图 $\vec{G}$ 的最大度。

证明 算法 1 是整个有向 k-plex 求解算法的基础框架，而算法 2 和算法 3 则是对算法 1 中第 13—15 行递归构建有向 k-plex 的优化。算法 1 首先进行简并排序，该步骤可以在 $O(|E|)$ 的时间内完成，然后算法 1 将从图 $\vec{G}$ 的每个顶点出发，递归构建极大有向 k-plex。从算法 1 的第 3、第 4 行不难看出，候选集大小 $|C|$ 和排除集大小 $|X|$ 均不超过 $\partial+\partial d$。对于递归算法 Enum 而言，它的每次递归都是从候选集 C 中选择一个顶点加入结果集 S 中，因此递归搜索树共有 $2^{|C|}$ 个顶点，每个顶点都需要单独完成向结果集添加顶点后重新构建候选集、排除集的过程，该过程的时间复杂度为 $O(|C||S|)$，由于 $|S|\leqslant|C|+1$，因此每个顶点的时间复杂度为 $O(\partial^2 d^2)$，算法 1 的时间复杂度为 $O(|E|+|V|\partial^2 d^2 2^{\partial d})$。

算法 2、算法 3 是对递归搜索树中重新构建候选集、排除集过程的优化。其中，算法 2 仅需遍历结果集并判断它们与顶点 p 和 u 的关系即可，时间复杂度为 $O(|S|)$；算法 3 中统计顶点 u 的出边邻居在有向 k-plex 中的非入边邻居可以在 $O(|C|)$ 的时间内完成，而计算整个有向 k-plex 中还能接受的非出边邻居数量可以在 $O(|S|)$ 时间内完成，因此算法 3 的时间复杂度为 $O(|C|+|S|)$，显然算法 2 和算法 3 的时间复杂度均小于 $O(|C||S|)$，即它们可以在几乎不增加运算时间的前提下完成算法优化。综合以上分析，由算法 1、算法 2、算法 3 共同构建的极大有向 k-plex 求解算法的时间复杂度为 $O(|E|+|V|\partial^2 d^2 2^{\partial d})$。

定理 7(空间复杂度) 由算法 1—算法 3 共同构建的极大有向 k-plex 求解算法的空间复杂度为 $O(\partial d 2^{\partial d})$。

证明 算法的空间占用主要来源于递归搜索树中顶点的空间占用。虽然图 $\vec{G}$ 的每个顶点都能构建一颗单独的递归搜索树，但在算法运行的每个时间点，实际存在的递归搜索树有且只有一棵，搜索树的每个顶点的主要内存占用来源于三大集合：C，S，X。结合定理 6 证明中对搜索树节点以及三大集合顶点上限的计算，可得求解算法的空间复杂度为 $O(\partial d 2^{\partial d})$。

5 实验与分析

5.1 实验设置

有向 k-plex 的概念由本文首次提出并定义，在传统的算法中并没有专门用于解决极大有向 k-plex 枚举的相关算法。为了体现本文算法的特点，并对比算法的实际优化效果，本文在传统极大 k-plex 枚举算法(D2K 算法[18]和 BK-plex 算法[16])的基础上进行修改，使得它们可以用于有向 k-plex 枚举。改进后的 BK-plex 算法与 D2K 算法之间的最大差异在于图分解，改进后的 D2K 算法与本文算法之间的最大差异在于两种剪枝优化算法。实验结果很好地体现了本文提出的图分解和剪枝优化的实际效果。将第 3 章中的算法命名为 D-plex，用于与其他两种算法作区分。

在本次实验中，算法采用的有向图数据均来自于 KONECT[1)]。KONECT 是由德国科布伦茨-兰道大学收集的用于网络分析的真实世界图数据。它的覆盖范围非常广，包含了不同领域、不同大小、不同结构的多种现实世界图数据。我们测试了 KONECT 中符合测试条件的大部分图数据，并重点展示了在以下 6 组图数据上的实验结果。

表 2 实验数据集

Table 2 Experimental datasets

图数据集	顶点数	有向边数	底图简并度
Epinions	75879	508837	67
OpenFlights	2939	30501	28
Stanford	281903	2312497	71
Wikipedia-links(csb)	5561	191255	387
Wikipedia-links(gag)	2929	118603	100
TRECWT10g	1601787	8063026	140

这些图数据的顶点从数千到数百万、边从数万到数百万、简并度从数十到数百，且来自各个不同的领域。例如，Epinions 数据集来自在线社交网络，而 OpenFlights 则来自世界各地机场的航班，属于交通网络。本文选取的这 6 组图数据很

1) http://konect.cc/networks/

好地覆盖了 KONECT 数据集中大多数可实验图数据的涉及领域与数据结构。通过对比，能充分展现 D-plex 算法对现实世界图数据的分析与处理能力。

本文的所有算法均采用 C++语言实现，GCC 编译并设置 O3 的优化等级。本文实验所采用的 CPU 为 AMD Ryzen Threadripper 3990X，它具有 64 个核心数，128 线程，主频为 2.9GHZ，睿频为 4.3GHz。

5.2 实验结果

本文将 3 种算法分别运行在表 2 所列的数据集上，并在实验中设置了不同的 k 值和 q 值，以测试不同限制条件下各算法的表现。表 3 列出了 3 种算法的实际运行结果。值得一提的是，本文在测试过程中提前终止了运行时间超过10 h的测试样例，它们在表 3 中被置为 TE(Time Error)，代表超时。

表 3 实验结果

Table 3 Experiment results

data	k	q	k-plex/s	D-plex/s	D2K/s	BK-plex/s
Epinions	2	12	102952	32	123	TE
	3	12	7231655	3848	TE	TE
		20	614	31	199	TE
	4	20	99804	2712	TE	TE
Open Flights	2	12	95950	41	223	2585
		20	3578	3	28	425
	3	12	3782960	6083	TE	TE
		20	54655	409	1587	24088
Stanford	2	12	1061	19	90	32719
		20	325	10	61	30167
		30	45	3	40	27810
	3	12	11846	24	181	TE
		20	5775	12	72	29718
		30	91	4	32	29016
	4	12	60272	82	TE	TE
		20	15072	29	181	TE
		30	427	5	51	31733
Wikipedia-links (csb)	2	12	18	1.6	46	3109
		20	16	1.6	45	2996
		30	1	1.7	44	2821
	3	12	108	2.1	48	3487
		20	106	2	47	3312
		30	1	2.6	45	3187
	4	12	462	7.7	98	10897
		20	456	6.8	87	10728
		30	1	1.6	64	9615
Wikipedia-liks (gag)	2	12	34562	6.4	172	3275
		20	6265	21.7	85	4851
		30	148	10.99	44	6616
	3	12	11763105	4313	TE	TE
		20	734157	2805	TE	TE
		30	5627	450	2090	TE
TRECWT 10g	2	12	556000	1419	TE	TE
		20	176930	504	3014	TE

观察表 3 可以发现，D-plex 算法总是能在最短的时间内结束，而 D2K 算法通常比 BK-plex 算法更加高效。这证明了本文提出的有向图分解算法、剪枝优化算法在现实世界有向图中是十分有效的。观察 D2K 算法与 BK-plex 算法的运行时间，可以发现在有向图 OpenFlights 上，BK-plex 的运行时间通常不超过 D2K 算法的 20 倍，但在有向图 Stanford 上，这一数据则上升到数百倍以上。产生该结果的原因是，有向图 Stanford 的有向边远多于 OpenFlights，D2K 可以利用图分解降低有向图的复杂度，但 BK-plex 算法却缺乏这样的能力。这意味着本文给出的有向图分解更适用于大规模图。此外，对比 D-plex 算法与 D2K 算法可以发现，D-plex 算法的效率通常优于 D2K 算法数倍，但在简并度较大的图 Wikipedia-links 上，这一结果将上升到数十倍。造成该结果的原因是，本文提出的两种剪枝优化算法作用于图分解后的递归搜索树，而递归搜索树的规模与图的简并度相关，更大的候选集将为剪枝算法带来更大的优化空间。考虑到超大规模图相较于小图最大的区别在于图分解后生成的搜索树数量，以及搜索树中候选集的规模，这样的结果也意味着 D-plex 算法能很好地完成对大型图数据的分析。

为了更进一步展示 D-plex 算法处理现实世界图数据的能力，本文使用这 3 种算法依次求解 KONECT 数据集中各个有向图的极大有向 k-plex。将 KONECT 数据集中的有向图按照有向边从少到多的顺序排列，并依次处理。图 2 展示了当 $q=10$，k 值为 2，3，4 时 3 种算法在 1 s，10 s，100 s，900 s，1800 s，3600 s 时处理完成的实例数。

从图 2 可以看出，D-plex 算法在实例处理方面的效率较另外两种算法具有较大优势。随着有向图数据的复杂化，D-plex的优势变得更加明显。D-plex 算法在 $k=2$ 的情况下，能在 1 s 内在 KONECT 数据集中完成对超过 40 组小规模实例的处理，在 2 h 内能完成超过 100 组实例。这证明了 D-plex 算法能作为一种有向图的常规图挖掘手段，来为有向图挖掘稠密子图，而这也正是 D-plex 算法的价值所在。

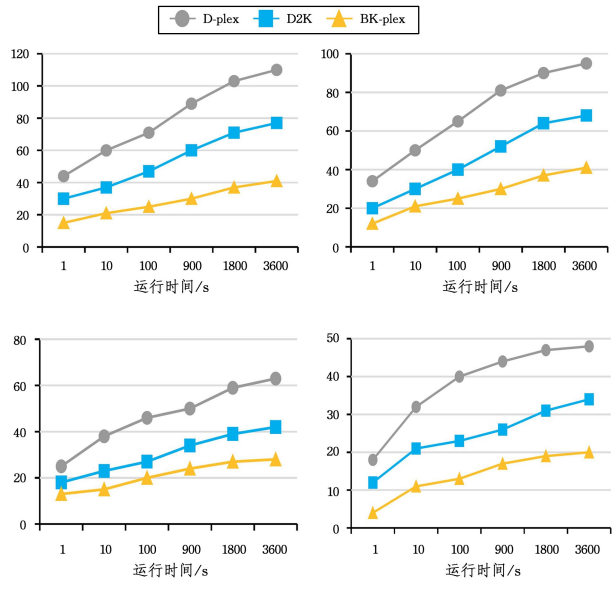

图 2 实例运行结果

Fig. 2 Instance running results

结束语 本文讨论了有向图中极大有向 k-plex 的枚举问题，并提出了一种基于图分解的递归枚举算法。首先对顶点做简并排序，之后依次枚举包含当前顶点的极大有向 k-plex。算法论证了包含当前顶点的有向 k-plex 一定处于一个与当前

顶点距离不超过 2 的有向图中，并尝试将当前有向图的顶点划分为候选集与排除集，最后通过维护候选集与排除集来递归枚举极大有向 k-plex。为了提升算法的运行效率，本文还为该递归枚举算法引入了基于支撑点的剪枝算法和基于有向 k-plex 顶点上界的剪枝优化算法，从而进一步提升了算法的运行效率。

对提出的算法在现实世界图数据上的表现进行了实验。实验结果表明，本文提出的优化算法具备实际效果，能有效解决真实世界有向图的极大有向 k-plex 枚举问题。考虑到极大有向 k-plex 枚举仅是有向稠密子图中的一种，未来的工作方向还包括最大有向 k-plex 枚举问题研究、有向伪团等其他松弛团模型的挖掘研究，以及进一步探索有向 k-plex 在真实场景中的具体应用。

参 考 文 献

[1] DU N, WU B, PEI X, et al. Community detection in large-scale social networks[C]// Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis. 2007: 16-25.

[2] XIE J, KELLEY S, SZYMANSKI B K. Overlapping community detection in networks: The state-ofthe-art and comparative study[J]. ACM Computer Surveys, 2013, 45(4): 43: 1-43: 35.

[3] TANG G, PEI J, LUK W. Email mining: tasks, common techniques, and tools[J]. Knowledge and Information Systems, 2014, 41(1): 1-31.

[4] CAPOCCI A, SERVEDIO V D, COLAIORI F, et al. Preferential attachment in the growth of social networks: The internet encyclopedia Wikipedia[J]. Physical Review E, 2006, 74(3): 036116.

[5] KLEINBERG J M. Authoritative sources in a hyperlinked environment[J]. Journal of the ACM, 1999, 46(5): 604-632.

[6] LI Z, XIONG H, LIU Y, et al. Detecting blackhole and volcano patterns in directed networks[C]// Proceedings of the 2010 IEEE International Conference on Data Mining. 2010: 294-303.

[7] ZHANG J, WANG C, WANG J. Who proposed the relationship?: recovering the hidden directions of undirected social networks[C]// Proceedings of the 23rd International Conference on World Wide Web. ACM, 2014: 807-818.

[8] GIATSIDIS C, THILIKOS D M, VAZIRGIANNIS M. D-cores: measuring collaboration of directed graphs based on degeneracy [J]. Knowledge and Information Systems, 2013, 35(2): 311-343.

[9] MA C, FANG Y, CHENG R, et al. Efficient algorithms for densest subgraph discovery on large directed graphs[C]// Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. ACM, 2020: 1051-1066.

[10] GUO G, YAN D, YUAN L, et al. Maximal directed quasi-clique mining[C]// Proceedings of the 38th IEEE International Conference on Data Engineering. IEEE Computer Society, 2022: 1900-1913.

[11] ALBA R D. A graph-theoretic definition of a sociometric clique [J]. Journal of Mathematical Sociology, 1973, 3(1): 113-126.

[12] FREEMAN L C. The sociological concept of "group": An empirical test of two models[J]. American Journal of Sociology, 1992, 98(1): 152-166.

[13] BALASUNDARAM B, BUTENKO S, HICKS I V. Clique relaxations in social network analysis: The maximum k-plex problem [J]. Operations Research, 2011, 59(1): 133-142.

[14] SEIDMAN S B, FOSTER B L. A graph-theoretic generalization of the clique concept[J]. Journal of Mathematical sociology, 1978, 6(1): 139-154.

[15] BRON C, KERBOSCH J. Algorithm 457: Finding all cliques of an undirected graph[J]. Communications of the ACM, 1973, 16(9): 575-576.

[16] WU B, PEI X. A parallel algorithm for enumerating all the maximal k-plexes[C]// Pacific-Asia Conference on Knowledge Discovery and Data Mining. 2007: 476-483.

[17] CONTE A, FIRMANI D, MORDENTE C, et al. Fast enumeration of large k-plexes[C]// Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017: 115-124.

[18] CONTE A, MATTEIS T D, SENSI D D, et al. D2K: scalable community detection in massive networks via small-diameter k-plexes[C]// Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018: 1272-1281.

[19] CONTE A, KANTÉ M M, UNO T, et al. Maximal strongly connected cliques in directed graphs: algorithms and bounds[J]. Discrete Applied Mathematics, 2021, 303: 237-252.

[20] CHEN J J, DAI Q Q, LI R H, et al. Research on directed clique enumeration algorithms that satisfy strong connectivity [J]. Computer Science and Exploration, 2024, 18(5): 1211-1222.

[21] PEARCE D J, KELLY P H J, HANKIN C. Efficient field-sensitive pointer analysis of C[J]. ACM Transactions on Programming Languages and Systems, 2007, 30(1): 4.

[22] SAITO H, TOYODA M, KITSUREGAWA M, et al. A large-scale study of link spam detection by graph algorithms[C]// Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web. ACM, 2007: 45-48.

HOU Jingle, born in 2000, master. His main research interests include graph data mining and cloud computing.

LI Zhenjun, born in 1979, Ph. D. His main research interests include deep learning, blockchain, privacy computing, etc.

（责任编辑：何杨）