ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.10, October 2010, pp.2477−2493
doi: 10.3724/SP.J.1001.2010.03831

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

从图数据库中挖掘频繁跳跃模式*

刘 勇+, 李建中, 高 宏

(哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

Mining Frequent Jump Patterns from Graph Databases

LIU Yong+, LI Jian-Zhong, GAO Hong

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

+ Corresponding author: E-mail: liuyong123456@hit.edu.cn

Liu Y, Li JZ, Gao H. Mining frequent jump patterns from graph databases. *Journal of Software*, 2010,21(10): 2477−2493. http://www.jos.org.cn/1000-9825/3831.htm

Abstract: Many algorithms on subgraph mining have been proposed. However, the number of frequent subgraphs generated by these algorithms may be too large to be effectively explored by users, especially when the support threshold is low. In this paper, a new problem of mining frequent jump patterns from graph databases is proposed. Mining frequent jump patterns can dramatically reduce the number of output graph patterns and still capture interesting graph patterns. Futhermore, jump patterns are robust against noise and dynamic changes in data. However, this problem is challenging due to the underlying complexity associated with frequent subgraph mining as well as the absence of Apriori property for jump patterns. By exploring the properties of jump patterns, two novel effective pruning techniques are proposed: Internal-Extension-Based pruning and external-extension-based pruning. Based on the proposed pruning techniques, an efficient algorithm GraphJP is presented for this new problem. It has been theoretically proven that the novel pruning techniques and the proposed algorithm are correct. Extensive experimental results demonstrate that the novel pruning techniques are effective in pruning the unpromising parts of search space, and GraphJP is efficient and scalable in mining frequent jump patterns.

Key words: data mining; graph mining; graph database; frequent subgraph; jump pattern

摘 要: 很多频繁子图挖掘算法已被提出.然而,这些算法产生的频繁子图数量太多而不能被用户有效地利用.为此,提出了一个新的研究问题:挖掘图数据库中的频繁跳跃模式.挖掘频繁跳跃模式既可以大幅度地减少输出模式的数量,又能使有意义的图模式保留在挖掘结果中.此外,跳跃模式还具有抗噪声干扰能力强等优点.然而,由于跳跃模式不具有反单调性质,挖掘它们非常具有挑战性.通过研究跳跃模式自身的特性,提出了两种新的裁剪技术:基于内扩展的裁剪和基于外扩展的裁剪.在此基础上又给出了一种高效的挖掘算法 GraphJP(an algorithm for mining jump patterns from graph databases).另外,还严格证明了裁剪技术和算法 GraphJP 的正确性.实验结果表明,所提出的裁剪技术能够有效地裁剪图模式搜索空间,算法 GraphJP 是高效、可扩展的.

* Supported by the National Natural Science Foundation of China under Grant Nos.60773063, 60903017 (国家自然科学基金); the National Basic Research Program of China under Grant No.2006CB303000 (国家重点基础研究发展计划(973)); the NSFC/RGC Joint Research Scheme under Grant No.60831160525 (NSFC/RGC 联合资助项目)

Received 2009-05-31; Revised 2009-10-09; Accepted 2010-01-21

关键词: 数据挖掘;图挖掘;图数据库;频繁子图;跳跃模式
中图法分类号: TP311 文献标识码: A

作为一种通用的数据结构,图可以用来表示数据对象之间的各种复杂关系.例如,图可以表示化合物的分子结构、蛋白质交互网络、社会网络等.随着科学与工程领域图数据的大量出现,从图数据库中发现有用的知识已成为数据挖掘领域一项重要的研究课题.目前,很多高效的频繁子图挖掘算法[1−6]已被提出.然而,在大多数情况下,这些挖掘算法将产生大量的甚至指数级的频繁子图.例如,文献[7]指出,在一个常用的化合物集合 CA 上,当相对支持度为 5%时,将输出接近 100 万个频繁子结构.大量的输出结果一方面使得对结果的分析和利用极其困难,另一方面也极大地降低了挖掘算法的效率.

为了解决这种输出图模式数量爆炸问题,已经提出了两类方法:(1) 挖掘闭频繁图模式[7];(2) 挖掘极大频繁图模式[8,9].如果频繁图模式 p 的所有真超图的支持度都与 p 不相等,则称 p 为闭频繁图模式.如果频繁图模式 p 的所有真超图都不是频繁的,则称 p 为极大频繁图模式.第 1 种方法给出的定义过于严格,使得很多频繁图模式不能被过滤掉,因此,闭频繁图模式的数量仍然很大;第 2 种方法将丢失大量图模式的支持度信息,而且使得很多有意义的图模式不能被产生.

为了克服现有方法的缺点,本文研究了图模式与其超图之间的确切关系,提出了一种新的意义度量,称为Δ-跳跃模式.如果在数据库 D 上,图模式 p 的支持度比 p 的任何真超图的支持度都大于一个整数Δ,则 p 被称为 D 上的Δ-跳跃模式.

跳跃模式具有很多重要的特性,这些特性使得挖掘它们具有重要的实际意义.首先,跳跃模式是稳定的(见第 2.2 节),它们对躁声和数据的变化不敏感.这一特性非常重要,因为在实际应用中躁声是普遍存在的,跳跃模式的这种抗干扰能力使得它们能够更广泛地被应用;其次,挖掘跳跃模式能够极大地减少输出结果的数量(见后文第 4.2 节),这使得后续的分析工作量被大大简化,提高了挖掘结果的可用性;最后,挖掘跳跃模式仍然能使有意义的图模式保留在挖掘结果中.下面的例子展示了在实际应用中跳跃模式的意义.

美国癌症研究所(NCI)给出了对 HIV 病毒具有强抵抗作用的一个化合物集合 CA.CA 中含有 422 个化合物,其中的化合物被分成如下几个化学类别[10]:Azido Pyrimidines,Dyes Polyanions,Pyrimidine Nucleosides,Heavy Metal Compounds,Purine Nucleosides.我们发现,每类中的核心子结构(最大公共子结构)都具有这样一个特性:如果核心子结构被扩展成任何一个多一条的新的子结构,则新的子结构的支持度就会快速地下降.例如,图 1 显示了 Azido Pyrimidines 类的核心子结构,它在 CA 中的支持度是 64.该子结构的任何真超图在 CA 中的支持度都小于等于 51.实际上,每类中的核心子结构在化学上都代表一个重要的功能团,各种新的化合物主要以这些功能团为基础来创建.某个功能团 G 在一些化合物中可能被扩展成 G_1,而在另一些化合物中又可能被扩展成 G_2.因此,这些功能团的任何超结构的支持度都会明显地减少.显然,我们的跳跃模式定义能够很好地刻画这一特性.

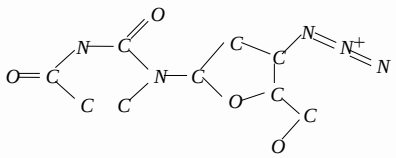


Fig.1 Core substructure in Azido Pyrimidines
图 1 在 Azido Pyrimidines 类的核心子结构

本文研究如何高效地从图数据库中挖掘频繁跳跃模式.各种图挖掘算法主要利用支持度的反单调性质(Apriori 性质)对搜索空间进行裁剪.然而,跳跃模式不具有这样的性质,这使得挖掘跳跃模式非常具有挑战性.通过仔细研究跳跃模式自身的特性,我们提出了两种新的裁剪技术——基于内扩展的裁剪和基于外扩展的裁剪.通过将这两种新的裁剪技术集成到著名的 DFS(depth-first search)编码枚举框架中[3],我们设计了一种高效的挖掘算法 GraphJP(an algorithm for mining jump patterns from graph databases).在理论上,我们严格地证明了这两种裁剪技术的正确性以及算法 GraphJP 的正确性.

大量真实和合成数据上的实验结果表明,这两种新的裁剪技术能够有效地裁剪图模式搜索空间,算法 GraphJP 能够高效、可扩展地挖掘频繁跳跃模式.此外,一个重要的发现是,当Δ值稍微大于 0 时(例如Δ=2),频繁

Δ-跳跃模式的数量就比闭频繁图模式的数量少很多.

本文第 1 节介绍预备知识.第 2 节给出问题定义,研究跳跃模式的性质.第 3 节介绍新的裁剪技术,给出完整的算法,证明裁剪技术和算法的正确性.第 4 节通过实验证明裁剪技术和挖掘算法的有效性.第 5 节介绍相关工作.第 6 节总结全文.

1 预备知识

本文主要考虑连通的无向标号简单图.通过简单修改,本文提出的方法也适用于有向图、无标号图和不连通图.若无特别说明,本文中的图均指连通的无向标号图.一个图 G 定义为一个四元组 $G=(V,E,\Sigma,l)$,其中:V 是顶点集合;$E\subseteq V\times V$ 是边集合;Σ是标号集合;$l:V\cup E\to\Sigma$是一个函数,用来对顶点和边分配标号.

定义 1.1(子图同构). 给定两个图 $G=(V,E,\Sigma,l)$和 $G'=(V',E',\Sigma',l')$,一个从 G 到 G'的子图同构是一个单射函数 $f:V\to V'$,满足:(1) $\forall u\in V,l(u)=l'(f(u))$;(2) $\forall(u,v)\in E,(f(u),f(v))\in E'$并且 $l((u,v))=l'((f(u),f(v)))$.单射函数 f 也称为 G 在 G'中的一个嵌入.

如果存在从 G 到 G'的子图同构,称 G 是 G'的子图,G'是 G 的超图,记为 $G\subseteq G'$.如果 $G\subseteq G'$且 $G\neq G'$,称 G 是 G'的真子图,G'是 G 的真超图,记为 $G\subset G'$.子图的同构测试已被证明是一个 NP-完全问题.如果 $G\subseteq G'$,也称 G'包含 G.

给定图集合 $D=\{G_1,G_2,\ldots,G_n\}$和图模式 P,P 在 D 中的支持集定义为 D 中包含 P 的图集合,记为 $D_{supp}(P)=\{G_i|P\subseteq G_i,G_i\in D\}$.$|D_{supp}(P)|$称为 P 在 D 中的支持度,记为 $supp(P;D)$.$|D_{supp}(P)|/|D|$称为 P 在 D 中的相对支持度.支持度具有反单调性质:如果 $P_1\subseteq P_2$,则 $supp(P_1;D)\geqslant supp(P_2;D)$.对于用户给定的一个最小支持度阈值 min_$sup$,如果 $supp(P;D)\geqslant$min_sup,则称 P 在 D 中是频繁的.D 中所有频繁图模式集合记为 $FS=\{P|supp(P;D)\geqslant$min_$sup\}$.如果在 D 中 P 的任何真超图与 P 支持度都不相同,则称 P 是 D 中的闭图模式.D 中所有闭频繁图模式集合记为 $CS=\{P|P\in FS,\neg\exists P'\in FS$ 使得 $P\subset P'$并且 $supp(P;D)=supp(P';D)\}$.在下文中,有时使用 $supp(P)$表示 $supp(P;D)$.

2 问题定义

2.1 跳跃模式定义

本质上,闭图模式是根据图模式与其超图之间的某种特殊关系来定义的.下面,我们扩展闭图模式的定义以更精确地刻画图模式与其超图之间的关系.

定义 2.1(绝对跳跃值). 设 P 是数据库 D 中的一个图模式.P 在 D 中的绝对跳跃值定义为

$$JV_{abs}(P;D)=\min\{supp(P;D)-supp(P';D)|\forall P',P\subset P'\}.$$

定义 2.2(绝对跳跃模式). 设 P 是数据库 D 中的一个图模式.给定一个整数 $\Delta(\Delta\geqslant 0)$,如果 $JV_{abs}(P;D)>\Delta$,则称 P 是 D 中的 Δ-绝对跳跃模式.

定义 2.3(相对跳跃值). 设 P 是数据库 D 中的一个图模式.P 在 D 中的相对跳跃值定义为

$$JV_{rel}(P;D)=\min\left\{\frac{supp(P;D)-supp(P';D)}{supp(P;D)}\middle|\forall P',P\subset P'\right\}.$$

定义 2.4(相对跳跃模式). 设 P 是数据库 D 中的一个图模式.给定一个小数 $\delta(0\leqslant\delta\leqslant 1)$,如果 $JV_{rel}(P;D)>\delta$,则称 P 是 D 中的 δ-相对跳跃模式.

直觉上,图模式 P 在数据库 D 中的绝对(相对)跳跃值是 P 与 P 的真超图之间最小的绝对(相对)支持度差异.

例 2.1:图 2 显示了一个图数据库 D,图 3 显示了 D 中图模式的一个子集.显然,P_1 在 D 中的支持度是 4,P_1 的任何超图在 D 中的支持度都小于等于 3.因此,P_1 在 D 中的绝对跳跃值 $JV_{abs}(P_1;D)=1$,P_1 在 D 中的相对跳跃值 $JV_{rel}(P_1;D)=1/4$.类似地,可以得到 $JV_{abs}(P_2;D)=2$,$JV_{rel}(P_2;D)=2/3$,$JV_{abs}(P_3;D)=0$,$JV_{rel}(P_3;D)=0$.如果 $\Delta=1$,则图 3 显示的图模式集合中只有 P_2 是 D 中的 Δ-绝对跳跃模式.

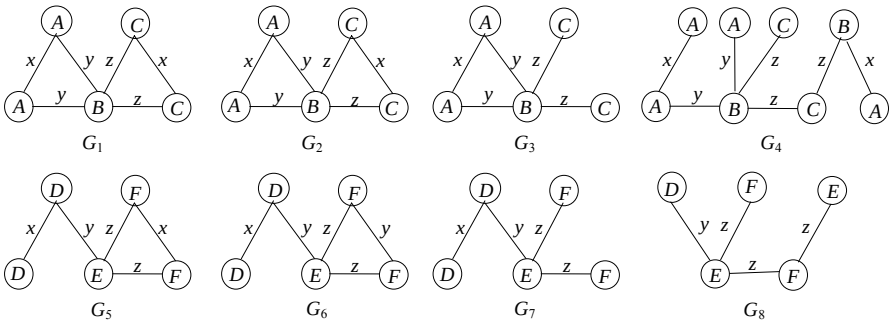


Fig.2 A graph database D

图 2 图数据库 D

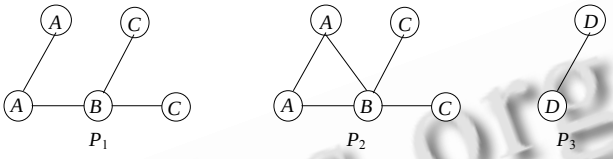


Fig.3 A subset of graph patterns

图 3 图模式子集

根据上面给出的定义,现定义本文要解决的问题如下:

定义 2.5(频繁跳跃模式挖掘).

输入:图数据库 $D=\{G_1,G_2,\ldots,G_n\}$,最小支持度阈值 min_sup,最小绝对跳跃阈值Δ(或最小相对跳跃阈值δ);

输出:D 中所有频繁的绝对Δ-跳跃模式(或相对δ-跳跃模式)集合,即$\{P|supp(P;D)\geqslant \min_sup, JV_{abs}(P;D)>\Delta\}$(或者$\{P|supp(P;D)\geqslant \min_sup, JV_{rel}(P;D)>\delta\}$).

当上下文清楚的时候,我们就使用Δ-跳跃模式表示绝对Δ-跳跃模式,使用δ-跳跃模式表示相对δ-跳跃模式.

2.2 跳跃模式性质

引理 2.1. 设 P 是数据库 D 中的一个Δ_1-跳跃模式.若$\Delta_1>\Delta_2$,则 P 也是数据库 D 中的一个Δ_2-跳跃模式.

证明:根据绝对Δ-跳跃模式定义,$JV_{abs}(P;D)>\Delta_1$.因为$\Delta_1>\Delta_2$,所以 $JV_{abs}(P;D)>\Delta_2$.

因此,P 是 D 中的Δ_2-跳跃模式. □

引理 2.2. 设 P 是数据库 D 中的一个δ_1-跳跃模式.若$\delta_1>\delta_2$,则 P 也是数据库 D 中的一个δ_2-跳跃模式.

证明:类似于引理 2.1 的证明. □

引理 2.3. P 是数据库 D 中的一个绝对 0-跳跃模式,当且仅当 P 是数据库 D 中的闭图模式.

证明:设 P 是 D 中的绝对 0-跳跃模式.根据绝对Δ-跳跃模式定义,$JV_{abs}(P;D)>0$.如果存在 P 的真超图 P',使得 $supp(P;D)=supp(P';D)$,再根据绝对跳跃值定义,$JV_{abs}(P;D)\leqslant supp(P;D)-supp(P';D)=0$.这与 $JV_{abs}(P;D)>0$ 相矛盾.因此,P 是 D 中的闭图模式.

设 P 是 D 中的闭图模式.根据闭图模式定义,不存在 P 的真超图 P',使得 $supp(P;D)=supp(P';D)$.因此,对 P 的任何真超图 P'都有 $supp(P;D)-supp(P';D)>0$.根据绝对跳跃值定义,可得 $JV_{abs}(P;D)>0$.再根据绝对Δ-跳跃模式定义,可知 P 是 D 中的绝对 0-跳跃模式. □

引理 2.4. P 是数据库 D 中的一个相对 0-跳跃模式,当且仅当 P 是数据库 D 中的闭图模式.

证明:类似于引理 2.3 的证明. □

引理 2.3 和引理 2.4 说明,闭图模式实际上是一种特殊的跳跃模式(0-跳跃模式).因此,本文给出的算法也可以直接用来挖掘闭频繁图模式.

跳跃模式的一个重要特征是,它们具有一定程度的稳定性.例如:当数据库中的数据发生变化时,闭图模式集合中的Δ-跳跃模式(Δ>0)仍然可能是闭图模式;并且Δ值越大,这种可能性就越大.见下面的引理和推论.

引理 2.5. 设 P 是数据库 D 中的Δ_1-跳跃模式.D'是另一个数据库,并且满足$|D-D'|\leqslant\Delta_2(0\leqslant\Delta_2\leqslant\Delta_1)$,则 P 是 D'中的$(\Delta_1-\Delta_2)$-跳跃模式.

证明:用反证法证明该引理.假定 P 不是数据库 D'中的$(\Delta_1-\Delta_2)$-跳跃模式.根据绝对跳跃模式定义,存在 P 的真超图 P',使得 $supp(P;D')-supp(P';D')\leqslant\Delta_1-\Delta_2$.因为

$$\begin{aligned}supp(P;D')-supp(P';D')&=(supp(P;D'-D)+supp(P;D'\cap D))-(supp(P';D'-D)+supp(P';D'\cap D))\\&=(supp(P;D'-D)-supp(P';D'-D))+(supp(P;D'\cap D)-supp(P';D'\cap D))\\&\geqslant supp(P;D'\cap D)-supp(P';D'\cap D),\end{aligned}$$

所以 $supp(P;D'\cap D)-supp(P';D'\cap D)\leqslant\Delta_1-\Delta_2$.

因为$|D-D'|\leqslant\Delta_2$,可得 $supp(P;D-D')\leqslant\Delta_2$.因此,

$$\begin{aligned}supp(P;D)-supp(P';D)&=(supp(P;D-D')+supp(P;D\cap D'))-(supp(P';D-D')+supp(P';D\cap D'))\\&=(supp(P;D-D')-supp(P';D-D'))+(supp(P;D\cap D')-supp(P';D\cap D'))\\&\leqslant supp(P;D-D')+(\Delta_1-\Delta_2)\\&\leqslant\Delta_2+(\Delta_1-\Delta_2)=\Delta_1.\end{aligned}$$

根据绝对跳跃模式定义,可知 P 不是数据库 D 中的Δ_1-跳跃模式.这与已知条件相矛盾.因此,P 一定是数据库 D'中的一个$(\Delta_1-\Delta_2)$-跳跃模式. □

推论 2.1. 设 P 是数据库 D 中的Δ-跳跃模式.D'是另一个数据库,满足$|D-D'|\leqslant\Delta$,则 P 是 D'中的闭图模式.

证明:根据引理 2.3 和引理 2.5,容易证明该推论. □

推论 2.1 说明,在删除或改变数据库 D 中任意Δ个数据图之后,D 中原有的Δ-跳跃模式仍然是 D 中的闭图模式.对于δ-跳跃模式,我们也有如下类似的引理和推论.

引理 2.6. 设 P 是数据库 D 中的δ_1-跳跃模式.D'是另一个数据库,满足$|D-D'|\leqslant supp(P;D)\times\delta_2(0\leqslant\delta_2\leqslant\delta_1)$,则 P 是 D'中的$(\delta_1-\delta_2)$-跳跃模式.

证明:类似于引理 2.5 的证明. □

推论 2.2. 设 P 是数据库 D 中的δ-跳跃模式.D'是另一个数据库,满足$|D-D'|\leqslant supp(P;D)\times\delta$,则 P 是 D'中的闭图模式.

证明:根据引理 2.4 和引理 2.6,容易证明该推论. □

下面通过实验证明:随着Δ值的增加,Δ-跳跃模式对噪声的抗干扰能力增强.即,当向数据库中加入噪声而使数据库发生改变时,原来的Δ-跳跃模式有很大的可能性仍然是Δ-跳跃模式.Δ值越大,这种可能性越大.

下面的实验使用 CA 数据集(见第 4.1 节).对 CA 中的每个图,应用下面的过程加入噪声.对 CA 中每个图 G 的每个结点 v,扔一枚不公平的硬币,正面出现的概率为 p(噪声比),反面出现的概率为 $1-p$.如果正面出现,则将 v 的标号改为任意的一个结点标号(从所有结点标号中均匀地随机选择).如果反面出现,则不对 v 进行任何改动,继续处理 G 的下一个结点.使用上面的过程加入噪声,可以从数据集 D 创建一个新的数据集 D'.现研究如下问题:

D 中的频繁Δ-跳跃模式有多大可能性仍然是 D'中的频繁Δ-跳跃模式?

为了度量这种可能性,使用如下定义的稳定率.设 FJS 是 D 中的频繁Δ-跳跃模式集合,FJS'是 D'中的频繁Δ-跳跃模式集合.频繁Δ-跳跃模式的稳定率定义为$|FJS\cap FJS'|/|FJS|$.

使用最小相对支持度 min_sup=10%,对上面的实验重复 10 次取平均值,结果如图 4 所示.为了比较,我们也显示了闭图模式(0-跳跃模式)的稳定率.可以看出,随着Δ值的增加,Δ-跳跃模式的稳定率也在增加.Δ-跳跃模式的这一特性非常重要,因为有意义的图模式经常是那些具有高跳跃值的跳跃模式.跳跃模式的跳跃值越高,它的稳定性越强.因此,即使在数据中存在大量噪声的情况下,通过挖掘Δ-跳跃模式,我们仍能得到有意义的图模式.同样,δ-跳跃模式也具有这一特性,随着δ值的增加,δ-跳跃模式的抗噪声干扰能力也在增强.

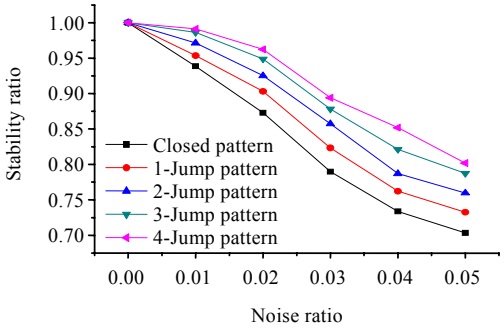


Fig.4 Stability ratio of Δ-jump patterns

图 4 Δ-跳跃模式的稳定率

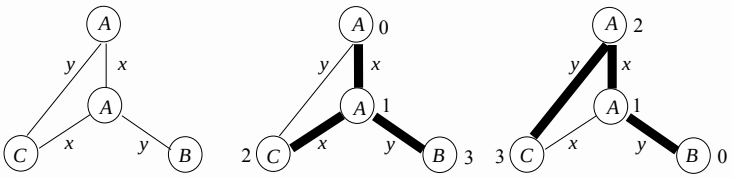
3 挖掘频繁跳跃模式

3.1 DFS编码搜索树

本文的算法利用了著名的 DFS 编码搜索树[3]来挖掘频繁跳跃模式.本节简单介绍有关的基本概念.

3.1.1 最小 DFS 编码

当在图 G 上执行深度优先搜索(DFS)时,一个对应的 DFS 树被创建.不同的 DFS 搜索可以创建不同的 DFS 树.例如:对图 5(a)中的 G,图 5(b)和图 5(c)显示了两棵不同的 DFS 树.在 DFS 树中的边称为前边(在图 5(b)和图 5(c)中加粗显示),否则称为后边.如图 5(b)和图 5(c)所示,根据 DFS 访问结点的顺序,我们可以对每个结点进行编号.边 e 可以表示成(i,j),其中,i 和 j 是与 e 相邻的结点标识(编号).如果 $i<j$,$e=(i,j)$表示一条前边,否则 $e=(i,j)$表示一条后边.



(a) 图 G (a) Gragh G

(b) DFS 树 T_1 (b) DFS tree T_1

(c) DFS 树 T_2 (c) DFS tree T_2

Fig.5 DFS trees of a graph

图 5 图的 DFS 树

给定图 G 的一个 DFS 树 T,可以在 G 的所有边上定义一种线性顺序$\prec$如下:设 $e_1=(i_1,j_1)$和 $e_2=(i_2,j_2)$是 G 的任意两条边.

(1) 若 e_1 和 e_2 都是前边,$j_1<j_2$ 或者$(i_1>i_2\wedge j_1=j_2)$,则 $e_1\prec e_2$;

(2) 若 e_1 和 e_2 都是后边,$i_1<i_2$ 或者$(i_1=i_2\wedge j_1<j_2)$,则 $e_1\prec e_2$;

(3) 若 e_1 是前边,e_2 是后边,$j_1<i_2$,则 $e_1\prec e_2$;

(4) 若 e_1 是后边,e_2 是前边,$i_1<j_2$,则 $e_1\prec e_2$.

现在将边 $e=(i,j)$表示为五元组(i,j,l_i,l_e,l_j),其中,l_i 和 l_j 为对应结点的标号,l_e 为边标号.给定图 G 的一个 DFS 树 T,根据上面定义的线性顺序,可以为 G 定义一个 DFS 编码 $code(G,T)$.例如:如果图 5(b)中的 T_1 是图 5(a)中 G 的 DFS 树,则有 $code(G,T_1)=\{(0,1,A,x,A)$-$(1,2,A,x,C)$-$(2,0,C,y,A)$-$(1,3,A,y,B)\}$.

假定在图 G 的标号集合上也存在一种线性顺序,那么根据边上的线性顺序和标号集合上的线性顺序,可以构造一种 DFS 字典顺序.根据 DFS 字典顺序,任何两个 DFS 编码都可以比较大小.例如在图 5 中,$code(G,T_1)<code(G,T_2)$.在图 G 的所有 DFS 编码中,最小 DFS 编码 $\min(G)$称为 G 的规范编码.在图 5 中,$\min(G)=code(G,T_1)$.最小 DFS 编码的一个重要性质是:如果两个图 G_1 和 G_2 同构,当且仅当 $\min(G_1)=\min(G_2)$.

3.1.2 最右扩展

给定图 G 的一个 DFS 树 T,第 1 个被访问的结点称为根,最后一个被访问的结点称为最右结点.从根到最右结点的路径称为最右路径.例如在图 5(c)中,最右路径是 0→1→2→3.当扩展频繁子图时,文献[3]执行如下受限的扩展,避免产生重复的频繁子图:(1) 向后扩展:连接最右结点和最右路径上的另一个结点,增加一条新边;(2) 向前扩展:引进一个新结点,连接最右路径上的一个结点和该新结点,增加一条新边.这两种受限的扩展称为最右扩展.

3.1.3 DFS 编码搜索树

文献[3]采用先深搜索的办法挖掘频繁子图,只在最小 DFS 编码上进行最右扩展,而且可以保证挖掘结果的完整性.当采用此办法挖掘频繁子图时,会在搜索的过程中形成一个树状的搜索空间(称为 DFS 编码搜索树).其中的每个结点代表一个频繁子图.图 6 显示了从图 2 中的数据库挖掘频繁子图时(min_sup=2)形成的 DFS 编码搜索树.在该编码搜索树每个结点的旁边都有一对数(a:b),其中,a 表示该结点被枚举的顺序编号,b 表示该结点代表的频繁子图的支持度.在下文中,我们有时使用结点的顺序编号表示该结点代表的频繁子图.

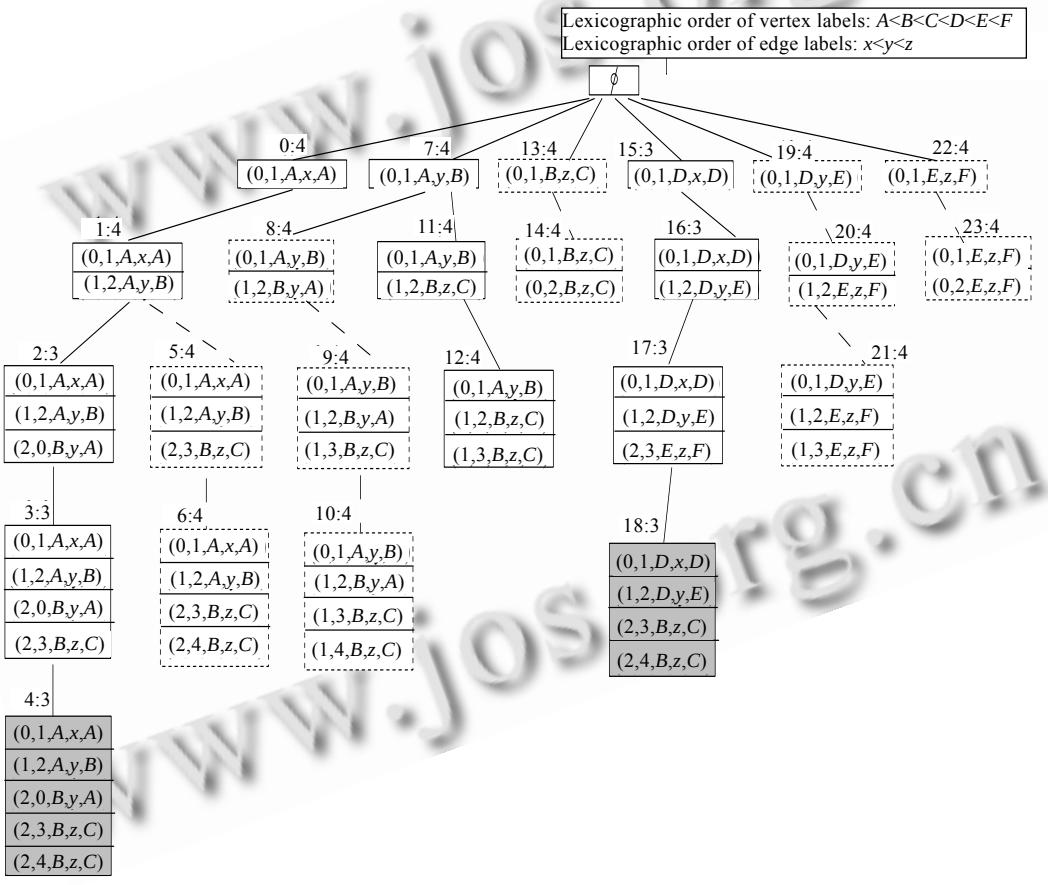


Fig.6 DFS codes tree of frequent subgraphs obtained by mining the database shown in Fig.2 with min_sup=2

图 6 用 min_sup=2 挖掘图 2 中的数据库时形成的 DFS 编码搜索树

3.2 裁剪搜索空间的基本思想

利用DFS编码搜索树,可以设计一种直接的算法来挖掘频繁跳跃模式.具体说,每当发现一个频繁子图p时,我们搜索p的所有多一条边的超图并计算它们的支持度,然后根据跳跃模式定义判断p是否为跳跃模式.然而,这种算法效率太低,因为它需要首先枚举所有的频繁子图.

实际上,当挖掘频繁跳跃模式时,DFS编码搜索树中的很多分枝不需要枚举,因为那些分枝不含有任何频繁跳跃模式.例如:当使用min_sup=2和Δ=1挖掘图2中的数据库时,形成的DFS编码搜索树如图6所示.该搜索树中的两个灰色结点对应仅有的两个频繁Δ-跳跃模式,其他结点都不是频繁Δ-跳跃模式.原则上,该搜索树中不包含频繁Δ-跳跃模式的分枝都应该被裁剪掉.本文给出的裁剪技术能够裁剪掉该搜索树中所有虚线连接的分枝.

在介绍具体的裁剪技术之前,先定义几个记号.$p\Diamond e$表示在图模式p中增加一条新边e后得到的一个图模式.注意:$p\Diamond e$表示e可能是也可能不是p的最右扩展.$p<q$表示p的最小DFS编码小于q的最小DFS编码,即$\min(p)<\min(q)$.根据最小DFS编码定义,容易证明下面两个引理,它们将在后面的证明中被使用.

引理3.1. 设p是DFS编码搜索树中的一个结点,q是p的任意一个后裔,则$p\subset q, p<q$.

引理3.2. 如果$p\Diamond e<p, p\Diamond e\subseteq q$,则$q<p$.

下面给出裁剪技术的基本思想.设p是DFS编码搜索树中的任一结点,如果p的所有后裔都不是跳跃模式,那么可以安全地裁剪以p为根的搜索分枝.具体说,如果对p的任意后裔q,都存在扩展边e满足$supp(q)-supp(q\Diamond e)\leqslant\Delta$,说明$p$为根的搜索分枝不含有$\Delta$-跳跃模式,可以安全地裁剪该分枝.基于这个思想,本文提出了两种新的裁剪技术——基于内扩展的裁剪和基于外扩展的裁剪,它们将在下面两节加以详细介绍.

3.3 基于内扩展的裁剪

首先定义几个新概念.在图模式p中增加一条新边e可得到图模式$p\Diamond e$.如果e引进一个新结点,称e是p的外扩展边,否则称e是p的内扩展边.下面,我们使用$e=(i,j,e_l)$表示e是p的内扩展边,其中,i和j是p的结点标识,e_l是新边的标号.我们使用$e=(i,e_l,v_l)$表示e是p的外扩展边,其中,i是p的结点标识,e_l是新边的标号,v_l是新结点的标号.注意:如果e是p的内扩展边,e可能是也可能不是p的向后扩展边.例如在图6中,$e=(2,0,y)$是图模式{(0,1,A,x,A)-(1,2,A,y,B)-(2,3,B,z,C)}(结点5)的内扩展边,但不是它的向后扩展边.

定义3.1(内关联). 设G是数据库中的一个图,图模式p是G的一个子图,$e=(i,j,e_l)$是图模式p的内扩展边.如果对p到G的每一个子图同构f,G都有一条标号为e_l的边$(f(i),f(j))$,则称在G上(i,j,e_l)内关联于p.

如果在图G上$e=(i,j,e_l)$内关联于p,意味着p在G上的每次出现都蕴含着$p\Diamond e$的出现.

定义3.2(内关联数). 设D是一个图数据库,$e=(i,j,e_l)$是图模式p的内扩展边.在D上,e相对于p的内关联数定义为$Ass_Internal_D(e,p)=|\{G_i|G_i\in D, e$在$G_i$上内关联于$p\}|$.

例3.1:边$e=(2,0,y)$是图模式$p=\{(0,1,A,x,A)-(1,2,A,y,B)\}$的内扩展边.对于图2中的数据库$D$,容易验证,在$G_1$,$G_2$和$G_3$上,$e$内关联于$p$.因此,在$D$上,$e$相对于$p$的内关联数$Ass_Internal_D(e,p)=3$.

引理3.3. 设D是图数据库,$e=(i,j,e_l)$是图模式p的内扩展边.如果$Ass_Internal_D(e,p)\geqslant supp(p)-\Delta, p\subset q, p\Diamond e\not\subseteq q$,则$supp(q)-supp(q\Diamond e)\leqslant\Delta$.

证明:把p的支持集划分成不相交的两个子集S_1和S_2,其中,$S_1=\{G_i|G_i\in D$,在G_i上e内关联于$p\}$,$S_2=\{G_i|G_i\in D$,在G_i上e不与p内关联$\}$.显然,$supp(p)=|S_1|+|S_2|$,$Ass\_Internal_D(e,p)=|S_1|$.因为$p\subset q$,我们可以假定在$S_1$里有$x$个图含有$q$,在$S_2$里有$y$个图含有$q$.因此,$supp(q)=x+y$.设$G$是$S_1$中含有$q$的任意一个图,$I_q$是$q$在$G$中的任意一个实例.因为$p\subset q$,$I_q$也含有$p$在$G$中的一个实例$I_p$.因为在$G$上$e$内关联于$p$,所以$I_p$能够被扩展成$I_p\Diamond e$.因为$p\Diamond e\not\subseteq q$,我们有$I_p\Diamond e\not\subseteq I_q$.因为$e$是$p$的内扩展边,使用$e$扩展$I_q$不会引进新的结点.因此,$I_q$能够被$e$扩展成$I_q\Diamond e$,$G$含有$q\Diamond e$.因为$G$是$S_1$中含有$q$的任意一个图,我们有$supp(q\Diamond e)\geqslant x$.因此,

$$supp(q)-supp(q\Diamond e)\leqslant(x+y)-x=y\leqslant|S_2|=supp(p)-|S_1|=supp(p)-Ass\_Internal_D(e,p)\leqslant\Delta.$$ □

根据引理3.1~引理3.3,我们为Δ-跳跃模式导出如下基于内扩展的裁剪条件.

定理3.1. 设D是图数据库,$e=(i,j,e_l)$是图模式p的内扩展边.如果$Ass_Internal_D(e,p)\geqslant supp(p)-\Delta, p\Diamond e<p$,则在DFS编码树上,$p$的所有后裔都不是$\Delta$-跳跃模式.

证明:设在 DFS 编码树上,q 是 p 的任意一个后裔.根据引理 3.1,我们有 $p\subset q$,$p<q$.现在用反证法证明 $p\Diamond e\not\subseteq q$.如果 $p\Diamond e\subseteq q$,根据引理 3.2,我们有 $q<p$.这与 $p<q$ 相矛盾.因此,$p\Diamond e\not\subseteq q$.因为 $Ass\_Internal_D(e,p)\geqslant supp(p)-\Delta$,$p\subset q$,$p\Diamond e\not\subseteq q$,根据引理 3.3,我们有 $supp(q)-supp(q\Diamond e)\leqslant\Delta$.根据$\Delta$-跳跃模式的定义,$q$ 不是Δ-跳跃模式. □

对于 DFS 编码树中的一个图模式 p,如果定理 3.1 中的条件成立,则 p 的所有后裔都不是Δ-跳跃模式.而且,因为 $supp(p)-supp(p\Diamond e)\leqslant supp(p)-Ass\_Internal_D(e,p)\leqslant\Delta$,可知 p 本身也不是Δ-跳跃模式.因此,可以安全地裁剪以 p 为根的分枝.这种裁剪技术称为基于内扩展的裁剪.

例 3.2:假定用 min_sup=2 和Δ=1 在图 2 中的数据库中挖掘频繁Δ-跳跃模式.生成的 DFS 编码搜索树如图 6 所示.该搜索树中结点 5 对应的图模式为 p={(0,1,A,x,A)-(1,2,A,y,B)-(2,3,B,z,C)},边 e=(2,0,y)是 p 的内扩展边.根据最小 DFS 编码定义,$p\Diamond e<p$.容易计算 $supp(p)$=4,$Ass\_Internal_D(e,p)$=3.因此,$Ass\_Internal_D(e,p)\geqslant supp(p)-\Delta$.利用基于内扩展的裁剪技术,以结点 5 为根的分枝可以被裁剪掉.同理,以结点 8 为根的分枝也可以利用这种裁剪技术被裁剪掉.

类似地,我们也为δ-跳跃模式导出了如下基于内扩展的裁剪条件.

定理 3.2. 设 D 是图数据库,min_sup 是最小支持度,$e=(i,j,e_l)$是图模式 p 的内扩展边.如果

$$Ass\_Internal_D(e,p)\geqslant supp(p)-\min\_sup\times\delta, p\Diamond e<p,$$

则在 DFS 编码树上 p 的所有后裔都不是频繁δ-跳跃模式.

证明:设在 DFS 编码树上,q 是 p 的任意一个后裔.根据引理 3.1,我们有 $p\subset q$,$p<q$.现在用反证法证明 $p\Diamond e\not\subseteq q$.如果 $p\Diamond e\subseteq q$,根据引理 3.2,我们有 $q<p$.这与 $p<q$ 相矛盾.因此,$p\Diamond e\not\subseteq q$.因为 $Ass\_Internal_D(e,p)\geqslant supp(p)-\min\_sup\times\delta$,$p\subset q$,$p\Diamond e\not\subseteq q$,根据引理 3.3,我们有 $supp(q)-supp(q\Diamond e)\leqslant\min\_sup\times\delta$.根据 q 是否频繁,我们分两种情况加以讨论:

(1) 如果 q 不是一个频繁模式,结论显然成立;

(2) 如果 q 是一个频繁模式,则 $supp(q)\geqslant\min\_sup$.因此,

$$\frac{supp(q)-supp(q\Diamond e)}{supp(q)}\leqslant\frac{\min\_sup\times\delta}{\min\_sup}=\delta.$$

根据δ-跳跃模式的定义,q 不是δ-跳跃模式,结论成立. □

3.4 基于外扩展的裁剪

除了利用内扩展边导出有效的剪枝条件以外,我们还可以利用外扩展边设计有效的裁剪技术.首先,给出外关联和外关联数的定义.

定义 3.3(外关联). 设 G 是数据库中的一个图,图模式 p 是 G 的一个子图,$e=(i,e_l,v_l)$是图模式 p 的外扩展边.如果对 p 到 G 的每一个子图同构 f,G 都有一个结点 v,并且同时满足如下 3 个条件:

(1) 结点 v 的标号是 v_l;

(2) ($f(i)$,v)是一条标号为 e_l 的边;

(3) 不存在 p 中的结点 j,使得 $f(j)=v$.

那么,称在 G 上 $e=(i,e_l,v_l)$外关联于 p.

定义 3.4(外关联数). 设 D 是图数据库,$e=(i,e_l,v_l)$是图模式 p 的外扩展边.在 D 上,e 相对于 p 的外关联数定义为 $Ass_External_D(e,p)=|\{G_i|G_i\in D,$在 G_i 上 e 外关联于 $p\}|$.

例 3.3:边 e=(0,x,D)是图模式 p={(0,1,D,y,E)}的外扩展边.对于图 2 中的数据库 D,容易验证,在 G_5,G_6 和 G_7 上,e 外关联于 p.因此,在 D 上,e 相对于 p 的外关联数 $Ass\_External_D(e,p)$=3.

如果在图 G 上,$e=(i,e_l,v_l)$外关联于 p,则意味着 p 在 G 上的每次出现都蕴含着 $p\Diamond e$ 的出现.注意:如果 e 是 p 的外扩展边,则 e 可能是也可能不是 p 的向前扩展边.例如:边 e=(2,z,C)是图模式{(0,1,A,x,A)-(1,2,A,y,B)}的外扩展边和向前扩展边;然而 e=(2,x,A)只是图模式{(0,1,A,y,B)-(1,2,B,y,A)-(1,3,B,z,C)}的外扩展边,但不是它的向前扩展边.

利用引理 3.3 的思想,很可能做出这样的假设:设 D 是图数据库,$e=(i,e_l,v_l)$是图模式 p 的外扩展边.如果

$$Ass\_External_D(e,p)\geqslant supp(p)-\Delta, p\subset q, p\Diamond e\not\subseteq q,$$

则 $supp(q)-supp(q\Diamond e)\leqslant\Delta$.

然而,这样的假设并不成立.请看下面的反例.

例 3.4:考虑图 7 中的数据库和模式.边 $e=(1,y,C)$是图模式 $p=\{(0,1,A,x,B)\}$的外扩展边.容易计算得到

$$supp(p)=3, Ass\_External_D(e,p)=3.$$

设$\Delta=1$,显然,$Ass\_External_D(e,p)\geqslant supp(p)-\Delta$.对于图 7 中的模式 q,我们有 $p\subset q, p\Diamond e\not\subseteq q$.上面假设的条件成立.然而,$supp(q)=3$,$supp(q\Diamond e)=1$.因此,$supp(q)-supp(q\Diamond e)>\Delta$.上面假设的结论并不成立.

在上面的例子中,尽管在 G_1,G_2,G_3 上,边 $e=(1,y,C)$外关联于图模式 $p=\{(0,1,A,x,B)\}$,然而在 G_1,G_2 上,边 e 中标号为 C 的端点只能出现在 q 的实例中.因此,在 G_1,G_2 上,q 不能被扩展成 $q\Diamond e$,从而使得上面假设的结论不成立.对于内关联,这种情况从来不会发生,因为使用内扩展边扩展图模式时不会引进任何新结点.对于一般的外关联,我们无法保证例子 3.4 中的情况不会发生.因此,我们不能像内关联那样简单地利用外关联设计裁剪技术.

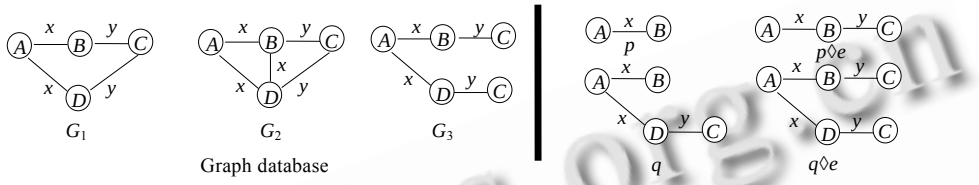


Fig.7 A counter example based on external extension pruning

图 7 基于外扩展裁剪的反例

幸运的是,当外关联满足一些额外的约束时,可以保证例子 3.4 中的情况不会发生.在介绍额外约束之前,再来定义几个新概念.在一个图中,不在任何环中出现的边称为桥.例如在如图 7 所示的 G_3 中,标号为 y 的两条边就是桥.

定义 3.5(桥关联). 设 G 是数据库中的一个图,图模式 p 是 G 的一个子图,$e=(i,e_l,v_l)$是图模式 p 的外扩展边.如果对 p 到 G 的每一个子图同构 f,G 都有一个结点 v,并且同时满足如下 4 个条件:

(1) 结点 v 的标号是 v_l;

(2) $(f(i),v)$是一条标号为 e_l 的边;

(3) 不存在 p 中的结点 j,使得 $f(j)=v$;

(4) $(f(i), v)$是 G 中的桥.

那么,称在 G 上 $e=(i,e_l,v_l)$桥关联于 p.

定义 3.6(桥关联数). 设 D 是图数据库,$e=(i,e_l,v_l)$是图模式 p 的外扩展边.在 D 上,e 相对于 p 的桥关联数定义为 $Ass_Bridge_D(e,p)=|\{G_i|G_i\in D,$在 G_i 上,e 桥关联于 $p\}|$.

例 3.5:边 $e=(1,y,C)$是图模式 $p=\{(0,1,A,x,B)\}$的外扩展边.对于如图 7 所示的数据库 D,可以验证,在 G_1,G_2,G_3 上,e 外关联于 p,因此,$Ass\_External_D(e,p)=3$.然而,只在 G_3 上,e 桥关联于 p,因此,$Ass\_Bridge_D(e,p)=1$.

根据外关联和桥关联的定义可以看出,桥关联是一种特殊的外关联.因此,桥关联数应小于等于外关联数,即 $Ass\_Bridge_D(e,p)\leqslant Ass\_External_D(e,p)$.使用桥关联,我们有下面的引理.

引理 3.4. 设 D 是图数据库,$e=(i,e_l,v_l)$是图模式 p 的外扩展边.如果 $Ass\_Bridge_D(e,p)\geqslant supp(p)-\Delta, p\subset q, p\Diamond e\not\subseteq q$,则 $supp(q)-supp(q\Diamond e)\leqslant\Delta$.

证明:类似于引理 3.3 的证明. □

根据引理 3.1、引理 3.2 和引理 3.4,我们可以为Δ-跳跃模式导出如下基于外扩展的剪枝条件.

定理 3.3. 设 D 是图数据库,$e=(i,e_l,v_l)$是图模式 p 的外扩展边.如果 $Ass\_Bridge_D(e,p)\geqslant supp(p)-\Delta, p\Diamond e<p$,则在 DFS 编码树上,$p$ 的所有后裔都不是Δ-跳跃模式.

证明:参照定理 3.1 的证明过程,根据引理 3.1、引理 3.2 和引理 3.4 容易证明该定理. □

例 3.6:演示定理 3.3 的作用.假定用 min_*sup*=2 和Δ=1 在图 2 中的数据库 D 中挖掘频繁Δ-跳跃模式.生成的 DFS 编码搜索树如图 6 所示.该搜索树中结点 19 对应的图模式为 p={(0,1,*D*,*y*,*E*)},边 e=(0,*x*,*D*)是 p 的外扩展边.根据最小 DFS 编码定义,$p\Diamond e<p$.容易计算 $supp(p)$=4,$Ass\_Bridge_D(e,p)$=3.因此,$Ass\_Bridge_D(e,p)\geqslant supp(p)-\Delta$.根据定理 3.3,以结点 19 为根的分枝不含有Δ-跳跃模式.该分枝可以被裁剪掉.

除了使用桥关联,在某些特殊情况下,仅仅使用外关联也可以得到有效的剪枝条件,如定理 3.4 所示.

定理 3.4. 设 D 是图数据库,$e=(i,e_l,v_l)$是图模式 p 的外扩展边.如果 $Ass\_External_D(e,p)\geqslant supp(p)-\Delta$,并且根据标号集上定义的顺序 v_l 小于 p 中所有结点的标号,则在 DFS 编码树上,p 的所有后裔都不是Δ-跳跃模式.

证明:设在 DFS 编码树上 q 是 p 的任意一个后裔,根据引理 3.1,我们有 $p\subset q$,$p<q$.现在用反证法证明 q 中不含有标号为 v_l 的结点.若 q 中含有标号为 v_l 的结点,根据最小 DFS 编码定义,我们有 $q<p$.这与 $p<q$ 相矛盾.

把 p 的支持集划分成不相交的两个子集 S_1 和 S_2,其中,S_1={G_i|$G_i\in D$,在 G_i 上,e 外关联于 p},S_2={G_i|$G_i\in D$,在 G_i 上,e 不与 p 外关联}.显然,$supp(p)=|S_1|+|S_2|$,$Ass\_External_D(e,p)=|S_1|$.因为 $p\subset q$,我们可以假定在 S_1 里有 x 个图含有 q,在 S_2 里有 y 个图含有 q.因此,$supp(q)=x+y$.设 G 是 S_1 中含有 q 的任意一个图,I_q 是 q 在 G 中的任意一个实例.因为 $p\subset q$,I_q 也含有 p 在 G 中的一个实例 I_p.因为 e 在 G 上外关联于 p,I_p 能够被扩展成 $I_p\Diamond e$,$I_p\Diamond e$ 中引进了一个标号为 v_l 的新结点 v.因为 q 中不含有标号为 v_l 的结点,所以 I_q 中不含有这个新结点 v.因此,I_q 能够被扩展得到 $I_q\Diamond e$,G 含有 $q\Diamond e$.因为 G 是 S_1 中含有 q 的任意一个图,我们有 $supp(q\Diamond e)\geqslant x$.因此,

$$supp(q)-supp(q\Diamond e)\leqslant(x+y)-x=y\leqslant|S_2|=supp(p)-|S_1|=supp(p)-Ass\_External_D(e,p)\leqslant\Delta.$$

根据Δ-跳跃模式的定义,q 不是Δ-跳跃模式. □

例 3.7:演示定理 3.4 的作用.假定用 min_*sup*=2 和Δ=1 在图 2 中的数据库 D 中挖掘频繁Δ-跳跃模式.生成的 DFS 编码搜索树如图 6 所示.该搜索树中结点 13 对应的图模式为 p={(0,1,*B*,*z*,*C*)},边 e=(0,*y*,*A*)是 p 的外扩展边.容易计算,$supp(p)$=4,$Ass\_External_D(e,p)$=3.因此,$Ass\_External_D(e,p)\geqslant supp(p)-\Delta$.而且,标号 A 小于 p 中所有结点的标号.定理 3.4 的条件被满足,因此,以结点 13 为根的分枝可以被裁剪掉.

利用外扩展边,定理 3.3 和定理 3.4 给出了两种不同的裁剪技术.这两种裁剪技术在本文中被称为基于外扩展的裁剪.类似地,我们也为δ-跳跃模式设计了如下基于外扩展的裁剪技术.

定理 3.5. 设 D 是图数据库,min_*sup* 是最小支持度,$e=(i,e_l,v_l)$是图模式 p 的外扩展边.如果

$$Ass\_Bridge_D(e,p)\geqslant supp(p)-\text{min\_}sup\times\delta,p\Diamond e<p,$$

则在 DFS 编码树上,p 的所有后裔都不是频繁δ-跳跃模式.

证明:参考定理 3.2 和定理 3.3 的证明过程,容易证明该定理. □

定理 3.6. 设 D 是图数据库,min_*sup* 是最小支持度,$e=(i,e_l,v_l)$是图模式 p 的外扩展边.

如果 $Ass\_External_D(e,p)\geqslant supp(p)-\text{min\_}sup\times\delta$,并且根据标号集上定义的顺序 v_l 小于 p 中所有结点的标号,则在 DFS 编码树上,p 的所有后裔都不是频繁δ-跳跃模式.

证明:参考定理 3.2 和定理 3.4 的证明过程,容易证明该定理. □

3.5 完整算法

使用第 3.3 节和第 3.4 节给出的裁剪技术可以过滤掉大量的非频繁跳跃模式,然而,剩余的模式也不全是跳跃模式,需要从中选出跳跃模式.幸运的是,利用第 3.3 节和第 3.4 节的裁剪技术,这很容易实现.为了尽早地裁剪图模式搜索空间,对每个频繁图模式 p,我们检查 p 的所有可能扩展边,以判断第 3.3 节和第 3.4 节中的剪枝条件是否满足.因此,对每个频繁图模式 p,我们得到了 p 的所有多一条边的超图及其支持度,根据跳跃模式定义,可以直接判断 p 是否是跳跃模式.本节将第 3.3 节和第 3.4 节的裁剪技术集成到 DFS 编码搜索框架中,给出挖掘频繁Δ-跳跃模式的完整算法 GraphJP.

算法 GraphJP 如下工作:初试化时,算法先扫描数据库,得到频繁边集合,然后删除数据库中不频繁的边和结点,最后检测并标记数据库中每个图的所有桥(定理 3.3 的裁剪技术需要利用这些信息).在这之后,对每个 1-边频繁子图,算法调用子过程 MiningJumpPatterns 进行深度优先搜索,发现所有频繁Δ-跳跃模式.

在子过程 MiningJumpPatterns 的第 1 行,扫描数据库,发现当前图模式 p 的所有可能扩展边(包括非最右扩

展边).所有非最右扩展边记录在 *NRE*,所有最右扩展边记录在 *RE*.根据 DFS 编码搜索树的构造,如果边 e 不是 p 的最右扩展,$p\Diamond e$ 将在 p 之前被发现,而且 $p\Diamond e<p$.因此,对 *NRE* 中的每条边 e 都有 $p\Diamond e<p$.第 2 行、第 3 行对 *NRE* 中的内扩展边应用基于内扩展的裁剪技术.如果 *NRE* 中存在内扩展边 e 使得 $Ass\_Internal_D(e,p)\geqslant supp(p)-\Delta$,根据定理 3.1,说明以 p 为根的分枝不含有 Δ-跳跃模式.因此,可以裁剪以 p 为根的分枝,从子过程返回.类似地,第 4 行~第 7 行对 *NRE* 中的外扩展边应用基于外扩展的裁剪技术(定理 3.3 和定理 3.4).第 8 行计算当前图模式 p 的绝对跳跃值 $JV_{abs}(p)$,因为在第 1 行已经枚举了 p 的所有扩展边,从而得到了 p 的所有多一条边的超图的支持度.如果 $JV_{abs}(p)$大于 Δ,说明 p 是频繁 Δ-跳跃模式,将 p 放入结果集.注意:尽管算法检查 p 的所有可能扩展边,然而算法也像 gSpan[3]那样,只对 p 进行最右扩展.对 *RE* 中的每个最右扩展边 e(第 11 行),如果 $p\Diamond e$ 是频繁的,并且 $p\Diamond e$ 是对应图模式的最小 DFS 编码,则算法对 p 的孩子 $p\Diamond e$ 递归调用子过程继续深度优先搜索(第 13 行).在处理完 p 的一条最右扩展边 e 之后(完成对 $p\Diamond e$ 的递归调用),算法可以利用边 e 再次应用裁剪技术.第 14 行、第 15 行再次使用基于内扩展的裁剪技术.设 e'是剩余的任意一条最右扩展边,$q=p\Diamond e'$是 p 的一个剩余孩子.显然,$q\Diamond e<q$.如果 $Ass\_Internal_D(e,p)\geqslant supp(p)-\Delta$,容易证明,$Ass\_Internal_D(e,q)\geqslant supp(q)-\Delta$.根据定理 3.1 可知,以 q 为根的分枝不含有 Δ-跳跃模式.因此,如果 $Ass\_Internal_D(e,p)\geqslant supp(p)-\Delta$,说明 p 的剩余孩子为根的分枝中都不含有 Δ-跳跃模式,算法可以扔掉 p 的剩余的最右扩展边,从子过程 MiningJumpPatterns 返回.类似地,第 16 行~第 19 行再次应用基于外扩展的裁剪技术(定理 3.3 和定理 3.4).算法完成之后,*JS* 中包含所有频繁 Δ-跳跃模式.

定理 3.7. 算法 GraphJP 产生的结果是正确而完备的.

证明:

(1) 正确性证明.设 p 是结果集 *JS* 中的任一图模式.根据算法的执行步骤可知,p 只能在子过程 MiningJumpPatterns 的第 9 行被加入 *JS*.根据该行的测试条件和 Δ-跳跃模式定义可知,p 是 Δ-跳跃模式.又因为进入子过程 MiningJumpPatterns(被算法 GraphJP 第 6 行和子过程 MiningJumpPatterns 第 13 行调用)的任何图模式一定是频繁的,因此,p 是频繁 Δ-跳跃模式.

(2) 完备性证明.设 p 是数据库中的任一频繁图模式.如果 p 不在结果集 *JS* 中,说明 p 可能在子过程 MiningJumpPatterns 中的第 3 行、第 5 行、第 7 行、第 15 行、第 17 行、第 19 行被裁剪掉,也可能在第 9 行被过滤掉.若 p 在第 3 行或第 15 行被裁剪掉,根据定理 3.1 可知,p 不是 Δ-跳跃模式.若 p 在第 5 行或第 17 行被裁剪掉,根据定理 3.3 可知,p 不是 Δ-跳跃模式.若 p 在第 7 行或第 19 行被裁剪掉,根据定理 3.4 可知,p 不是 Δ-跳跃模式.若 p 在第 9 行被过滤掉,根据 Δ-跳跃模式定义可知,p 不是 Δ-跳跃模式.因此,如果频繁图模式 p 不在结果集 *JS* 中,说明 p 一定不是 Δ-跳跃模式. □

算法(GraphJP).

输入:图数据库 D;最小支持度阈值 min_*sup*;跳跃距离阈值 Δ;

输出:D 中所有频繁 Δ-跳跃模式集合 *JS*.

1. 扫描 D 得到所有频繁边;
2. 删除 D 中不频繁的边和结点;
3. 检测并标记 D 中所有图的桥;
4. S^1={所有频繁边的最小 DFS 编码}; //S^1 中的 DFS 编码按 DFS 字典顺序排序
5. **For** S^1 中的每个 DFS 编码 p
6. 调用子过程 *MiningJumpPatterns*(p,D,min_*sup*,Δ,*JS*);
7. 输出 *JS*;

子过程(MiningJumpPatterns).

输入:DFS 编码 p;图数据库 D;最小支持度阈值 min_*sup*;跳跃距离阈值 Δ;

输出:频繁 Δ-跳跃模式集合 *JS*.

1. 扫描 D,发现 p 的所有扩展边,*RE* 记录 p 的所有最右扩展边,*NRE* 记录 p 的所有非最右扩展边;
2. **If** $\exists e\in NRE$ 使得 $Ass\_Internal_D(e,p)\geqslant supp(p)-\Delta$, **Then**

3. 子过程结束; //定理 3.1 的裁剪技术;
4. **If** $\exists e \in NRE$ 使得 $Ass\_Bridge_D(e,p) \geq supp(p)-\Delta$, **Then**
5. 子过程结束; //定理 3.3 的裁剪技术;
6. **If** $\exists e \in NRE$ 使得 $Ass\_Externa_D(e,p) \geq supp(p)-\Delta$,$e$ 引进的新结点标号小于 p 中所有结点标号, **Then**
7. 子过程结束; //定理 3.4 的裁剪技术;
8. 根据 *RE* 和 *NRE* 的扩展边,计算机 p 的绝对跳跃值 $JV_{abs}(p)$;
9. **If** $JV_{abs}(p)>\Delta$, **Then** 把 p 放入 *JS*;
10. 按 DFS 字典顺序排序 *RE* 中所有最右扩展边;
11. **For** *RE* 中的每个最右扩展边 e
12. **If** $supp(p\Diamond e) \geq \min\_sup$,$p\Diamond e=\min(p\Diamond e)$, **Then**
13. 调用子过程 *MiningJumpPatterns*($p\Diamond e$,*D*,min_*sup*,Δ,*JS*);
14. **If** $Ass\_Internal_D(e,p) \geq supp(p)-\Delta$, **Then**
15. 子过程结束; //定理 3.1 的裁剪技术;
16. **If** $Ass\_Bridge_D(e,p) \geq supp(p)-\Delta$, **Then**
17. 子过程结束; //定理 3.3 的裁剪技术;
18. **If** $Ass\_Externa_D(e,p) \geq supp(p)-\Delta$,$e$ 引进的新结点标号小于 p 中所有结点的标号, **Then**
19. 子过程结束; //定理 3.4 的裁剪技术;

4 实验结果及分析

4.1 实验环境

我们从两个真实的化合物集合中导出了几个图集合.第 1 个化合物集合(PTE)用来评价化合物的毒性,含有 340 个化合物,可从以下网址获得:http://oldwww.comlab.ox.ac.uk/oucl/groups/machlearn/PTE.第 2 个化合物集合(AIDS)用来测试化合物对艾滋病病毒的抑制作用,含有大约 44 000 个化合物,可从以下网址获得:http://dtp.nci.nih.gov/docs/3d_database/structural_information/structural_data.html.AIDS 化合物根据实验结果被分成 3 类:CA(confirmed active),CM(confirmed moderately)和 CI(confirmed inactive).其中,CA 含有 422 个化合物,CM 含有 1 081 个化合物,CI 含有剩余的化合物.除了真实的图数据,我们也使用文献[2]中的图数据生成器生成了具有各种不同特征的图集合.有 6 个参数控制合成的图集合:(1) $|D|$(合成图的数量);(2) $|T|$(合成图的平均大小);(3) $|L|$(潜在的频繁子图模式的数量);(4) $|I|$(潜在的频繁子图模式的平均大小);(5) $|V|$(结点标号的数量);(6) $|E|$(边标号的数量).合成的图集合用类似“*D*1k*V*4*E*2*I*5*T*20*L*20”的记号表示.例如,*D*1k*V*4*E*2*I*5*T*20*L*20 代表参数设置为 $|D|$=1k,$|T|$=20, $|L|$=20,$|I|$=5,$|V|$=4,$|E|$=2 的一个合成图集合.

本文算法使用 C++语言实现,用带有-O3 优化选项的 g++编译.用于实验的计算机具有 PIV 3.0GHz CPU 和 1GB 内存,运行 RedHat Linux 8.0 操作系统.

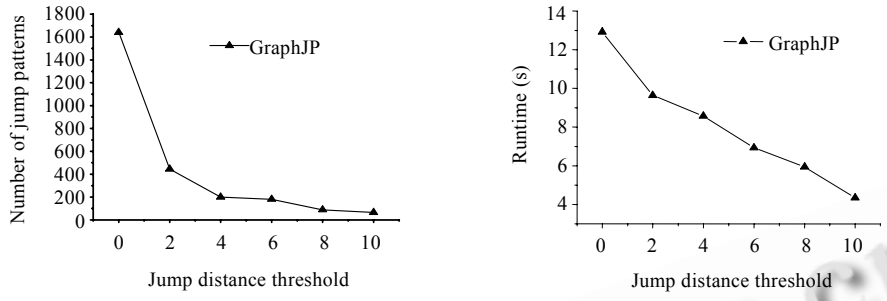
4.2 跳跃距离阈值的作用

本节考察不同的跳跃距离阈值对输出的跳跃模式数量和挖掘效率的影响.对于每个图集合,固定一个尽可能小的支持度,同时变化跳跃距离阈值Δ的大小.图 8 显示了在 PTE 数据集上当Δ值变化时,输出的频繁Δ-跳跃模式数量和算法 GraphJP 的挖掘时间.图 9 显示了在 *D*1k*V*4*E*2*I*5*T*20*L*20 数据集上的实验结果.

从图 8(a)和图 9(a)可以看出,当Δ值增加时,频繁Δ-跳跃模式的数量有所下降.根据Δ-跳跃模式的定义可知,当Δ值增加时,意味着更多的频繁图模式能够被过滤掉,因而频繁Δ-跳跃模式的数量减少.在实验中,我们发现了一个非常有意思的现象:当Δ值从 0 开始稍微增加时,频繁Δ-跳跃模式的数量就急剧下降.例如在 PTE 数据集上,当Δ值从 0 变化到 2 时,频繁Δ-跳跃模式的数量从 1 641 急剧下降到 445.因为 0-跳跃模式等价于闭图模式(见引理 2.3),所以,当Δ稍微大于 0 时,频繁Δ-跳跃模式的数量比频繁闭图模式的数量少很多.在实际应用中,用户往往

希望得到一个小的而有意义的图模式集合,而有意义的图模式经常是那些具有高跳跃值的跳跃模式.因此在实际应用中,我们可以使用大的Δ值来挖掘图数据库,快速地得到一个小而有意义的图模式集合.

从图 8(b)和图 9(b)可以看出,随着Δ值的增加,算法的运行时间也在迅速地缩短.这很容易理解:根据定理 3.1、定理 3.3 和定理 3.4 可知,当Δ值增加时,裁剪条件更容易被满足,更多的分枝不含有Δ-跳跃模式,可以被安全地裁剪掉,因而算法的挖掘效率也被极大地提高.

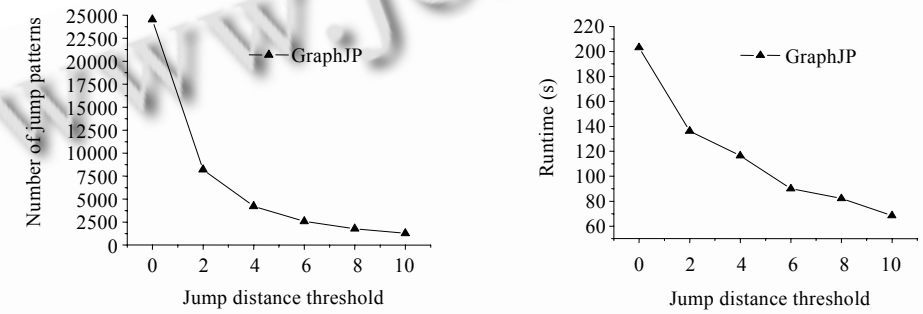


(a) Number of frequent Δ-jump patterns
(a) 频繁Δ-跳跃模式数量

(b) Runtime
(b) 运行时间

Fig.8 Experimental results on PTE (min_*sup*=3.5%)

图 8 在 PTE 数据集上的实验结果(min_*sup*=3.5%)



(a) Number of frequent Δ-jump patterns
(a) 频繁Δ-跳跃模式数量

(b) Runtime
(b) 运行时间

Fig.9 Experimental results on *D*1k*V*4*E*2*I*5*T*20*L*20 (min_*sup*=3.5%)

图 9 在 *D*1k*V*4*E*2*I*5*T*20*L*20 数据集上的实验结果(min_*sup*=3.5%)

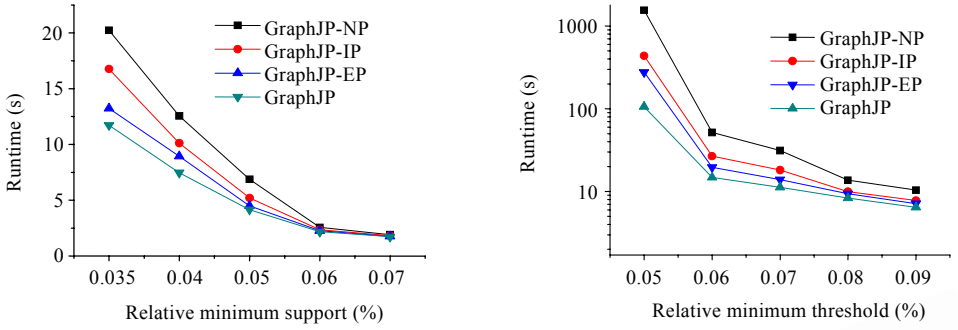
4.3 裁剪技术的有效性

本节评价算法 GraphJP 中使用的裁剪技术的有效性.我们根据裁剪技术的不同组合形成了 GraphJP 算法的各种变体:GraphJP-NP 表示不使用任何裁剪技术,GraphJP-IP 表示只使用基于内扩展的裁剪技术,GraphJP-EP 表示只使用基于外扩展的裁剪技术,GraphJP 表示使用所有裁剪技术.

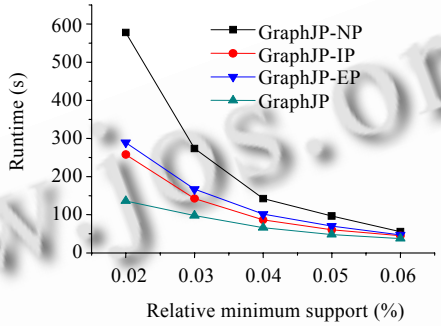
对于每个图集合,我们固定跳跃距离阈值Δ等于 2,变化最小支持度的大小.图 10 显示了不同算法变体的运行时间.可以看出,基于内扩展的裁剪技术和基于外扩展的裁剪技术能够大幅度地提高算法 GraphJP 的效率.而且支持度越低,这两种裁剪技术越有效.例如在 CA 数据集上,当支持度是 5%时,GraphJP 比 GraphJP-NP 快 1 个数量级还多.

在化合物数据集(PTE 和 CA)上,基于外扩展的裁剪技术比基于内扩展的裁剪技术更有效.这是因为化合物中主要以稀疏图为主,大多数频繁子结构都是树结构.因此,外扩展边的数量要明显多于内扩展边,从而使得基

于外扩展的裁剪条件更容易被满足.相反地,合成图集合 *D*1k*V*4*E*2*I*5*T*20*L*20 中含有更多的环,这使得内扩展边的数量多于外扩展边的数量,因此在这个数据集上,基于内扩展的裁剪技术比基于外扩展的裁剪技术更有效.



(a) PTE, Δ=2 (b) CA, Δ=2



(c) *D*1k*V*4*E*2*I*5*T*20*L*20, Δ=2

Fig.10 Effectiveness test of pruning techniques (runtime)

图 10 裁剪技术的有效性(运行时间)

4.4 算法可扩展性

本节研究算法 GraphJP 的可扩展性.我们使用参数 *V*4*E*2*I*5*T*20*L*20 生成了从 10K~100K 的合成数据,固定 min_*sup*=6%,Δ=2,运行 GraphJP,实验结果如图 11 所示.可以看出,算法 GraphJP 具有很好的可扩展性.随着数据量的增加,GraphJP 的运行时间也在线性地增加.

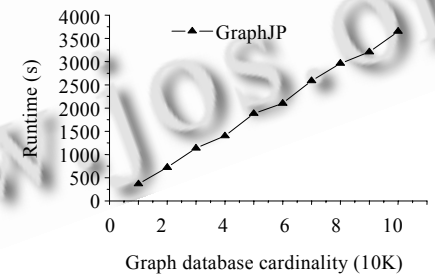


Fig.11 Scalability on *V*4*E*2*I*5*T*20*L*20, min *sup*=6%, Δ=2

图 11 在 *V*4*E*2*I*5*T*20*L*20 上的可扩展性,min_*sup*=6%,Δ=2

5 相关工作

频繁子图挖掘算法大致可以分为两类:第 1 类算法根据 Apriori 性质使用逐级搜索策略来枚举所有频繁子图,包括 AGM[1]和 FSG[2];第 2 类算法采用深度优先搜索策略来枚举所有频繁子图,包括 gSpan[3],MoFa[4],FFSM[5],GASTON[6].通常,第 2 类算法比第 1 类算法有更好的内存利用率,因而具有更高的挖掘效率.

挖掘频繁子图经常会产生指数级数量的图模式,使得挖掘结果难以被利用.为了解决这个问题,两类主要方法已被提出:(1) 挖掘闭频繁图模式[7];(2) 挖掘极大频繁图模式[8,9].第 1 类方法严格强调图模式的支持度,使得图模式的输出数量仍然很大.第 2 类方法只输出搜索空间中边界上的图模式,将会使一些重要图模式丢失.最近,文献[11]提出从图数据库中挖掘一个代表模式集合,使得任何频繁图模式都能被一个代表模式δ-覆盖.然而,文献[11]中最有效的算法 RP-GD 仍然需要枚举所有闭频繁图模式,因此,RP-GD 的挖掘效率要低于 CloseGraph 算法[7].

本文提出了一个新问题:挖掘频繁跳跃模式.跳跃模式扩展了闭图模式的定义,使得闭图模式成为一种特殊的跳跃模式(0-跳跃模式).如第 4.2 节所示,当跳跃距离阈值Δ稍微大于 0 时,频繁Δ-跳跃模式的数量就比频繁闭图模式的数量少很多.而且,挖掘频繁Δ-跳跃模式(Δ>0)也比挖掘频繁闭图模式快很多.

其他与图模式挖掘有关的研究包括利用索引提高挖掘效率[12]、图模式并行挖掘算法[13]、挖掘图产生器[14],等等.

6 结 论

本文提出了从图数据库中挖掘频繁跳跃模式的问题,并给出了解决该问题的第 1 种高效算法 GraphJP.为了高效地裁剪图模式搜索空间,本文提出了两种新的裁剪技术:基于内扩展的裁剪和基于外扩展的裁剪.实验结果表明,本文提出的裁剪技术能够使算法的性能提高一个数量级,GraphJP 算法具有线性可扩展性和高挖掘效率.因为图代表了通用的模式类型,本文给出的度量定义和挖掘方法也很容易被扩展来挖掘其他的模式类型,例如项集模式、序列模式、树模式等.

References:

[1] Inokuchi A, Washio T, Motoda H. An apriori-based algorithm for mining frequent substructures from graph data. In: Cheng M, Yu PS, Liu B, eds. Proc. of the 4th European Conf. on Principles of Data Mining and Knowledge Discovery. Lyon: Springer-Verlag, 2000. 13−23.

[2] Kuramochi M, Karypis G. Frequent subgraph discovery. In: Cercone N, Lin TY, Wu X, eds. Proc. of the 1st IEEE Int'l Conf. on Data Mining. San Jose: IEEE Computer Society, 2001. 313−320.

[3] Yan X, Han J. gSpan: Graph-Based substructure pattern mining. In: Aggrawal R, Dittrich K, Ngu AH, eds. Proc. of the 2nd IEEE Int'l Conf. on Data Mining. Maebashi: IEEE Computer Society, 2002. 721−724.

[4] Borgelt C, Berhold MR. Mining molecular fragments: Finding relevant substructures of molecules. In: Aggrawal R, Dittrich K, Ngu AH, eds. Proc. of the 2nd IEEE Int'l Conf. on Data Mining. Maebashi: IEEE Computer Society, 2002. 51−58.

[5] Huan J, Wang W, Prins J. Efficient mining of frequent subgraphs in the presence of isomorphism. In: Wu X, Tuzhilin A, eds. Proc. of the 3rd IEEE Int'l Conf. Data Mining. Melbourne: IEEE Computer Society, 2003. 549−552.

[6] Nijssen S, Kok JN. A quickstart in frequent structure mining can make a difference. In: Kim W, Kohavi R, Gehrke J, DuMouchel W, eds. Proc. of the 10th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Seattle: ACM, 2004. 647−652.

[7] Yan X, Han J. Closegraph: Mining closed frequent graph patterns. In: Getoor L, Senator TE, Domingos P, Faloutsos C, eds. Proc. of the 9th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Washington: ACM, 2003. 286−295.

[8] Huan J, Wang W, Prins J, Yang J. Spin: Mining maximal frequent subgraphs from graph databases. In: Kim W, Kohavi R, Gehrke J, DuMouchel W, eds. Proc. of the 10th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Seattle: ACM, 2004. 581−586.

[9] Thomas LT, Valluri SR, Karlapalem K. Margin: Maximal frequent subgraph mining. In: Proc. of the 6th IEEE Int'l Conf. Data Mining. Hong Kong: IEEE Computer Society, 2006. 1097−1101. http://www.computer.org/portal/web/csdl/doi/10.1109/ICDM.2006.102

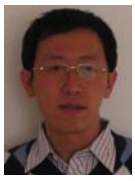
[10] http://dtp.nci.nih.gov/docs/aids/searches/list.html

[11] Liu Y, Li J, Gao H. Summarizing graph patterns. In: Bernstein P, Bertino P, Dayal U, Lockemann P, Weikum G, Whang KY, eds. Proc. of the 24th IEEE Int'l Conf. on Data Mining. Cancun: IEEE Computer Society, 2008. 903−912.

[12] Wang C, Wang W, Pei J, Zhu Y, Shi B. Scalable mining of large disk-based graph databases. In: Kim W, Kohavi R, Gehrke J, DuMouchel W, eds. Proc. of the 10th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Seattle: ACM, 2004. 316−325.

[13] Chen YK. Adaptive parallel graph mining for CMP architectures. In: Proc. of the 6th IEEE Int'l Conf. Data Mining. Hong Kong: IEEE Computer Society, 2006. 97−106. http://www.computer.org/portal/web/csdl/doi/10.1109/ICDM.2006.15

[14] Zeng Z, Wang J. Zhang J, Zhou L. FOGGER: An algorithm for graph generator discovery. In: Kersten ML, Novikov B, Teubner J, Polutin V, Manegold S, eds. Proc. of the 12th Int'l Conf. Extending Database Technology. Saint-Petersburg: ACM, 2009. 517−528.



刘勇(1975－),男,博士生,黑龙江哈尔滨人,主要研究领域为图数据挖掘.

高宏(1966－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据仓库与数据挖掘,传感器网络.

李建中(1950－),男,教授,博士生导师,CCF 高级会员,主要研究领域为统计与科学数据库系统,并行数据库系统,数据仓库与数据挖掘,传感器网络,数据网格.