频繁子图挖掘算法综述*)

王艳辉 吴 斌 王 柏
(北京邮电大学计算机科学与技术学院通信软件工程中心 北京 100876)

摘 要 本文介绍了基于图的频繁子图挖掘算法的研究情况,提出频繁子图挖掘算法的分类方法,对一些经典的算法进行了分析和评价,归纳出频繁子图挖掘的一般步骤以及实现这些步骤的方法,展望了频繁子图挖掘的未来研究方向。

关键词 关联规则,标记图,Canonical code,子图同构

Survey of Frequent Subgraph Mining

WANG Yan-Hui WU Bin WANG Bai
(Telecommunications Software Engineering Group, School of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing 100876)

Abstract This paper provides a survey of the study in frequent subgraph mining, brings forward a classification of frequent subgraph mining, reviews and analyses some typical algorithms, conclude the general steps on finding frequent subgraphs in graph database and the methods which can be applied to the corresponding steps, views some future directions in frequent subgraph mining.

Keywords Association rule, Labeled graph, Canonical code, Subgraph isomorphism

1 引言

关联规则算法首先是由 Agrawal 等人于 1993 年提出的,是基于客户交易数据库,用来解决项集之间关联问题的,后来诸多研究人员对关联规则的挖掘问题进行了大量研究,除了应用于非结构型数据(non-structure)以外,还研究和提出了许多应用于半结构型(semi-structure)数据和结构型(structure)数据的算法。

图形数据广泛存在于我们的生活中,如化学、生物、国防等领域都大量使用到了图形数据,因此,对基于图的频繁子图挖掘算法的研究是非常必要的。基于图的数据挖掘提出时间并不长,但是由于图论作为数学的一个研究领域已经有很长的研究历史,因而频繁子图挖掘的发展很快,并被广泛地应用到许多领域中,如在化学领域中,通过频繁子图挖掘算法找出构成有毒物质的分子结构,以及通过对网站浏览日志的挖掘,分析出最频繁的浏览模式等。

Akihiro Inokuchi 等人最早将 Apriori 算法思想应用到频繁子图挖掘中[1],引起了诸多学者对频繁子图挖掘的注意,各种算法也就应运而生[2,3],最近,韩家炜等人提出了将 FP-growth 思想应用到子图挖掘中[18,5],使得频繁子图挖掘算法得到了迅速发展。后来,许多研究人员,如 Jun Huan 等人提出了 FFSM[8]等基于 FP-growth 思想的算法,使得频繁子图挖掘算法得到了进一步的发展。

2 频繁子图挖掘算法的基本概念

2.1 基本概念和问题描述

有标记图:设 $V(G)=\{v_1,v_2,\cdots v_n\}$,是图 G 的顶点集合,$E(G)=\{e_h=(v_i,v_j)\mid v_i,v_j\in V(G)\}\subseteq V\times V$,,是图 G 的边的集合,顶点标记集合 $L(V(G))=\{lb(v_i)\mid \forall v_i\in V(G)\}$,边的标记集合 $L(E(G))=\{lb(e_h)\mid \forall e_h\in E(G)\}$,则有标记图可以表示为 $G=(V(G),E(G),L(V(G)),L(E(G)))$。

输入数据库 $GD=\{G_1,G_2,\cdots,G_n\}$,其中 $G_i=(V(G_i),E(G_i),L(V(G_i)),L(E(G_i)))$。

频繁子图挖掘(frequent subgraph Mining):假定输入数据库 $GD=\{G_i\mid i=0,1,\cdots,n\}$,给定一个最小支持度阈值 min_sup,规定:如果子图 g 与 G_i 子图同构,则 $o(g,G_i)=1$,否则 $o(g,G_i)=0$,$\delta(g,GD)=\sum_{G\in GD}o(g,G_i)$,如果 $\delta(g,GD)\geqslant$ min_sup,则 g 是一个频繁子图。频繁子图挖掘就是要从输入数据库中找出所有频繁子图。

2.2 频繁子图挖掘的分类

我们将频繁子图挖掘按照不同情况进行分类:

(1)按照输入数据库的类型进行分类:分为 transaction 和 large graph 两种类型。transaction 型挖掘所处理的输入数据库是由许多图构成的,每个图可能只包含几十到几百个顶点;而 large graph 型挖掘所处理的输入数据库有且只有一个大图,这个大图包含成千上万个顶点。

(2)按照采用度量的不同进行分类:分为支持度(support),支持度-置信度,MDL(minimum description length)。支持度型挖掘是以子图在输入数据库中出现的次数来作为度量的,大部分算法都是基于支持度的;MDL 型挖掘是以压缩输入数据库的程度来度量的,一般采用公式 $value(s,g)=dl(g)/(dl(s)+dl(g|s))$来计算,其中 s 是子图,g 是输入数据库,$dl(g)$表示输入数据库 g 的存储空间,$dl(g|s)$表示把 g 中所有出现 s 的地方都用同一个顶点替换后的图形所需的存储空间;支持度-置信度型挖掘是以既要满足最小支持度又要满足最小置信度来衡量的。还有其它一些度量方法,这里就不再介绍了。

(3)按照挖掘出的频繁子图的类型进行分类:分为一般子

*)本课题得到国家自然科学基金(60402011)资助。王艳辉 硕士研究生,主要研究数据挖掘与数据仓库。吴 斌 副教授。王 柏 教授,博士生导师。

图，连通子图，诱导子图，顺序树，无序树等。诱导子图是指，假定图 G，Gs 满足如下条件 $V(Gs)\subset V(G)$，$E(Gs)\subset E(G)$，并且 $\forall u,v\in V(Gs)$，$(u,v)\in E(Gs)$，$\Leftrightarrow(u,v)\in E(G)$，则 Gs 是 G 的诱导子图。

给出了频繁子图挖掘的分类之后，在下面的分析过程中，就可以考虑某个具体的方法用于哪一类子图挖掘，某类子图挖掘又可以用哪些不同的算法进行处理。

3 频繁子图挖掘算法

关联规则挖掘算法中有两种经典算法：Apriori 算法和 FP-growth 算法。频繁子图挖掘算法也是基于这两种思想的，由于 FP-growth 算法的效率要高于 Apriori 算法，因此 FP-growth 算法在频繁子图挖掘中得到了广泛的应用，下面分别介绍这两类算法的思想。

3.1 Apriori 算法在频繁子图挖掘中的应用

Apriori 算法[20]采用了 generation- and- test 思想：k-频繁子集用于生成 $k+1$-子集，根据 downward closure property 性质(如果 $k+1$ 子集的任何一个 k 子集是非频繁的，则 $k+1$ 子集一定也是非频繁的)进行剪枝，生成 $k+1$ 候选集，通过对数据库进行扫描判断候选子集中哪些是频繁的。如此下去，直到不能找到频繁项集。

最早将 Apriori 思想用于频繁子图挖掘的是 AGM 算法[1]，该算法能在 transaction 型输入数据库中挖掘出满足最小支持度和最小置信度的所有频繁诱导子图。

AGM 算法采用邻接矩阵来表示图，并且根据邻接矩阵来生成这个图的 code。由于一个图可以用多个邻接矩阵表示，为了唯一地表示一个图，就将 min(code)作为该图的唯一邻接矩阵，这样可以避免直接进行子图同构的计算。

1)根据 apriori 算法的思想，首先将频繁顶点作为初始集，然后判断 X_k，Y_k(k 表示子图中包含的顶点个数)是否包含同一个 $k-1$ 子图 X_{k-1}，如果包含，则将 X_k，Y_k 中 code 大的作为第二个操作因子，生成候选子图 $Z_{k+1}=X_k+Y_k$。这样通过每次添加一个顶点来生成候选 $k+1$ 子图 Z_{k+1}。

2)根据 apriori 性质：如果 Z_{k+1} 的 k-子图中有一个是非频繁的，那么 Z_{k+1} 也一定非频繁地来进行剪枝。

3)计算候选子图的支持度就是找出候选子图与输入数据库中哪些图是子图同构的关系。由于子图同构是一个 NP 问题，为避免直接计算子图同构，AGM 引入了 min(code)来唯一标识一个图，从而计算出每一个候选子图的支持度。

AGM 算法的思路比较简单，以递归统计为基础，可以挖掘出所有频繁子图。但是对于大型数据库来说执行效率非常低，主要是因为 AGM 在生成候选子图时要判断是否存在相同的 $k-1$ 子图，如果 k 很大时，这需要很长时间。并且通过每次添加一个顶点来产生候选子图时会产生许多冗余 $k+1$ 子图。在剪枝的过程中，也需要很多时间来判断每个 $k+1$ 候选子图的所有 k 子图是否都是频繁的。剪枝后的候选子图仍然很多，因此需要大量的重复扫描数据库来计算候选子图的支持度。这就占用了大量的内存空间和 cpu 处理时间，很难发现长模式的频繁子图，执行效率不高。因此，FSG 算法[10]对 AGM 算法进行了改进，采用的是每次添加一条边，而不是每次添加一个顶点，并加强了候选子图的剪枝，在计算候选子图的支持度时也采取了一定的优化措施，执行效率较 AGM 有所提高。而 AcGM 算法则采用一些限制条件对 AGM 算法进行了改进，算法的具体描述可以参考文[3]。

3.2 FP-growth 算法

FP-growth 算法的主要思想[19,22]是将产生频繁集的数据压缩到一棵频繁模式树 FP-tree 中，用 FP-tree 存储项的关联信息，然后对模式树产生频繁集。gSpan 算法[18]和 FFSM 算法[8]是 FP-growth 思想在频繁子图挖掘中的典型应用，它们都能在 transaction 型输入数据库中挖掘出所有频繁连通子图。下面将详细介绍这两种算法。

3.2.1 gSpan 算法　mini Code：首先对图进行深度优先搜索，从而形成一棵 DFS Tree，然后根据 $<_T$ 来扫描该图，则扫描的边的顺序就构成了一个序列，称为 DFS Code，不同的 DFS Code 之间按照字典顺序排序。对这个图的所有 DFS Code 进行排序，找出最小的 DFS Code 来唯一标识这个图，这个最小的 DFS Code 被称为 mini Code。

空间树：将图的 mini Code 作为一个空间树的节点，对该节点进行最右路径扩展，把生成的新子图作为该节点的孩子，同一父节点的不同子节点之间是按照字典顺序排序的，这样就生成了一棵空间树。如果将该空间树的所有非频繁子节点剪枝就形成了一棵频繁子图树。

Occurrence：如果子图 g 与输入数据库中图 G 具有子图同构关系，则 G 被称为 g 的 occurrence。

算法的基本思想：

1)计算所有边的支持度，将所有非频繁边从输入数据库中删除，并以频繁边作为初始子图。

2)产生候选子图：对 k 频繁子图的 mini Code 的 DFS Tree 进行最右路径扩展，每次添加一条边，得到 $k+1$ 候选子图。该方法经证明能够保证挖掘结果的完整性。

3)剪枝：如果 $k+1$ 候选子图不是 mini Code 形式的，则认为该图是冗余的，从候选子图中删除。

4)每次在计算 k 频繁子图的支持度的时候，同时记录 k-频繁子图的所有 occurrence。这样，$k+1$ 候选子图的支持度就可以通过对 k 频繁子图的所有 occurrence 进行最右路径扩展获得。

5)缩减数据库：当某条频繁边的所有子节点都生成后，就将该边从输入数据库中删除，缩减输入数据库。

该算法的核心思想如下：

```
Subgraph_Mining(GS,S,s)
1. if s! =min(s)
2. return ;
3. S←SU{s};
4. generate all s' potential children with one edge growth;
5. Enumerate(s);
6. for each c,c is s'child do
7.    if(support≥min_sup)
8.      s←c;
9. Subgraph_Mining(GS,S,s);
```

该算法在生成候选子图的时候，由于采用了最右路径扩展以及相应的剪枝措施，大大减少了冗余候选子图的产生，在计算候选子图支持度的时候采用了对 occurrence 进行扫描，而不是对输入数据库进行扫描，避免了大量重复扫描数据库的问题。但是，对于大型输入数据库，该算法效率仍然很低。

3.2.2 基于 gSpan 算法的频繁闭子图挖掘算法 CloseGraph　基于图的频繁闭子图挖掘算法有许多种，有基于 AGM 的，也有基于 gSpan 算法。这里简单地介绍一下基于 gSpan 算法的频繁闭子图挖掘算法——CloseGraph 算法[17]。

闭图(closed graph)：假定图 g，如果输入数据库中不存在图 G，使得 G 是 g 的超图，且 G 的支持度等于 g，则图 g 称为闭图。

频繁闭子图挖掘：是指在输入数据库中找出所有支持度不小于最小支持度的闭子图。

Equivalent Occurrence：假定图 g，图 $g'=g\diamond_x e$，$I(g,D)$ 表示 g 在输入数据库 D 中具有子图同构关系的所有图的数

量，$L(g,g',D)$ 表示 g' 在输入数据库 D 中具有子图同构关系的所有图的数量，如果 $I(g,D)=L(g,g',D)$，那么，称 g' 与 g 具有 Equivalent Occurrence 关系。

定理 1(early termination) 假定图 $g'=g\diamond_x e$，D 表示输入数据库，$I(g,D)=L(g,g',D)$，如果 $\forall h$，$g\sqsubset h$，$g'\not\sqsubset h$，$I(g,D)=L(g,h\diamond_{xf}e,D)$ 或 $I(g,D)=L(g,h\diamond_{xb}e,D)$，那么可以只对 g' 进行扩展，而不需要对 g 进行其他边的扩展。

CloseGraph 算法与 gSpan 算法的思想是一致的，只是在进行剪枝的过程加了一个判断条件，判断定理 1 是否成立，如果成立，则只对 g' 进行扩展，不需要再对 g 进行最右路径扩展。

其核心思想如下：

```
CloseGraph(s,p,D,min_sup,S)
Input: A DFS code s, its parent p, a graph database D and min_sup.
Output: the closed frequent graph set S.
1: if s≠min(s), then
2:    return;
3: if ∃e', g' = g_p◇_x e' and g'<g, and I(g_p,D)= L(g_p,g',D) and
      g_p is not a failure case of early termination then
4: return;
5: set C to ϕ;
6: scan D once ,find every edge e such that s can be extended to fre-
   quent s◇_x e into C;
7: detect any possible failure of early termination in s;
8: if not s◇_x e∈C, support(s)= support (s◇_x e) then insert s into
   S;
9: remove s◇_x e from C which cannot be right-most extended from s;
10: sort C in DFS lexicographic order;
11: for each s◇_x e in C do
12: call CloseGraph(s◇_r e,s,D,min_sup,S);
13: return;
```

该算法是基于 gSpan 算法的，它具有 gSpan 的所有优点，但是它仅是用来挖掘频繁闭子图的，因此效率要比 gSpan 高。

3.2.3 FFSM 算法 CAM：使用邻接矩阵表示图，按照从上到下，从左到右的顺序扫描邻接矩阵的下三角(包括对角线)，将得到的边的序列称为图的 code，mini(code)称为图的 canonical code，相应的邻接矩阵称为图的 CAM(Canonical Adjacency matrix)。

CAM Tree：规定空节点作为 CAM Tree 的根结点，将图的 CAM 做为树的节点，每个子节点都比其父节点多一条边。

算法的基本思想：

1)根据 FFSM-Join 产生候选子图：规定如果图 A 的 CAM 中最后一行至少包含 2 条边，则称图 A 为 inner 类型，否则称为 outer 类型。FFSM-Join 根据 k-频繁子图 A 和 B 所属类型的不同，采用不同的方式进行合并，生成 $k+1$ 候选子图，其核心思想如下：

```
join case 1: both A and B are inner matrices
1:if f≠k then
2:join (A,B)={C} where C is a m×m matrix and
   c_{i,j} = { a_{i,j}  0<i,j≤m
             { b_{i,j}  otherwise
3: else
4: join(A,B)=ϕ
5: end if
join case 2:A is an inner matrix and B is an outer matrix
join(A,B)={C} where C is a n×n matrix and
c_{i,j} = { a_{i,j}  0<i,j≤m
          { b_{i,j}  otherwise
join case 3:both A and B are outer matrices
1:let matrix D be a (m+1) (m+1) matrix where (case 3b)
   d_{i,j} = { a_{i,j}  0<i,j≤m
             { b_{m,j}  i=m+1,0<j<m
             { 0        i=m+1,j=m
             { b_{m,m}  i=m+1,j=m+1
2: if (f≠k∧a_{m,m}=b_{m,m}) then
3:C is m×m matrix where (case 3a)
   c_{i,j} = { a_{i,j}  0<i,j≤m,i≠n∧j≠k
             { b_{i,j}  otherwise
4:join (A,B)={C,D}
5:else
6:   join(A,B)={D}
7: end if
```

2)FFSM-Join 产生的候选子图不能保证挖掘结果的完整性，因此必须采用 FFSM-Extension 进一步产生候选子图，FFSM-Extension 算法每次添加一个顶点。其核心思想如下：

```
input:a n×n adjacency matrix A
output:a set S of adjacency matrices B(with size (n+1)) s.t. A is B
's maximal proper submatrix.
1: if (A is an outer adjacency matrix) then
2: for (n_l,e_l)∈ΣV×ΣE do
3:      create an n×n matrix B s.t.
4: b_{i,j} = { a_{i,j}  0<i,j≤n
             { 0        i=n+1,0<j<n
             { e_l      i=n+1,j=n
             { n_l      i=n+1,j=n+1
5:       S←S∪{B}
6:   end for
7: else
8:  S←ϕ
9: end if
```

3)通过 FFSM-Join 和 FFSM-Extension 两种方法，能够保证挖掘结果的完整性。通过将非频繁子图以及如果候选子图的邻接矩阵不是 suboptimal 形式的剪枝来减小候选子图的数量。

4)输入数据库中，与 k-频繁子图具有子图同构关系的所有图，称为 embedding set。$k+1$ 候选子图的支持度，可以通过扫描 embedding set 获得，避免重复扫描整个输入数据库，从而提高了计算支持度的速度。

FFSM 算法的核心思想如下：

FFSM-Explore

```
Input :U,a suboptimal CAM list and W,a set of frequent connected
subgraphs' CAMs
Output:set W contains CAMs of all frequent subgraphs searched so
far.
1: for X∈P do
2:   if(X. isCAM) then
3:       W←W∪{X},C←ϕ
4:       for Y ∈P do
5:           C←C∪FFSM-Join(X,Y)
6:       end for
7:       C←C∪FFSM-Extension(X,Y)
8:       remove CAM(s) from C that is either infrequent or not sub-
         optimal
9:       FFSM-Explore(C,W)
10:   end if
11: end for
```

FFSM 算法通过定义 canonical code 唯一标识图，避免了直接计算子图同构问题。使用 FFSM-Join 和 FFSM-Extension 来扩展频繁子图，并通过相应的剪枝策略来获得候选子图。在计算支持度时，只对 embedding set 进行扫描，提高了计算的速度和效率。

3.3 其他算法

3.3.1 基于图的频繁树挖掘算法 频繁子树挖掘算法已经发展了很长时间，但是基于图的频繁子树挖掘算法并不是很多。这里介绍一种在无环标记图中挖掘出无序，无根的树型结构图(称为 free tree)的算法 FreeTreeMiner[16]，该算法的基本思想是：

1)首先确定一个图的中心，然后将这个中心作为根节点。这样我们就将一个无环图转换为一棵有根的无序树。然后我们将这个无序树转换成一棵有序树。通过以上两个步骤，将一个无环图转换成为一棵有序有根的树。

2)然后选取输入数据库中频繁顶点作为初始子树进行扩展产生候选子树。其扩展方法如下：首先，规定离根节点最远的叶子节点被标记为可扩展点。每次通过对该可扩展点进行扩展获得候选子树。

3)为了不产生冗余的候选子树，FreeTreeMiner 算法每次扫描输入数据时，除了计算候选子树的支持度外，还将所有与可扩展点关联的边以及边的支持度记录到一个 extension table 中。通过扫描 extension table 中的内容，选取所有频繁的可扩展边进行扩展，获得频繁子树。

FreeTreeMiner算法采用了深度优先搜索的策略。并使用extension table记录可扩展边以及可扩展边的支持度，避免冗余子树的生成，提高了挖掘的速度。

3.3.2 *非精确算法* SeuS算法[14]是最早提出的large graph型频繁子图挖掘算法，该算法不能求出所有频繁子图，是一个不精确的算法。该算法的基本思想是：首先，对输入数据库进行summarize；然后，在挖掘的过程中记录下k-频繁子图的所有可扩展边，每次只对这些可扩展边进行扩展，减少了候选子图的产生。由于SeuS算法不是直接对源数据库进行挖掘，而是对summarize后的数据库进行挖掘，因此说该算法是不精确的，该算法采用了启发式的思想。

SUBDUE[9]是一种基于MDL度量的不精确算法。它既可以用于transaction型数据库，也可以用于large graph型数据库。该算法采用了贪心策略，首先对输入数据库进行计算，得出输入数据库所需的存储空间；然后，根据公式 *matchcost* (g_1,g_2)<用户指定值来计算出子图g_1的所有instance；在g_1的所有instance中选取一个能使MDL值最大，并大于给定值的子图作为结果；最后，在计算候选子图时，对上面得到的结果添加一条边，重复上面的过程。SUBDUE算法选取MDL作为度量，并可以根据background knowledge等知识对MDL的计算方法进行修改，从而可以灵活地应用到实际问题中，特别是那些不要求精确计算支持度的领域，比如社会人际关系网络图这种本身定义就很模糊的领域。

3.3.3 *large graph型频繁子图挖掘算法* large graph型频繁子图挖掘算法与transaction型频繁子图挖掘算法在实现上有所差异，主要表现在large graph型挖掘算法的输入数据库是一张大图，而不是由许多小图构成的。因此，计算支持度的时候，不能像transaction型挖掘算法那样，只要候选子图与某个图具有子图同构关系，支持度就增加一次，而是要在这个大图中找出所有的子图同构，来计算候选子图的支持度。

最近Michihiro等人提出的HSIGRAM和VSIGRAM两种算法是用来挖掘出large graph型输入数据库中的所有频繁子图。该算法除了考虑子图同构问题外，还考虑了同一个图的不同子图同构之间是否有overlap问题，是在大图中寻找频繁子图很有效的算法。HSIGRAM的具体算法可参考文[12]。

3.3.4 *其他算法* Akihiro Inokuchi等人提出了能够在有向图和无向图中挖掘出所有频繁诱导子图的算法，算法的具体描述可以参考文[4]。还有一些算法，如SPIN算法是对FFSM算法进行了改进，用来挖掘最频繁的n个子图问题，具体可以参考文[7]。还有针对其他问题的，比如寻找结构相似的频繁子图挖掘问题，用户自定义约束模式挖掘问题等等。

4 频繁子图挖掘算法的问题及分析

频繁子图挖掘算法的两个性能瓶颈：一是产生候选子图，二是计算候选子图的支持度。要解决第一部分，主要就是快速地生成候选子图，不要产生冗余的候选子图；第二部分就是要解决子图同构问题，而子图同构是一个NP问题，因此必须避免或者简化子图同构问题。

从上面所论述的算法以及其他算法的研究过程中，我们归纳出基于图的频繁子图挖掘算法的主要步骤，以及每个步骤可以采用哪种方法进行解决。

(1)唯一标识一个图(Canonical code)：唯一标识一个图可以用来简化子图同构问题，从而可以方便地判断冗余子图，并快速地计算出候选子图的支持度。有以下几种方法来唯一标识一个图：DFS Code、邻接矩阵、Tree code symbols。当然除了列出的方法外，还可以采用其他的方法来对图进行唯一标识。选取何种方法对图进行标识，取决于算法。对图进行唯一标识是解决子图挖掘的一个最重要的步骤。

(2)使用简化措施来简化输入数据库或者对子图同构问题进行简化定义，来避免子图同构问题。这个步骤是可选的。

(3)生成候选子图，可以采用如下方法：每次添加一个顶点，可能会引入一条或者多条边；每次添加一条边，有两种情况：引入一条边和一个新的顶点；只引入了一条边，将子图中的两个顶点进行连接。将每次选取的边或顶点添加到子图中去，可以采用以下几种方法：

- level-wise search：通过合并两个具有相同$k-1$子图的k子图，生成$k+1$子图，因为每次都要计算所有的k-子图是否含有相同的$k-1$子图，非常浪费时间，而且这样产生的$k+1$子图也不一定在输入数据库中存在。因此该方法效率不高。
- 贪心方法：该方法可以用在不精确的算法中。
- 深度优先方法：该方法在上面介绍的gSpan等算法中可以看到。
- inductive logic programming：该方法比较适合小型输入数据库。

(5)对生成的候选子图进行剪枝：无论采用何种算法都可能会产生冗余候选子图，而计算候选子图的支持度需要多次扫描数据库，非常浪费时间，因此必须对生成的候选子图进行剪枝。有以下几种方法用于剪枝：

- 如果k子图中有一个$k-1$子图是非频繁的，则该k子图也一定是非频繁的，可以剪掉。
- 如果k-子图不是Canonical Code，则认为该子图是冗余的，可以剪掉。
- 可以只考虑那些在输入数据库中与该子图有关联的边(称为可扩展边)生成的子图，如果生成的图中不包含这样的可扩展边，则可以剪掉。

(6)计算支持度：计算候选子图支持的问题在采用了Canonical code来唯一标识一个图后，就是寻找与该子图的Canonical Code相同的图的个数问题。为避免对输入数据库进行多次全局扫描，可以采用如下的措施：

- embedding list (el)：对于只有一个顶点的图，存储输入数据库中出现该顶点的所有图的编号。对于多顶点子图，存储输入数据库中所有出现该子图的图的编号以及该子图在这些图中的位置。这种方法计算起来速度快，但是要占用大量的存储空间，不适合大型输入数据库。
- recomputed embedding：每次记录下出现该频繁子图的那些图的编号，通过扫描出现过k频繁子图的那些图来计算$k+1$候选子图的支持度，这样每次只记录图的编号，不用记录子图出现的位置，通过重复计算来获得$k+1$子图的支持度，从而避免了全局扫描数据库，是一个很有效的方法。可以用来处理大型数据库，但是速度不如第一种方法。

结论与展望 本文讨论了基于图的频繁子图挖掘的各种算法及其应用，目前在这方面的研究已经取得了很大成绩，并被应用到一些实际的系统中。本文重点介绍了基于Apriori思想和FP-growth思想的频繁子图挖掘算法，并对其他算法进行了简要介绍。归纳出频繁子图挖掘算法的一般步骤和可以采用的解决方法。

虽然目前存在的算法很多，且执行效率很高，但是在处理

(下转封四)

(上接第 196 页)

大型数据库时仍然需要大量的时间和空间,因此我们觉得可以在下面一些方向上做进一步的研究。在处理大型数据库时,如何采用并行处理和分布式处理;求不精确的频繁子图问题在现实生活中广泛存在,但是目前的算法还停留在最初提出的 SUBDUE 上,有待进一步提高;在 large graph 型数据库中进行挖掘的算法也比较少,目前主要都是基于 transaction 型数据库的,因此对 large graph 型数据库的频繁子图挖掘算法的研究也是一个重要的研究方面;现实生活中有各种各样的图形:有向图,无向图,加权图,无连通图等,但目前的算法大部分都是针对连通图的挖掘,对加权图等的挖掘算法很少,因此对加权图等的挖掘算法的研究也是一个重要的研究方向;挖掘结果如何可视化等。

参 考 文 献

1 Inokuchi A, Washio T, Motoda H. An apriori-based algorithm for mining frequent substructures from graph data. In: Proc. of the 4th European Conf. on Principles and Practice of Data Mining and Knowledge Discovery (PKDD-2000), 2000. Proc. of the 8th ACM SIGKDD intl. conf. on Knowledge discovery and data mining. ACM press, 2002. 13～23

2 Inokuchi A, Washio T, Motoda H. Applying the Apriori-based Graph Mining Method to Mutagenesis Data Analysis. Journal of Computer Aided Chemistry, 2001, 2: 87～92

3 Inokuchi A, Nishimura T K, Motoda H. A Fast Algorithm for Mining Frequent Connected Subgraph IBM Research. Tokyo Research Laboratory, 2002

4 Inokuchi A, Motoda T H. Complette Mining of Frequent Patterns from Graphs: Mining Graph Data. In: Machine Learning, 2003. 321～354

5 Han Jia-Wei, Pei Jian, Yan Xi-Feng. From Sequential Pattern Mining to Structured Pattern Mining: A Pattern-Growth Approach. Journal of Computer Science and Technology, 2004, 19 (3): 257

6 Han Jia-Wei, Wang Jianyong, Lu Ying. Petre Tzvetkov. Minging Top-K Frequent Closed Patterns without Minimum Support. http: // 140.116.247.141/～tsengsm/COURSE/DM/Paper/DM-PAPER-LIST-04.HTM

7 Huan Jun, Wang Wei, Prins Jan, Yang Jiong. SPIN: Mining Maximal Frequent Subgraphs from Graph Databases: [UNC Technical Report TR04-018]. 2004

8 Huan Jun, Wang Wei, Prins Jan. Efficient Mining of Frequent Subgraph in the presence of Isomorphims. In: Proc. of 2003 IEEE Intl. Conf. on Data Mining (ICDM'03), 2003

9 Holder L, Cook D, Djoko S. Substructure discovery in the SUBDUE system. In: Proc. of the Workshop on Knowledge Discovery in Databases, 1994. 169～180

10 Kuramochi M, Karypis G. Frequent Subugraph discovery. In: Proc. 2001 Int. Conf. Data Mining (ICDM'01), San Jose, CS, Nov. 2001. 313～320

11 Koyuturk M, Grama A, Szpankowski W. An Efficient Algorithm for Detecting Frequent Subgraphs in Biological Networks BioInformatics Suppl. In: 12th Intl. Conf. Intelligent Systems Molecular Biology (ISMB'04) vol. 20, pp. i200a～i207

12 Kuramochi M. George Karypis. Finding Frequent Patterns in a Large Sparse Graph. In: Proc. of the 2004 SIAM Data Mining Conf. 2004

13 Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. In Proc. of the 1993 ACM SIGMOD Intl. Conf. on Management of Data, P. Buneman and S. Jajodia, Eds. Washington, D.C., USA: ACM Press, 1993. 207～216

14 Ghazizadeh S, Chawathe S. SeuS: Structure Extraction using Summaries. In: Proc. of the 5th Intl. Conf. on Discovery Science, 2002

15 Asai T, Abe K, et al. Efficient SubStructure Discovery from Large Semi-structured Data. In SIAM SDM'02, April 2002

16 Ruckert U, Kramer S. Frequent Free Tree Discovery in Graph Data, https: // portal.acm.org /poplogin.cfm? dl = ACM&-coll = GUIDE&-comp_id = 968018&-want_href = delivery% 2Ecfm% 3Fid% 3D968018% 26type% 3Dpdf&-CFID = 33435631&-CFTOKEN=519350&-td=1102493452296

17 Yan Xifeng, Han Jiawei. CloseGraph: Mining Closed Frequent Graph Patters. In: Proc. of the ninth ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining. ACM Press, Aug. 2003. 286～295

18 Yan Xifeng, Han Jiawei. gspan: Graph-based substructure pattern mining: [Technical Report UIUCDCS- R-2002-2296]. Department of Computer Science, University of Illinois at Urbana-Champaign, 2002

19 陈安龙,唐常杰,陶宏才,元昌安,谢方军. 基于极大团和 FP-Tree 的挖掘关联规则的改进算法. 软件学报, 2004, 15 (8): 1198～1207

20 李力,翟东海,靳蕾. 基于图的频繁闭项集挖掘算法. 西南交通大学学报, 2004, 39(3): 385～389

21 楼宇波,马坚,周皓峰,袁晴晴,施伯乐. 基于频繁链接的 Web 权威资源挖掘. 计算机研究与发展, July 2003, 40(7): 1095～1103

22 王新宇,杜孝平,谢昆清. FP-growth 算法的实现方法研究. 计算机工程与应用, Sep. 2004. 174～176

计算机科学

(1974 年 1 月创刊)

第 32 卷第 10 期 (月刊)

2005 年 10 月 25 日出版

ISSN 1002－137X

CN50—1075/TP

定价: 25.00 元　国外定价: 5 美元

邮发代号: 78—68

发行范围: 国内外公开

主管单位: 国家科学技术部

主办单位: 国家科技部西南信息中心

编辑出版: 《计算机科学》杂志社

重庆市渝中区胜利路 132 号　邮政编码: 400013

电话: (023) 63500828　E-mail: jsjkx@swic.ac.cn

网址: www.jsjkx.com

社　长: 牟炳林

主　编: 彭　丹

副 主 编: 朱宗元

主编助理: 徐书令

印 刷 者: 重庆科情印务有限公司

总发行处: 重庆市邮政局

订 购 处: 全国各地邮政局

国外总发行: 中国国际图书贸易总公司 (北京 399 信箱)

国外代号: 6210—MO