频繁子图数据挖掘研究综述

钱炘祺*

(南京大学 计算机科学与技术系，南京 210093)

A Survey of Algorithms for Mining Frequent Substructure in Graph Data

XinQi Qian*

(Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

Abstract: With the development of subjects including bioinformatics, cheminformatics and web log mining video indexing, the demand of the development of data mining has become more and more urgent. Graph data mining (graph mining for short), an extremely important branch of data mining, has been developing at high speed for decades. Problem of graph mining can be classified into 3 parts: graph clustering, graph classification and frequent subgraph mining. In this paper, we investigate some typical frequent subgraph mining algorithms of different stages throughout its development history. New progress and future research directions are pointed at the end of the paper.

Key words: graph mining; frequent subgraph; frequent substructure; SUBDUE; ILP; AGM; FSG; gSpan; FFSM;

摘 要: 生物信息学，化学信息学等近年来的蓬勃发展加大了对数据挖掘技术发展的需求。图数据挖掘（以下简称为图挖掘），作为数据挖掘中一个相当重要的分支，也有了相当的发展。图挖掘所涉及的领域主要是图数据聚类，图数据分类和频繁子图（子结构）挖掘等。本文着重对频繁子图挖掘领域的发展不同阶段的代表性算法进行了介绍和比较，并讨论了图挖掘的最新进展及未来的研究方向。

关键词: 图挖掘;频繁子图;频繁子结构; SUBDUE; ILP; AGM; FSG; gSpan; FFSM

中图法分类号: TP301 **文献标识码**: A

1 引言

近年来，数据挖掘领域的发展越来越快。作为其中的一个重要分支，图挖掘领域也受到人们极大的关注，正不断地寻求创新以及更优的算法。图的应用领域非常广泛。无论是在网络拓扑，语义网络这样的人工图据中，还是在化合物分子结构，生物基因表达这样的真实世界数据中，图都是应用相当普遍的一种数据表达形式。在这些数据中寻找有效信息，就是图挖掘所做的工作。

在目前的研究中，图挖掘主要涉及了以下几个方面的研究[1]：

1）图数据的聚类，即在预先不确定划分标准的情况下，将图数据聚类成簇的一种方法。其满足聚类后，簇内部边联系紧密，而簇间则相对稀疏。

2）图数据的分类，即按照某种预先确定的划分标准，将图数据集中零散的图划分到诸多不同类别中去的一种方法。

3）频繁子图挖掘，即寻找在图数据集中频繁出现在各个图数据中的子结构。其广泛应用于生物化学领域。

* 作者简介:钱炘祺 南京大学计算机科学与技术系 2011 级本科生 学号 111220090

例如寻找频繁出现分子结构问题。

本文余下部分将先对图的基本概念做一个梳理，然后着重在频繁子图挖掘部分介绍几个有时代发展代表性的算法。最后做出相关比较和总结，并讨论图挖掘的最新进展和未来的研究方向。

2 图的基础理论

对于图$G=(V,E)$，V表示图中顶点的集合，一般用$n=|V|$表示图的顶点规模；而E表示边的集合，相应的$m=|E|$表示边的规模。图G的密度$\delta(G)$是指其边的个数与最大可能的边个数之比，即$\delta(G)=\frac{m}{C_n^2}$。如果一个图$G$的密度$\delta(G)=1$，则称图$G$为完全图。

对于给定图$G=(V,E)$，其子图$G_s=(V_s,E_s)$满足$V_s\subseteq V,E_s\subseteq E$且$\forall\{u,v\}\in E_s,u,v\in V_s$。当且仅当$G_s$是$G$的子图，且$G_s$为完全图时，称$G_s$是$G$的诱导子图。若$G_s$是$G$的无环子图，则称之为$G$的生成树。

两个图$G_i=(V_i,E_i)$和$G_j=(V_j,E_j)$是同构的，当且仅当存在一个V_i到V_j的一一映射$F:V_i\to V_j$，使得E_i中每一条边$\forall\{u,v\}\in E_i$，存在唯一$\{f(u),f(v)\}\in E_j$。

3 频繁子图挖掘算法

3.1 基于贪心的方法——$SUBDUE$算法[2]

基于贪心策略的频繁子图挖掘是频繁子图挖掘领域最先发展起来的技术之一，其中最著名的是$SUBDUE$算法。这里以$SUBDUE$算法为例，介绍基于贪心策略的频繁子图挖掘。

$SUBDUE$算法基于最小描述长度原则（minimum description length, MDL）来发现子结构。严格来说，其所谓的频繁与之后我们介绍的频繁子图的概念有所不同。其所指的某个子结构“频繁”是用MDL原则评定的，而不是单纯指其出现的频率高：

设S为图G的一个子结构，则$(G|S)$表示在图G中使用单个顶点替换子结构S后得到的图数据。$I(S)$表示子结构S的描述长度，同样$I(G|S)$即表示在图G中使用单个顶点替换子结构S后得到的图的描述长度。由此，$SUBDUE$算法定义了“频繁子图”，即最优子结构：能使$I(S)+I(G|S)$最小的子结构S。

图描述长度$I(S)$的计算可分为三个部分，即：$I(S)=v\ bits+r\ bits+e\ bits$

其中$v\ bits$表示编码图的顶点标签所需的 bit 位数：

$$v\ bits=\lg v+v\lg l_u$$

$r\ bits$表示编码邻接矩阵A的各行所需的 bit 位数：

$$r\ bits=\lg(b+1)+\sum_{i=1}^{v}\lg(b+1)+\lg C(v,k_i)$$

其中$b=max_ik_i$，k_i表示邻接矩阵A第i行中为 1 的个数。

$e\ bits$表示编码邻接矩阵A中以$A[i,j]=1$所表示的边所需的 bit 位数。

$$e\ bits=\lg m+\sum_{i=1}^{v}\sum_{j=1}^{v}\lg m+e(i,j)[1+\lg l_u]$$

$SUBDUE$算法是计算驱动的定向搜索。其初始枚举单节点最优子结构，然后由此进行迭代。在每次迭代中，算法选择最优子结构后，通过增加一个顶点（即一条与已有结点相邻的边）扩展当前子结构，直到所有的最优子结构都被列出或者所有需要考虑的候选最优子结构数量到达一定限制。

$SUBDUE$算法的优点在于其一般生成较少数量且能够最好的压缩图数据集的子结构。

3.2 基于ILP的方法

对于许多挖掘和学习问题来说，描述关系通常比描述属性更能得出简洁且精确的规则。与之相对应的，一阶谓词逻辑在很多情况下的表现也优于命题逻辑。[3]而且，图可以比较容易的使用一阶逻辑来表示。对基于ILP的方法而言，其优点不局限于发现知识，还可以在知识归纳中运用正反例。或者说，其目标是归纳出一个

可正确分类的正样本集和负样本集的规则集。是故，归纳逻辑编程（Inductive Logic Programming，ILP）被提出而应用在频繁子图的搜索上。其优点在于大多算法能找出出现频率高的子图，且能作为交好的类识别器。但其缺点在于不能保证发现所有的频繁子图。

1998 年 Dehaspe 基于ILP提出可对频繁子图进行完全挖掘的$WARMR$算法，其算法核心思想与$Apriori$算法类似。[4]

3.3 基于$Apriori$的方法——AGM算法

比较遗憾的是，基于ILP的算法由于需要图结构对一些特别的特征和谓词实例进行预特征化，故仅能发现有限特征的子结构，无法发现所有的频繁子图。而且虽然其在后续发展中结合了分层搜索以最小化对数据库的访问，效率有了极大的提升，但是，其搜索所需空间仍就很大。由此 A. Inokuchi 等人最先提出了基于$Apriori$思想的频繁子结构挖掘算法（AGM算法）[5]。

AGM算法将图用邻接矩阵X_k表示为$G(X_k)$，满足按照顶点的标签来排序顶点在邻接矩阵中顺序：

$$num\big(lb(v_i)\big) \le num\big(lb(v_{i+1})\big)\ for\ i = 1,2,\dots k-1$$

同时为每个邻接矩阵表示的图都建立图的标记形式$code(X_k)$。对一个已顶点排序的邻接矩阵X_k，

$$X_k = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & & x_{1,k} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,k} \\ x_{3,1} & x_{3,2} & x_{3,3} & & x_{3,k} \\ & \vdots & & \ddots & \vdots \\ x_{k,1} & x_{k,2} & x_{k,3} & \cdots & x_{k,k} \end{pmatrix}$$

若其为无向图，则其编码标记形式记为：

$$code(X_k) = x_{1,1}x_{1,2}x_{2,2}x_{1,3}x_{2,3}x_{3,3}x_{1,4}\dots x_{k-1,k}x_{k,k}$$

若是有向图，则在每个之$x_{i,j}$后加上其对应的$x_{j,i}$，如下：

$$code(X_k) = x_{1,1}x_{1,2}x_{2,1}x_{2,2}x_{1,3}x_{3,1}x_{2,3}x_{3,2}x_{3,3}\dots x_{k-1,k}x_{k,k-1}x_{k,k}$$

该算法通过计算$sup(G_s)$，即图数据集GD中包含G_s子结构的图G的个数与图数据集GD中图G的总个数之比，其是否超过阈值$minsup$来决定G_s是否是一个频繁子结构：

$$sup(G_s) = \frac{number\ of\ G\ where\ G_s \subset G \in GD}{total\ number\ of\ G \in GD}$$

类似于$Apriori$算法，频繁诱导子图的候选集依据子图的大小分层搜索所得。例如，若X_k和Y_k都是 k阶顶点排序的邻接矩阵，若X_k和Y_k仅第k行和第k列的数据不同，即包含一个$k-1$阶子图，则可由之构造Z_{k+1}：

$$X_k = \begin{pmatrix} X_{k-1} & x_1 \\ x_2^T & x_{k,k} \end{pmatrix},\ Y_k = \begin{pmatrix} X_{k-1} & y_1 \\ y_2^T & y_{k,k} \end{pmatrix}$$

$$Z_{k+1} = \begin{pmatrix} X_{k-1} & x_1 & y_1 \\ x_2^T & x_{k,k} & z_{k,k+1} \\ y_2^T & z_{k+1,k} & y_{k,k} \end{pmatrix} = \begin{pmatrix} X_k & y_1 \\ y_2^T & z_{k+1,k} \quad y_{k,k} \end{pmatrix}$$

并将X_k称为第一矩阵，将Y_k称为第二矩阵。为减少出现重复无用的构造，使Z_{k+1}的构造满足：

$$code(X_k) \le code(Y_k)$$

并将之称为标准形式。

同时，该文定义了图邻接矩阵的规范形式X_c，即对同一图G的所有标准形式的邻接矩阵集$NF(G)$中具有最小$code$的那一个邻接矩阵：

$$X_c = \arg\min_{X \in NF(G)} code(X)$$

是故，对$G(X_k)$删去任意一个顶点v_m得到X_{k-1}^m，对其使用T_{k-1}^m变形矩阵进行标准化得到标准的邻接矩阵X'^m_{k-1}，再使用S_{k-1}矩阵，将其变形为规范形式（记对应于X'^m_{k-1}的规范化变形矩阵为S_{k-1}^m），即：

$$(T_{k-1}^m S_{k-1}^m)^T X_{k-1}^m T_{k-1}^m S_{k-1}^m$$

其中变形矩阵的求法如下：

$$s_{ij} = \begin{cases} s_{ij}^{m} & 0 \le i \le k-1 \; and \; 0 \le j \le k-1, \\ 1 & i = k \; and \; j = k, \\ 0 & otherwise, \end{cases}$$

$$t_{ij} = \begin{cases} t_{ij}^{m} & i < m \; and \; j \ne k, \\ t_{i-1,j}^{m} & i > m \; and \; j \ne k, \\ 1 & i = m \; and \; j = k, \\ 0 & otherwise, \end{cases}$$

$$X_{ck} = \arg \min_{m=1,\dots,k} code((T_{k-1}^{m} S_{k-1}^{m})^{T} X_{k-1}^{m} T_{k-1}^{m} S_{k-1}^{m})$$

其中，s_{ij}，s_{ij}^{m}，t_{ij}，t_{ij}^{m}表示矩阵S_k^m，S_{k-1}^m，T_k^m，T_{k-1}^m的元素。而能使得X_{ck}最小的$T_{k-1}^m S_{k-1}^m$即为X_k的S_k变形矩阵。

显然，由于一个$k+1$阶图是频繁子结构当且仅当其所有诱导子图是频繁子结构。故若$G(X_k)$通过删去一个任意一个顶点$v_m(1 \le m \le k)$所得到的$k-1$阶子图$G(X_{k-1})$都是频繁子结构的话，则$G(X_k)$被看作是候选频繁子结构。

在得到所有的候选频繁子结构之后，扫描数据集，对图数据集中的每一个图G的X_k都要标准化来检测候选频繁子结构是否在该图G中。而这个过程会得到图G每个诱导子图的标准形式。是故，频繁度的计算是基于所有图G的诱导子图的标准形式的。

该文所提出的AGM算法在效率上与采用了分层搜索的ILP算法架构相比有了比较大的提升，而且在化学分子结构这样的真实世界图数据中也有不错的表现。但是其在判断模式图X和Y是否具有相同子图时，仍要花费较多时间。且每添加一个顶点产生新候选频繁子结构时，会产生许多冗余的$k+1$顶点的子结构。

3.4 对$Apriori$方法的改进——FSG算法[6]

FSG是AGM算法的一种改进。同基于$Apriori$的方法一样，其采用了分级扩展的方法。但优化之处在于：

- 其采用了相对稀疏的图表示方法，来最小化存储空间和计算开销。
- 每次添加一条边来扩大频繁子图的大小，由此使得生成候选集更有效。
- 采用了对小图更有效的规范标签和图同构算法。
- 其对生成候选集进行了各种优化并统计了能适用于大规模的图数据库的优化措施。

其具体的算法思想如下：

首先，枚举出所有的频繁 1 阶和 2 阶子图。然后以此为源，迭代计算。

在每次迭代中，首先生成比上次迭代出的频繁子图多一条边的候选子图。然后计算每个候选子图的频繁度，除去不到阈值的，将满足阈值的加入频繁子图集。伪码如下：

Algorithm of $fsg(D,\sigma)$ (Frequent Subgraph)

```
1   F^1 ← detect all frequent 1-subgraphs in D
2   F^2 ← detect all frequent 2-subgraphs in D
3   k ← 3
4   while F^{k-1} ≠ ∅ do
5     C^k ← fsg- gen(F^{k-1})
6     for each candidate g^k ∈ C^k do
7       g^k · count ← 3
8       for each transaction t ∈ D do
9         if candidate g^k is included in transaction t then
10          g^k · count ← g^k · count + 1
11    F^k ← {g^k ∈ C^k | g^k · count ≥ σ|D|}
12    k ← k + 1
13  return F^1, F^2, … , F^{k-2}
```

其中，D表示图数据集，t表示数据集中的一个图数据，k-$(sub)graph$表示k条边的（子）图，g^k表示k阶子图。C^k表示具有k条边候选集。F^k表示k阶频繁子图。fsg-$gen(F^k)$是生成候选集C^k的算法，伪码如下：

Algorithm of fsg-$gen(F^k)$ (Candidate Generation)

```
1   C^k ← ∅
2   for each pair of g_i^k, g_j^k ∈ F^k, i ≤ j such that cl(g_i^k) ≤ cl(g_j^k) do
3     for each edge e ∈ g_i^k do {create a (k − 1)-subgraph of g_i^k by
            removing an edge e}
4       g_i^{k−1} ← g_i^k − e
5       if g_i^{k−1} is included in g_j^k then {g_i^k and g_j^k share the same core}
6         T^{k+1} ← fsg-join(g_i^k, g_j^k)
7         for each g_j^{k+1} ∈ T^{k+1} do
8           {test if the downward closure property holds for g_j^{k+1}}
9           flag ← true
10          for each edge f_l ∈ g_j^{k+1} do
11            h_l^k ← g_j^{k+1} − f_l
12            if h_l^k is connected and h_l^k ∉ F^k then
13              flag ← false
14              break
15          if flag = true then
16            C^{k+1} ← C^{k+1} ∪ {g^{k+1}}
17  return C^{k+1}
```

在判断频繁度计算时，FSG算法使用 Transaction ID lists 来对每个频繁子图维护一个事务标识符的列表。则当计算g^{k+1}的频繁度时，先计算其k阶子图的 TID 列表的交叉量，若低于阈值，则舍弃g^{k+1}，否则计算和搜索也仅限于g^{k+1}的k阶子图的 TID 列表的交叉事务集中。

FSG算法对AGM算法的优化主要体现在其采用了基于边的候选频繁子图生成方法，效率有所提升。不过仍产生了冗余的候选模式子图，所以仍需进行规范化判断。

3.5 基于模式增长的方法——$gSpan$算法[7]

AGM和FSG算法都采用了基于$Apriori$逐层推进的方法。而这种$Apriori$模式的算法会遇到两个瓶颈：从k阶频繁子图构造$k+1$阶频繁子图相当复杂且代价昂贵，同时子图同构测试是一个 NPC 问题，所以处理误报的代价也是极其昂贵。为了解决这些问题，Xifeng Yan 提出了$gSpan$（graph-based Substructure pattern mining）算法，通过对图进行深度优先搜索遍历来发现频繁子图。

$gSpan$在构建深度搜索树对图进行遍历的时候，对每个顶点按照其遍历顺序标注下标。其中v_0为根，而v_n是最右顶点。且从v_0到v_n的路径称为最右路径。

若在图G中存在两条边$e_1=(i_1,j_1)$和$e_2=(i_2,j_2)$，则该算法根据以下规则定义了两者全序关系$\prec_T$：

- 若$i_1=i_2$且$j_1<j_2$，则$e_1 \prec_T e_2$。
- 若$i_1<i_2$且$j_1=j_2$，则$e_1 \prec_T e_2$。
- 若$e_1 \prec_T e_2$且$e_2 \prec_T e_3$，则$e_1 \prec_T e_3$。

给定图G的一棵DFS树T，可以通过全序关系$\prec_T$来构造一个边序列(e_i)，其满足$\forall 0 \le i \le |E|-1, e_i \prec_T e_{i+1}$。其称之为一个$DFS$序列，同时被定义为$code(G,T)$。

对于给定图G的DFS树T的集合，其有对应的DFS序列集Z：

$$Z = \{code(G,T) | T \text{ is a DFS tree of } G\}$$

相应的可以对标签集定义全序关系$\prec_L$。并可由此定义在集合$E_T \times L \times L \times L$上的全序关系$\prec_e$。

由此对DFS序列可定义全序关系$\prec_Z$：

若Z中存在$\alpha = code(G_\alpha, T_\alpha) = (a_0, a_1, \dots, a_m)$和$\beta = code(G_\beta, T_\beta) = (b_0, b_1, \dots, b_n)$，则$\alpha \preccurlyeq \beta$当且仅当：

- $\exists t, 0 \le t \le min(m,n), a_k = b_k \text{ for } k < t, a_t \prec_e b_t$

- $a_k = b_k \; for \; 0 \le k \le m, and \; n \ge m$

由此可以对给定图G的DFS序列集$Z(G) = \{code(G,T)|T \; is \; a \; DFS \; tree \; of \; G\}$找到图$G$的最小$DFS$序列，也即$min(Z(G))$，定义为图$G$的规范化标签。由此，两个图$G$与$G'$同构当且仅当$min(G) = min(G')$。

$gSpan$算法通过这样的定义构造，将寻找频繁子图的问题转化为相应的最小DFS序列的问题，而这样的序列模式挖掘问题可以用已有的序列模式挖掘算法解决，具体实现就不再赘述了。简要来说就是，在得到k阶频繁子图的DFS编码树后，对其进行最右路径扩展。每次增加一条边，产生$k+1$阶候选频繁子图。若$k+1$阶候选频繁子图具有最小DFS序列（即规范化标签$min(Z(G_{k+1}))$），则将其保留，否则丢弃该候选子图。而在计算k阶频繁子图的$support(G_k)$的时候，记录频繁子图的所有嵌入。由此，$k+1$阶候选频繁子图的$support(G_{k+1})$即可通过k阶频繁子图的嵌入进行最右扩展而计算得到。

$gSpan$算法由于解决了$Apriori$模式的算法所遇到的两个瓶颈，大幅提高了效率并降低了空间消耗，同时避免发生产生冗余候选频繁子图的情况。

3.6 混合型方法——$FFSM$算法[8] [9]

在图数据集中进行模式挖掘的过程中，实际上会遇到诸多相当棘手的问题，诸如子图同构检测。这些问题在图数据中通常意味着远超于其他数据结构的昂贵的时间代价。由此，一种新的频繁子图挖掘算法算法快速频繁子图挖掘（Fast Frequent Subgraph Mining，$FFSM$）被提出。$FFSM$采用了垂直搜索模式，最终能减少产生的冗余候选子图。

$FFSM$算法认为，任何的子图挖掘算法将面临两个挑战：

- 子图同构问题，即判断一个给定图是否是另一个图的子图。
- 有效找出所有频繁子图的策略。

$FFSM$算法的关键特征是：

- 一种图的规范化表示和两个有效的候选子图生成方法$FFSM-Join$和$FFSM-Extension$。
- 一种代数图框架来保证能枚举出所有的频繁子图。
- 通过维护一个嵌入集，完全避免了子图同构测试问题。

$FFSM$算法仍采用标签化的邻接矩阵表示图数据，其对角线元素表示顶点的标签，非对角线元素表示边的对应标签，0 表示没有边。同AGM算法相似，$FFSM$算法也定义了$code(M)$，不过不同在于扫描的是下三角。同时也定义了$code(M_1)$与$code(M_2)$之间的全序关系。而给定图G的规范化形式邻接矩阵也被定义为具有最大$code$值的邻接矩阵，记为$G'scanonical \; adjacency \; matrix(CAM)$。与$AGM$算法不同的是，这里采用的是最大$code$值，原因在于，这样的处理能带来以下的好处：

对给定连通图G和其一个子图H，若G的CAM为A，而H的CAM为B，则显然可得到$code(A) \le code(B)$。

由于推论可得，某图G的CAM的子矩阵是图G所对应的子图的CAM。并且由此可以构造图G的连通子图树：

- 根结点是一个空矩阵。
- 树上的每个结点都是图G的连通子图，以其CAM表示。
- 每一个以CAM矩阵M表示的非根结点子图，其父结点表示M的子矩阵。

对于给定邻接矩阵M和N，二元操作$join$和一元操作$extension$如下：

- 二元操作$join$将M和N叠加以产生一个矩阵集。该步骤能通过将其规范化形式进行排序来有效的消除冗余的生成
- 一元操作$extension$将M扩展一个顶点v以及M中最后一个顶点与v相连的一条边。

对于给定图$G = (V, E, \Sigma, l)$，其中Σ表示标签集合，$l: V \cup E \to \Sigma$。

结点集$L = u_1, u_2, \dots, u_n \subset V$与$n$阶邻接矩阵$M$相兼容，当且仅当其满足：

- $\forall i, \left(m_{i,i} = l(u_i)\right)$

- $\forall i,j(i\neq j),\left(m_{i,i}\neq 0\Rightarrow\left(m_{i,j}=l(u_i,u_j)\right)\right),0<i,j<n$

对频繁子图的嵌入，$FFSM$算法是这样定义的：

对一个n阶非CAM邻接矩阵M，给定图G以及G的一个与M相兼容的结点集L，M的一个嵌入$o_M=(g_i,L)$，其中g_i为图G的事务id。对一个非CAM邻接矩阵M的所有可能嵌入被定义为它的嵌入集。

由此，可以对$FFSM$算法进行伪码描述：

Algorithm of $FFSM$ (Fast Frequent Subgraph Mining)

```
1   S ← {the CAMs of the frequent nodes}
2   P ← {the CAMs of the frequent edges}
3   FFSM-Explore(P,S)
```

Algorithm of $FFSM\text{-}Explore(P,S)$

```
1   for X ∈ P do
2     if (X.isCAM) then
3       S ← S ∪ {X}, C ← Φ
4       for Y ∈ P do
5         C ← C ∪ FFSM-Join(X,Y)
6       end for
7       C ← C ∪ FFSM-Extension(X)
8       remove CAM(s) from C that is either infrequent
        or not suboptimial
9       FFSM-Explore(C,S)
10    end if
11  end for
```

$FFSM$算法通过解决潜在的子图同构问题并减少冗余候选子图的生成，大大提升了效率。

3.7 总结和比较

频繁子图挖掘技术在发展过程中，由基于贪心策略的$SUBDUE$算法到基于ILP的方法，再到 A. Inokuchi 等人提出了基于$Apriori$思想的AGM算法，频繁子图的效率逐步提升。FSG算法对AGM算法作出了图的表示形式以及生成候选频繁子图过程的优化，提升了效率并降低了开销，但是其效果并不十分明显。直到基于模式增长的方法——$gSpan$算法被提出，开创性的结合了DFS搜索，并定义了DFS序列来减少冗余频繁子图的生成，大幅提高了效率并降低了空间消耗。而$FFSM$算法为了解决基于$Apriori$思想的算法会面临的两个挑战做出了相应的对策，通过解决潜在的子图同构问题并减少冗余候选子图的生成，也大大提升了效率。

4 结束语

随着化学信息学，生物信息学等领域的发展逐步深入，其对本身数据处理和信息发掘的需求也进一步增加，尤其是大量的图数据以及适合处理为图结构的信息。这也促使了图数据挖掘，尤其是频繁子图挖掘技术的飞速发展。本文概括性的介绍了图数据挖掘，尤其是频繁子图挖掘领域的发展历程。目前的频繁子图挖掘技术与十多年前相比，无论在适用范围还是效率及空间开销上，都已经有了飞跃性的提升。也已经初步解决了子图同构的高代价问题。由于目前的发展都是针对确定性图，将来的发展方向也许会是在非确定性图中达到类似的效能。

致谢 在此，我向对本文的工作给予支持和建议的同学和老师，以及相关的学术会议出版的论文作者表示感谢。

References:

[1] Rehman S U, Khan A U, Fong S. Graph mining: a survey of graph mining techniques[C]//Digital Information Management (ICDIM), 2012 Seventh International Conference on. IEEE, 2012: 88-92.

[2] Holder L B, Cook D J, Djoko S. Substucture Discovery in the SUBDUE System[C]//KDD workshop. 1994: 169-180.

[3] King R D, Muggleton S H, Srinivasan A, et al. Structure-activity relationships derived by machine learning: The use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming[J]. Proceedings of the National Academy of Sciences, 1996, 93(1): 438-442.

[4] Dehaspe L, Toivonen H, King R D. Finding Frequent Substructures in Chemical Compounds[C]//KDD. 1998, 98: 1998.

[5] Inokuchi A, Washio T, Motoda H. An apriori-based algorithm for mining frequent substructures from graph data[M]//Principles of Data Mining and Knowledge Discovery. Springer Berlin Heidelberg, 2000: 13-23.

[6] Kuramochi M, Karypis G. Frequent subgraph discovery[C]//Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on. IEEE, 2001: 313-320.

[7] Yan X, Han J. gspan: Graph-based substructure pattern mining[C]//Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on. IEEE, 2002: 721-724.

[8] Huan J, Wang W, Prins J. Efficient mining of frequent subgraphs in the presence of isomorphism[C]//Data Mining, 2003. ICDM 2003. Third IEEE International Conference on. IEEE, 2003: 549-552.

[9] Nijssen S, Kok J N. A quickstart in frequent structure mining can make a difference[C]//Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004: 647-652.

附中文参考文献:

[10] 李海波. 频繁子结构挖掘算法研究与应用[D]. 华中科技大学, 2011.

[11] 高琳, 覃桂敏, 周晓峰. 图数据中频繁模式挖掘算法研究综述[J]. 电子学报, 2008, 36(8): 1603-1609.