海量图数据的管理和挖掘第一次作业

北京大学计算机科学技术研究所　张晓德　1601214529

一、 问题描述

实现 Lei Zou, Yansheng Lu, Huaming Zhang, Rong Hu: PrefixTreeESpan: A Pattern Growth Algorithm for Mining Embedded Subtrees. WISE 2006: 499-505.论文中的算法。

给定一个树数据集D = {T_i|i 是下标集合}和一个最小支持度 min_sup，以及一个模式子树t_i, 我们定义d(t_i, T) = 1, 当且仅当$t_i \in T$，否则d(t_i, T) = 0. 频繁模式挖掘问题定义为发现所有的模式t_i，使得该模式在各个树中出现的次数的和大于等于 min_sup，即$\mathrm{Sup}_D(t_i) = \sum_{T \in D} d(t_i, \mathrm{T}) \geq \min_sup$.

二、 算法和实现方式

此次作业主要实现了论文中的 PrefixTreeEspan 算法来挖掘 embedded substructure，算法如下，使用 C++语言实现。

Algorithm PrefixTreeESpan
Input:A tree database D, minimum support threshold *min_sup*
Output: All frequent subtree patterns
Methods:
1) Scan D and find all frequent label *b*.
2) **For each** frequent label *b*
3) Output pattern tree <b –1>;
4) Find all **Occurrences** of b in Database D, and construct <b –1>-projected database through collecting all corresponding *Project- Instances* in *D*;
5) **call** *Fre*(<b –1>, 1 , ProDB(D,<b -1>) , *min_sup*).

Function Fre(S, *n* , ProDB(D,S), *min_sup*)
Parameters: *S*: a subtree pattern ; *n*: the length of S; *ProDB(D,S)* : the <S>-projected database; *min_sup*: the minimum support threshold.
Methods:
1) Scan *ProDB(D,S)* once to find all frequent ***GEs*** *b*.
2) **For each** GE *b*
3) extent *S* by *b* to form a subtree pattern S', and output S'.
4) Find all **Occurrences** of *b* in ProDB(D,S), and construct $< S' >$-projected database through collecting all corresponding *Project- Instances* in *ProDB(D,S)* ;
5) **call** *Fre* (S' , *n+1* , *ProDB(D, S'),min_sup*).

C++实现时，用 list 来存储整个 database，list 的每个元素是一个 vector，vector 的每个元素是一个自己定义的结构体 TreeNode，该结构体有两个属性，树的节点标号 key 和所有的祖先节点 Ancestors。结构关系可以表示如下：

List database	0	…	…	…	…
	…	…	…	…	…
	i	TreeNode (key, Ancestors)	TreeNode (key, Ancestors)	TreeNode (key, Ancestors)	…
	…	…	…	…	…
	n	…	…	…	…

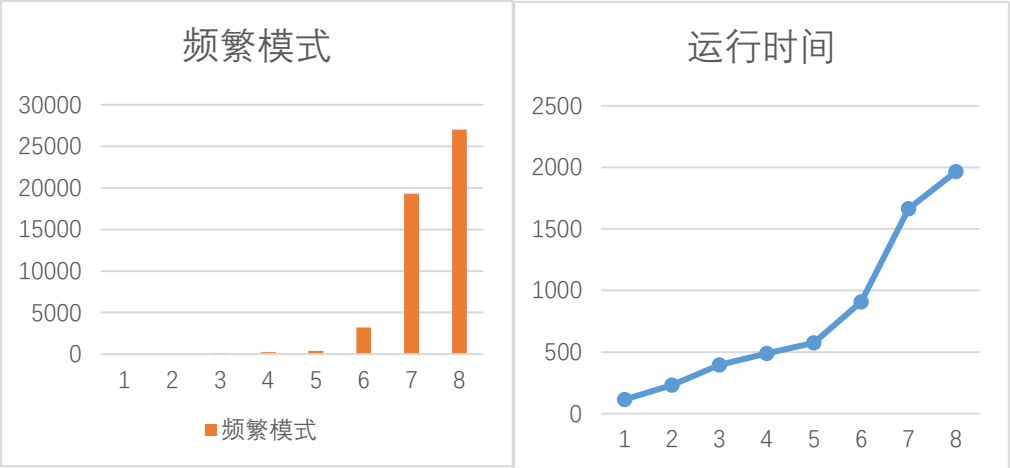
为了便于统计每个模式出现的次数，即论文中提到的每个 Growth Element 出现的次数，定义结构体 GE，属性包括节点标号 key 和祖先节点 ancestor，注意有相同的节点标号，但具有不同的祖先节点要认为是不同的 Growth Element，要分开统计。

定义好数据结构以后，即可按照论文中的算法进行编程实现，此处不再赘述。

三、 实验结果

1、数据集 T1M

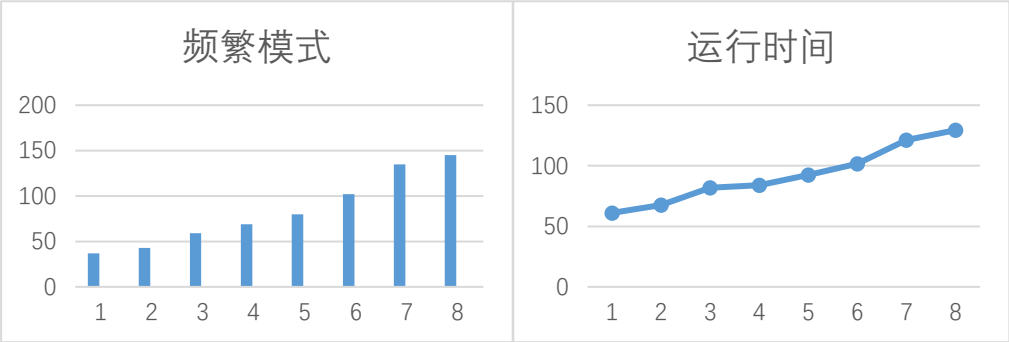
编号	1	2	3	4	5	6	7	8
min_sup	46500	20000	9000	6000	4650	3000	1500	465
频繁模式	11	38	125	260	385	3205	19312	27019
运行时间（mS）	114.9	230.9	394.3	488. 1	574. 2	907. 3	1664. 7	1964. 9



（注：该图中 1-8 对应的最小支持度如上表所示）

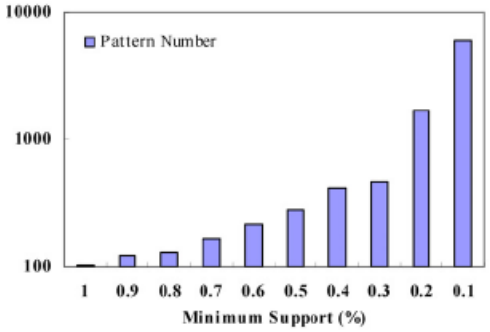
2、数据集 Cslog

编号	1	2	3	4	5	6	7	8
min_sup	2000	1800	1600	1500	1400	1200	1000	900
频繁模式	37	43	59	69	80	102	135	145
运行时间（mS）	61. 1	67. 7	81. 9	84. 1	92. 4	101. 6	121. 3	129. 4

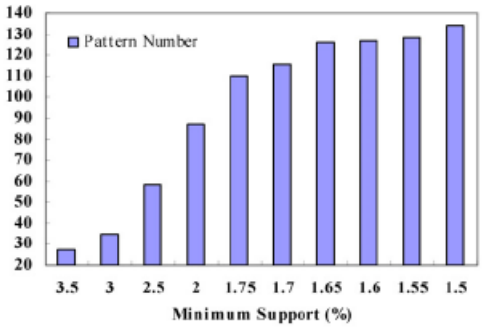


（注：该图中 1-8 对应的最小支持度如上表所示）

将上面两个数据集的运行结果和论文结果做对比，论文结果如下：



T1M



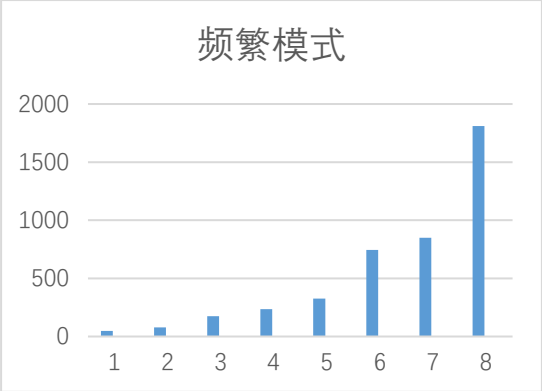
Cslog

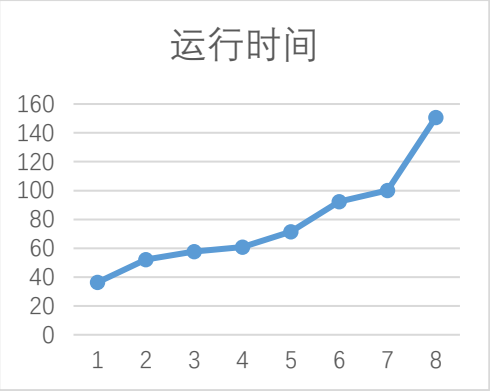
论文中是以百分比作为横轴，没有给出总数，所以此处无法用一样的数据做对比，但是从数据的增长趋势来看,T1M 上的频繁模式随着最小支持度的较小急剧增加，Cslog 上的频繁模式则缓慢增加，这一点和论文是符合的。

但是运行时间和论文相比要慢得多。

3、数据集 D10

编号	1	2	3	4	5	6	7	8
min_sup	3000	2500	2000	1800	1500	1000	800	500
频繁模式	48	77	176	236	326	745	850	1811
运行时间（mS）	36.4	52.2	57.8	60.8	71.4	92.3	100.2	150.7

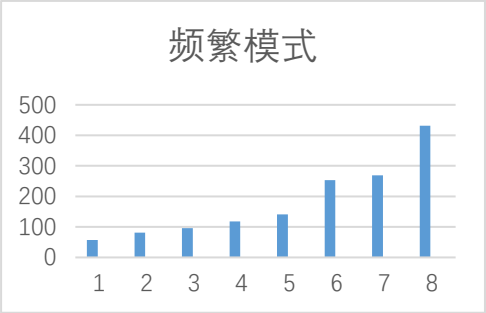


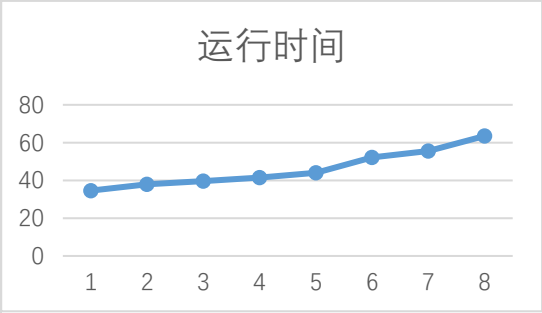


（注：该图中 1-8 对应的最小支持度如上表所示）

4、数据集 F5

编号	1	2	3	4	5	6	7	8
min_sup	3800	3000	2500	2000	1500	1000	800	500
频繁模式	57	81	96	118	141	253	269	432
运行时间（mS）	34.6	37.9	39.6	41.5	44.1	52.2	55.6	63.5





四、 运行环境和代码说明

运行代码的机器配置为 windows 7,8GB，3.6GHz

在编写代码时为了方便输入使用了输入输出重定向，运行不同的数据集需要修改主函数中数据集的名字和最小支持度 min_sup.