文章编号: 1001-0920(2009)05-0663-07

一种有效的基于图遍历的加权序列模式挖掘算法

耿汝年[1,2], 董祥军[1], 须文波[2]
(1. 山东轻工业学院 信息科学与技术学院, 济南 250353; 2. 江南大学 信息工程学院, 江苏 无锡 214122)

摘 要: 为解决加权遍历模式挖掘问题, 概括了加权有向图的种类, 提出一种边加权有向图与顶点加权有向图间的变换模型, 并基于该模型提出一种基于图遍历的加权序列模式挖掘算法 GTWSPMiner. 该算法根据遍历模式中的项的连续性特点, 采用一种加权前缀投影序列模式增长方法, 将原挖掘序列数据库的任务分解成一组挖掘局部投影数据库的小任务. 对比实验结果表明, 该算法能快速有效地挖掘加权频繁遍历模式.

关键词: 数据挖掘; 加权有向图; 遍历模式; 序列模式挖掘
中图分类号: TP311 文献标识码: A

Efficient algorithm for mining weighted sequential patterns based on graph traversals

GENG Ru-nian[1,2], DONG Xiang-jun[1], XU Wen-bo[2]
(1. School of Information Science and Technology, Shandong Institute of Light Industry, Ji. nan 250353, China; 2. School of Information Technology, Jiangnan University, Wuxi 214122, China. Correspondent: GENG Ru-nian, E-mail: gengrnn@163.com)

Abstract: To solve weighted traversal patterns mining problem, this paper generalizes the classes of weighted directed graph (WDG) and proposes a transformational model between edge-weighted directed graph (EWDG) and vertex-weighted dirlcted graph (VWDG). Based on the model, an effective algorithm called GTWSPMiner, is devised to discover weighted traversal patterns from weighted traversals database of the WDG. Based on the property that the items in a traversal pattern are consecutive, the algorithm adopts a weighted prefix-projected sequence pattern growth approach to decompose the task of mining original sequence database into a series of smaller tasks of mining locally projected database. Contrastive experimental results show that the algorithm is competent to mine weighted frequent traversal patterns efficiently.

Key words: Data mining; Weighted directed graph; Traversal pattern; Sequential pattern mining

1 引 言

近年来, 基于图遍历(访问)的数据挖掘一直是人们关注的研究热点. 图及其遍历可以模拟现实世界的多种数据访问形式. 例如, Web 结构可被模拟为一张有向图(DG), 在这张图上, 顶点代表 Web 页面, 有向边代表 Web 页面间的超链接. 用户对于 Web 页面的访问可被看作在一张图上的遍历, 每个遍历可看作一种模式. 在有向图中, 捕获用户的访问模式被称为挖掘遍历模式[1]. 一旦给出一张有向图及其上面相应的遍历, 就可以挖掘里面蕴藏的感兴趣的模式, 例如频繁模式和闭合频繁模式等. 现实中, 为了反映 Web 结构中元素的不同重要性, 可以在 DG 模型中给予每个站点或边赋予一个权值. 例如, 顶点的权值可以反映此顶点代表的页面的内容重要性, 边的权值可以代表用户在页面间的转换时间耗费或者用户停留在页面上的时间等. 这样 Web 结构便可被模拟为一张加权有向图(WDG), 对于 Web 站点的访问分析就转化为基于图遍历的加权模式挖掘问题. 然而, 传统的遍历模式挖掘算法没有考虑权值约束[2,3]. 此外, 在权值约束情况下, 支持度度量不再满足/反单调0特性, 传统的基于/向下闭合0特性的挖掘算法不再适用.

收稿日期: 2008-05-06; 修回日期: 2008-08-31.
基金项目: 山东省自然科学基金项目(Y2007G25, Y2008G26); 山东省优秀中青年科学家奖励基金项目(2006BS01017); 山东省教育厅科研发展计划项目(J06N06).
作者简介: 耿汝年(1976-), 男, 山东章丘人, 讲师, 博士生, 从事数据挖掘、人工智能、进化计算等研究; 董祥军(1968-), 男, 山东潍坊人, 教授, 博士, 从事数据挖掘与数据库技术等研究.

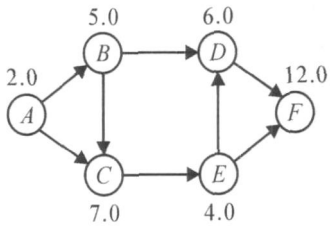
序列模式挖掘一直是数据挖掘领域中的一项重要而又活跃的研究课题. 研究人员目前已经提出多种序列挖掘算法[4-6], 但这些算法都没有考虑模式的权约束. 遍历模式中的项是连续而有序的, 因此一个遍历模式可被看作一个特殊序列模式, 进而可以用序列模式挖掘方法来挖掘遍历模式. 对于加权挖掘方法, 以前的研究工作大多与挖掘加权关联规则和加权频繁项集有关[7-11], 它们仅考虑了加权项集或关联规则挖掘, 并未考虑加权图遍历模式挖掘. 文献[12] 提出了一种加权图遍历模式挖掘算法))) GGTWFPMiner, 在挖掘过程中多次扫描遍历数据库(TDB), 当TDB很大时, 仍存在挖掘效率不高的弊端. 对于利用序列模式挖掘方法挖掘加权频繁图遍历模式的相关研究, 目前在相关电子数据库中还没有文献记载. 为更加有效地解决加权图遍历模式挖掘问题, 本文提出一种基于图遍历的加权序列模式挖掘算法(GTWSPMiner), 它利用基于前缀投影的加权模式增长方法, 有效地从加权有向图中挖掘加权频繁遍历模式.

2 问题表述

2.1 基本知识

$I = \{i_1, i_2, \ldots, i_n\}$ 是 n 个不同项的集合, 其中 $i_k(k = 1, 2, \ldots, n)$ 是不同的项. 一个项集就是 I 的一个子集. 一个序列 S 就是一个项集的有序列表, 记为 $S = \langle s_1 s_2 \ldots s_m \rangle$, 其中 s_j 是项集, $s_j \subseteq I, j \in \{1, \ldots, m\}$, s_j 也被称为 S 的一个元素, 表示为 $(i_{j1} i_{j2} \ldots i_{jl})$, i_{jk} 是 s_j 中的某个项, l 被称为 s_j 的大小. 如果 s_j 仅包含一个项, 则括号可省略. 一个项在一个序列的元素中至多可出现一次, 但可以在不同的元素中多次出现. 序列 S 的大小 $|S|$ 是序列中项集或者元素的个数. 项集 s_j 的大小 $|s_j|$ 是 s_j 中包含项的个数. 序列 S 的长度 $l(S)$ 是其包含的所有的项集大小之和, 即 $l(S) = |s_1| + |s_2| + \ldots + |s_m|$. 如果存在整数 $1 \leqslant i_1 < i_2 < \ldots < i_n \leqslant m$, 使得 $a_1 \subseteq b_{i1}, a_2 \subseteq b_{i2}, \ldots, a_n \subseteq b_{in}$, 则称序列 $A = \langle a_1 a_2 \ldots a_n \rangle$ 被序列 $B = \langle b_1 b_2 \ldots b_m \rangle$ 所包含, A是序列B的一个子序列, 序列B是序列A的一个超序列. 一个序列数据库(SDB) 就是一组二元组〈sid, S〉的集合, 其中 sid 为序列号, S 为序列.

此外, 从文献[2, 12] 可知以下基本知识: 一个加权有向图是由一组有限的顶点和边构成, 每一条边由一对有序顶点表示, 每一条边或每一个顶点都带有一定的权值, 记为 G. 存在 3 类 WDG: 顶点加权有向图(VWDG), 边加权有向图(EWDG) 以及以上两类的组合加权有向图(本质上 EWDG 与 VWDG 是等价的, 本文将在第 3 节给出 EWDG 与 VWDG 间的互换模型, 后续内容均针对 VWDG). 遍历就是在一个 DG 中, 一组沿着有向边的方向所构成的连续顶点序列. 遍历的长度就是遍历中包含顶点的个数, 如果一个遍历包含 k 个顶点, 则称其为 k-遍历或 k-模式. 在不特别指明情况下, 文中的模式均指遍历模式. 一个遍历事务数据库就是一组遍历事务的集合, 简称为遍历数据库, 记为 T. 一个遍历的子遍历就是在这个遍历中任何一个连续的子序列, 亦称为子模式. 图 1(b) 是一个从图 1(a) 所示的 VWDG 中得到的遍历数据库 T, 它包含 6 个遍历事务项. 模式 S 的支持度计数就是 T 中包含模式 S 的遍历数目, 记为 supc(S). 模式 S 的支持度 supp(S) 就是包含模式 S 的遍历数占 T 中所有遍历数的比率, 表示为 $supp(S) = supc(S) / |T|$, 其中 $|T|$ 为 T 中的总遍历个数.



(a) 一个 VWDG

序号	遍历项
1	〈A,B〉
2	〈B,C,E,F〉
3	〈A,C〉
4	〈B,C,E〉
5	〈A〉
6	〈A,C,E,D〉

(b) 遍历数据库 T

图 1 VWDG 与遍历数据库实例

2.2 相关定义及概念

从本质上看, 一个遍历就是图上的一条路径. 为研究简便, 本文假定在此路径上没有重复的顶点. 显然, 由遍历的定义易知, 遍历中的点是连续、有序的, 因此本文所讨论的遍历可以被看作一个元素大小都为 1 的序列, 相应地, 遍历数据库 TDB 可看作一个序列数据库 SDB. 此时挖掘加权有向图上的遍历模式问题就转化为挖掘加权序列模式问题.

定义 1(频繁序列)[4] 给定一个最小支持度阈值 min_sup, 如果序列 S 的支持度满足 $supp(S) \geqslant min\_sup$, 则称序列 S 是频繁遍历序列模式.

受文献[6, 8] 的启发, 给出以下定义:

定义 2(加权序列、序列的权) 一个加权序列就是其包含的顶点项带有权值的序列, 序列的权等于其所包含的所有顶点项的权值平均值[8].

给定一个加权序列 $S = \langle s_1 s_2 \ldots s_m \rangle$, 其中 $s_j = (i_{j1} i_{j2} \ldots i_{jl}), j \in \{1, \ldots, m\}, l = |s_j|$, 项 i_{jk} 的权值表示为 $w(i_{jk}), k \in \{1, \ldots, |s_j|\}$, 则 S 的权值表示为

$$weight(S) = \frac{\sum_{j=1}^{m} \sum_{l=1}^{|s_j|} w(i_{jl})}{l(S)} = \frac{\sum_{j=1}^{m} \sum_{l=1}^{|s_j|} w(i_{jl})}{\sum_{j=1}^{m} |s_j|}. \quad (1)$$

定义 3(加权支持度)[8]　序列 S 的加权支持度 wsupp(S) 定义为

$$wsupp(S) = weight(S) * supp(S). \quad (2)$$

定义 4(加权频繁序列)　给定一个最小加权支持度阈值 minwsup, 如果序列 S 的加权支持度满足

$$wsupp(S) \geq minwsup, \quad (3)$$

则称 S 为一个加权频繁序列模式.

定义 5(加权前缀)　给定一个加权序列 A= 3e1e2, , en4, 其中 ei 为加权序列数据库 SDB 中的加权频繁项集, 则序列 B= 3ec1ec2, , ecm4(m [n) 称为序列 A的一个加权前缀, 如果满足: 1) eci = ei, i [m-1; 2) ecm A em; 3) 项集(em - ecm) 中的所有项都是按字母顺序排列在 ecm 中的项后[6].

定义 6(加权后缀)　给定一个加权序列 A= 3e1e2, , en4, 其中 ei 为 SDB 中的加权频繁项集, 则序列 B= 3ec1ec2, , ecm4(m [n) 是序列 A的一个前缀, 序列 C= 3edm em+1, , en4 被称为序列 A关于加权前缀 B 的加权后缀, 其中 edm = (em - ecm). 如果 edm 非空, 则后缀 C也可表示为3(_ edm 中的项) em+1, , en4; 如果序列 B不是序列 A的一个子序列, 则序列 A关于 B的后缀为空[6].

显然, 因为本文所讨论的遍历序列的元素长度都为 1, 即 ecm = em, 也就是说定义 6 中的 ecm 为空, 因此不需要考虑定义 5 中的条件 3).

定义 7(加权投影数据库)[6]　在一个加权序列数据库 D 中, 给定一个加权序列模式 A, 则 A2 投影数据库就是 D 中所有序列关于前缀 A 的加权后缀集合, 记为 D | A.

本文所研究的问题可表述如下: 给定一个加权有向图 G, 一个最小加权支持度阈值 minwsup 和一组在 G 上的路径遍历事务项))) 遍历数据库 T, 采用序列模式挖掘方法挖掘所有带有权值约束的频繁遍历序列模式. 然而, 由式(1) ~ (3), 若一个加权序列是非加权频繁的, 那么它的加权超序列就有可能是加权频繁的, 即加权支持度度量既不满足/ 单调性0, 也不满足/ 反单调性0, 因此不能直接利用传统加权支持度的- 反单调性. 去剪切候选序列模式集.

2.3　修正序列加权支持度

为能利用加权支持度的/ 反单调性0 有效地剪切加权非频繁序列模式, 作者对序列模式的加权支持度做如下修正: 假设 G 上共有 n 个顶点, 记为{i1, i2, , , in}, 每个顶点的权值为 w(ij)(j = 1, 2, , , n), 则必然有 min(W) [w(ij) [max(W). 其中: W = {w(i1), w(i2), , , w(in)}, min(W) 和 max(W) 分别为 W 中的最小、最大权值. 为了让加权序列 S = 3s1s2, sm4(其中 sj(j I {1, , , m}) 是 G上的某个顶点) 的支持度满足/ 反单调性0, 本文修正 w(ij) 的值为 w(ij) = min(W) 或 max(W). 这样, 序列模式 S 的权值修正如下:

$$weight(S) = \frac{\sum_{j=1}^{m}\sum_{l=1}^{|s_j|} w(i_{j_l})}{l(S)} = \frac{\sum_{j=1}^{m}\sum_{l=1}^{|s_j|} \min(W)}{\sum_{j=1}^{m} |s_j|}, \quad (4)$$

$$weight(S) = \frac{\sum_{j=1}^{m}\sum_{l=1}^{|s_j|} w(i_{j_l})}{l(S)} = \frac{\sum_{j=1}^{m}\sum_{l=1}^{|s_j|} \max(W)}{\sum_{j=1}^{m} |s_j|}. \quad (5)$$

此时, 对于序列模式 Sc = S, 因为 supp(Sc) [supp(S), 如若采用修正后的序列模式权值定义(4) 或(5), 则得出 wsupp(Sc) [wsupp(S). 也就是说, 如果 wsupp(S) [minwsup, 则 wsupp(Sc) [minwsup, 即此时序列模式的加权支持度满足/ 反单调性0. 然而, 若采用式(4) 进行剪枝操作, 则显然有可能因为对加权支持度估计过低, 而剪枝掉一些本不该剪枝掉的候选序列模式, 从而造成信息失真. 采用式(5) 能避免这个问题, 但这样得出的加权支持度值只是一个近似值. 因此, 在最后阶段还要利用每一个挖掘出的序列模式 S 的真实权值去判断其是否是真正的加权频繁序列模式, 即检查每一个挖掘出的序列模式是否满足如下不等式:

$$\sum_{j=1}^{m}\sum_{l=1}^{|s_j|} w(i_{j_l}) / l(S) * supp(S) \geq minwsup. \quad (6)$$

3　EWDG 与 VWDG 间的变换模型

在图理论中[13], 一个网络可看作是一个带有加权边的有向图. 一个网络满足: 1) 至少含有一个入度(indegree) 为 0 的被称作源的顶点; 2) 至少含有一个出度(outdegree) 为 0 的被称作汇的顶点. 在这里, 一个顶点 v 的出度表示为 $deg^-(v)$, 入度表示为 $deg^+(v)$. 由前述内容知, Web 结构可抽象为一个 WDG, 即 Web 结构可看作是一个网络. 然而, 实际的 Web 是个动态、易变的网络. 出于简单考虑, 根据实际, 本文对代表 Web 结构的 WDG 作如下假设: 1) 有向边相链接的任意顶点间无访问延迟. 2) WDG 中所有边的带宽均完全一致. 3) WDG 中的大多数顶点间是双向链接的. 若 WDG 中的 A 和 B 两个顶点是可双向链接的, 则记为3A \ B4; 若是单向链接的, 则记为3A y B4 或3B y A4. 4) 权值仅是 Web 结构中顶点的重要性的一种度量, 边的权值是边的起始点所代表的 Web 元素的重要性的度量, 即有向边

$3v_i \ y \ v_j4$ 的权值(记为 weight($3v_i \ y \ v_j4$)) 表示顶点 v_i 的重要性.

3.1 从 EWDG 到 VWDG 的转换

关于从 EWDG 到 VWDG 的转换方法, 可参见文献[12], 本文不再赘述.

3.2 从 VWDG 到 EWDG 的转换

与从 EWDG 到 VWDG 的变换方法类似, VWDG 也可按照一定步骤转换为 EWDG. 本文约定 VWDG 中的顶点称为母顶点, 转换后的 EWDG 中的顶点称为子顶点. 图 2 具体描述了从 VWDG 到 EWDG 的变换方法: 图 2(a), 图 2(c) 和图 2(e) 是 VWDG; 图 2(b), 图 2(d), 图 2(f), 图 2(g) 和图 2(h) 是 EWDG. 其中图 2(f) 和图 2(g) 是中间变换过程产生的非最精简 EWDG. 图 2(b) 或图 2(d) 中的有向边 $3d_1 \ c_1b_24$, $3a_1 \ a_2b_14$, $3d_1 \ a_14$ 和 $3c_1b_2 \ a_2b_14$ 及相关带权有向边分别代表图 2(a) 或图 2(c) 中对应的带权顶点 c, a, d 和 b, 图 2(e) 到图 2(h) 描述了详细的变换过程.

变换过程经历了 3 个阶段:

阶段 1 分解 VWDG 中各个顶点阶段. 在此阶段中, 将 VWDG 中的每一顶点(即母顶点) 分解成 2 个不带权值的子顶点. 这些子顶点通过 1 条双向边链接在一起, 链接它们的边的权值等于它们的母顶点的权值大小. 如图 2(e) 中的顶点 a: 3.3, b: 2.1, c: 3.8 和 d: 4.4 分别分解为图 2(f) 中的子顶点及其带权双向边 $3a_1 \ a_24$: 3.3, $3b_1 \ b_24$: 2.1, $3c_1 \ c_24$: 3.8 和 $3d_1 \ d_24$: 4.4. 原 VWDG 中的母顶点间的链接关系由其相应子顶点继承, 如图 2(e) 中母顶点 a 的链接关系为 weight(a \ b) = 0 与 weight(d y a) = 0. 各个顶点分解为子顶点后, 顶点 a 与顶点 b 和 d 的链接关系被其子顶点继承, 即 weight($d_1 \ d_2$) = 4.4, weight(d_2 y a_1) = 0, weight($a_1 \ a_2$) = 3.3, weight($a_2 \ b_1$) = 0 和 weight($b_1 \ b_2$) = 2.1. 显然, 以上操作得到的 EWDG 与原 VWDG 具有相同功能. 此阶段对应于图 2(e) 到图 2(f).

阶段 2 区域转换阶段. 经过第 1 阶段的转换, 得到了 1 个 EWDG, 但它不是最简洁的, 因此需要继续转换. 在新的 EWDG 中, 由于在原 VWDG 中可能存在无权双向边(即权值为 0 的双向边), 这样就可能存在 weight(x \ y) = 0(其中 x 和 y 为新的 EWDG 中的顶点), 即用户对顶点 x 和 y 没有任何兴趣度, 但不能直接删除它们. 这是因为它们还与 EWDG 中的其他顶点存在链接关系, 如果删除它们, 势必使新 EWDG 的功能不同于原 VWDG 的功能. 但可以将它们及链接它们的双向边在新的 EWDG 中用一个顶点替代它们的功能. 如图 2(f) 中的顶点 a_2 和 b_1 及链接它们的双向边 $3a_2 \ b_14$ 和顶点 b_2 和 c_1 及链接它们的双向边 $3b_2 \ c_14$ 分别被简化为顶点 a_2b_1 和顶点 b_2c_1, 并使得转换后 EWDG 中的相关区域继承原来的 VWDG 中被转换区域的链接关系, 即使得

$$\text{weight}(a_1 \ a_2b_1) = \text{weight}(a_1 \ a_2) = 3.3,$$

$$\text{weight}(c_2 \ c_1b_2) = \text{weight}(c_1 \ c_2) = 3.8,$$

$$\text{weight}(a_2b_1 \ c_1b_2) = \text{weight}(b_1 \ b_2) = 2.1.$$

此阶段对应于图 2(f) ~ 图 2(g).

阶段 3 删除多余顶点, 理顺有向边的链接方向. 经过第 2 阶段的缩减后, 在新的 EWDG 中还可能存在无权单向边(即权值为 0 的单向边), 这样就可能存在 weight(x y y) = 0 或者 weight(y y x) = 0(x 和 y 为新的 EWDG 中的顶点). 其中 weight(x y y) = 0 说明用户对顶点 x 没有任何兴趣度, 而 weight(y y x) = 0 说明用户对顶点 y 没有任何兴

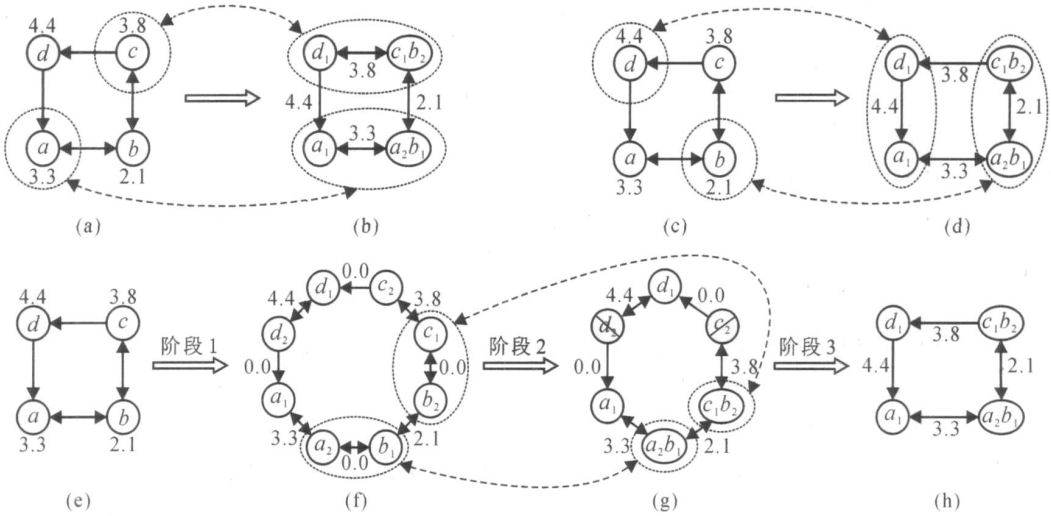


图 2 从 VWDG 到 EWDG 的变换过程

趣度. 此时可以删除这类权值为 0 的边及其起始点, 不会影响 EWDG 的结构功能. 这是因为尽管删除边及其起始点, 而边的终点还存在, 如果边的起始点与其他顶点的链接关系由边的终点继承, 则这样操作完全不会影响新的 EWDG 的功能. 如图 2(g) 中, 因为 weight(c_2 y d_1) = weight(d_2 y a_1) = 0, 删除边及其起始顶点 c_2 和 d_2, 并使得

$$weight(c_2 \backslash c_1b_2) = weight(d_1 \backslash c_1b_2),$$

$$weight(d_2 \backslash d_1) = weight(a_1 \backslash d_1),$$

最终得到最简洁的 EWDG 如图 2(h) 所示.

通过上述 2 种转换方法, 便在 EWDG 与 VWDG 之间建立了一种统一方法, 这种统一有利于简化基于图遍历的模式挖掘问题. 实践中, 用来模拟 Web 环境的WDG中的顶点的粒度大小, 可根据实际情况代表任意大小规模的子网.

GTWSP Miner(Graph Traversal-based Weighted Sequential Patterns Miner)

输入: (1) 基于加权有向图 G 的一个遍历数据库 SDB;
(2) 一个最小加权支持度阈值 minwsup;
(3) G 上各个顶点的权值

输出: G 的遍历数据库中, 所有的真正加权序列模式

方法:

1. 扫描 SDB 一次找出满足 supp(b)*Max(W)≥minwsup 全局近似加权频繁项目 b;
2. 移去所有满足条件 supp(c)*Max(W)<minwsup 的加权非频繁项 c, 对每个遍历序列元素内的剩余项(即近似候选加权频繁项), 按照字典排序;
3. 初始化 GTWSP. //GTWSP 用来储存加权序列模式;
4. Call GTWSPM(SDB,GTWSP,〈b〉,1).

Procedure GTWSPM(SDB|$_\alpha$,GTWSP,α,l)

输入: (1) SDB|α是α-投影数据库, 若$\alpha \neq \langle\rangle$, 否则, 它就是SDB;
(2) 加权序列模式集 GTSP;
(3) α是一个加权序列模式;
(4) l=|α|

输出: 真实加权序列模式集 GTWSP

方法:

1:扫描 SDB|α一次, 找出所有近似加权频繁项 β, 其满足 supp(〈β〉)*Max(W)≥minwsup, 这样
(a) β可被添加到α的最后一个元素形成一个新的加权序列模式; 或
(b) 〈β〉可被附加上α以形成一个新的加权序列模式.
2: for($\forall \beta \in$SDB|α)
3: 　添加 β 到 α 以形成一个加权序权序列α';
4: 　GTWSP=GTWSP∪〈α'〉
5: end
6: for($\forall S \in$ GTWSP)
7: 　if($\sum_{j=1,m}\sum_{i=1,|S|}w(i_{ji})/l(S)$*supp($S$)<minwsup)
8: 　　GTWSP=GTWSP-S;
9: 　else
10: 　　t=check-item-order(S);//判断 S 中的项, 与原先的相关遍历相比, 次序(包括前后邻居关系)是否正确, 正确返回 1, 否则返回 0;
11: 　　if(t==1)
12: 　　break;
13: 　　else
14: 　　GTWSP=GTWSP-S;
15: 　　end
16: 　end
17:end
18:for($\forall \alpha'$)
19: 　构建α'-投影数据库SDB|α';
20: 　call GTWSPM(SDB|α',GTWSP,α',l+1);
21:end
22:输出 GTWSP;

图 3　算法 GTWSPMiner

4　利用改进的加权投影模式增长方法从 WDG 遍历中挖掘加权频繁序列模式

由上述可知, 修订后的序列模式加权支持度具有/ 反单调性0, 即任何加权非频繁序列的超序列都是非频繁的. 根据这个性质, 基于前述两类 WDG 中的转换模型, 本文设计了一种有效、可扩展的基于图遍历的加权序列模式挖掘算法))) GTWSPMiner. 该算法运用加权投影模式增长方法, 采用/ 分而治之0 的策略, 有效地挖掘带有权值约束的加权遍历序列模式. 图 3 给出了算法 GTWSPMiner 的细节.

给定 minwsup = 1.5, 运用算法 GTWSPMiner 对图 1 中的遍历数据库 T 进行挖掘, 最终得到表 1 所示的每个前缀对应的加权频繁遍历序列模式.

表 1　投影数据库及加权频繁序列模式

前缀	前缀投影数据库	加权频繁序列模式
3A4	3B4, 3C4, 3C, E, D4	3A, C4: 2
3B4	3C, E, F4, 3C, E4	3B4: 3, 3B, C4: 2, 3B, C, E4: 2
3C4	3E, F4, 3E4, 3E, D4	3C4: 4, 3C, E4: 3
3D4	Á	Á
3E4	3F4, 3D4	3E4: 3
3F4	Á	3F4: 1

5　实验评估

由于目前没有 WDG 的真实数据集, 本实验采用合成数据集来测试算法的性能. 测试环境如下: 机器为 2.93 GHz Pentium IV PC机, 768M 内存, 操作系统为 Windows XP Professional. 算法用 C++ 语言编写, 在 VC++ 6.0 下编译运行, WDG 及其上的遍历的存储过程用 SQL Server 2000 实现, 所有实验运行时间单位为秒.

5.1　生成合成数据集

实验中, DG 根据以下主要参数实现: 顶点数目 V_n, 与每个顶点可连接的最大边数 E_{max}, 随后随机给 DG 中每个顶点赋予权值 w_i, 从而得到 VWDG. 为尽可能模拟现实情形, 权值分布采用高斯正态分布 (L= 5.0, R= 1.5), 如图 4 所示. 为便于比较算法性能, 实验生成 8 组遍历数据库, 每组都采用同样的权值集, 每组遍历中最大可遍历模式长度从 5 到 10 变化, 所有测试结果均取 8 组遍历数据测试集的平均值. 实验中具体参数设置如表 2 所示. 考虑到空间限制, 在此仅给出了较小的最小加权支持度阈值 (minwsup = 2.0) 和中等长度的最大可遍历长度 (Max_L = 7) 的实验结果. 对于其他 minwsup 和 Max_L, 其实验结果变化趋势效果与此类似.

5.2　算法有效性研究

针对minwsup, Max_L和执行时间等参数, 本

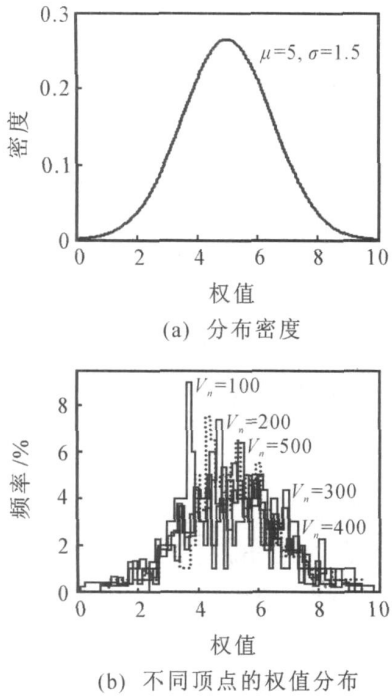


图 4 权值分布情况

表 2 实验参数

参数名称	参数含义	实验中采用值
V_n	顶点数, 即项数	100 ~ 500
E_{max}	每个顶点链接的最大边数	1 [E_{max} [4
w_i	顶点的权值范围	0 [w [10
| T |	遍历数据库大小	4000 ~ 12000
num_TSet	遍历组数	8
Max_L	最大可遍历长度	5 ~ 10

文将 GTWSPMiner 算法与 GGTWFPMiner 算法[12] 及传统的 Apriori 算法进行了性能对比实验, 其中 Apriori 算法是不带权值约束的算法.

图 5 显示了在 | T | = 10 000 时, 不同 minwsup 和 Max_L 的平均执行时间变化趋势, 其中较大 Max_L 和较小 minwsup 情况下, 算法 Apriori 的执行时间未在图中显示. 图 5(a) 表明, 对于不同的 Max_L, 算法 GTWSPMiner 比算法 GGTWFPMiner 和算法 Apriori 的执行效率高, 随着 Max_L 的增大, 算法 Apriori 的执行时间急速增加, 3 种算法的性能差别愈加明显. 图 5(b) 表明, 随着 minwsup 的降低, 3 种算法的执行时间都不断增加; 在 minwsup 的整个变化过程中, GTWSPMiner 算法执行效率优于 GGTWFPMiner 算法, 而 GGTWFPMiner 算法执行效率优于 Apriori 算法. 这是因为, 算法 GGTWFPMiner 是一种类似于基于 Apriori 算法的挖掘算法, 它采用一种/ 多次扫描 TDB y 产生候选模式 y 对候选模式进行频繁与否测试0 的挖掘机制, 对加权频繁模式进行挖掘过程中, 算法 GGTWFPMiner 要多次扫描整个 TDB, 而算法 GTWSPMiner 在挖掘过程中充分考虑了遍历事务中各顶点项的有序性, 而且它将整个挖掘任务分解成若干更小的任务, 不需要多次扫描整个 SDB; 尽管挖掘更小任务时, 需要扫描投影数据库, 但投影数据库比原始数据库小得多, 因此其执行时间耗费远低于算法 GGTWFPMiner. 此外, 算法 GTWSPMiner 和 GGTWFPMiner 是执行带有权值约束的频繁模式挖掘操作, 能有效减少候选集的搜索空间, 而 Apriori 算法仅进行不带权值约束的频繁模式挖掘, 它的搜索模式空间远大于带有权值约束的搜索空间, 因此其执行时间要远大于前两者.

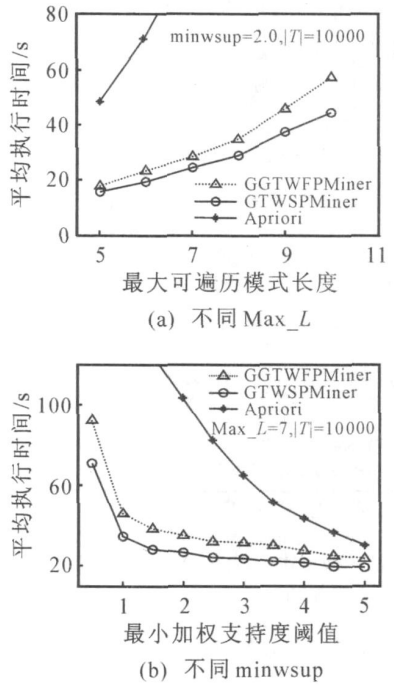
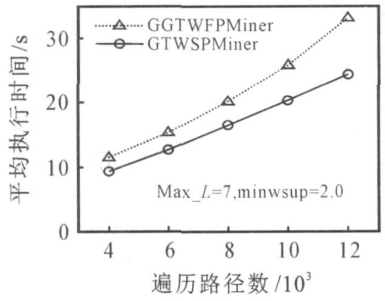


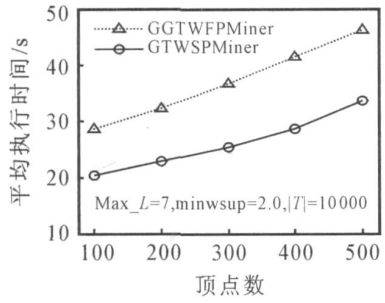
图 5 基于不同 Max_L 与 minwsup 的性能比较

5.3 算法可伸缩性(扩展性) 研究

本实验对算法 GTWSPMiner 的可扩展性进行了评估. 实验中分别对图中顶点数从 100 增加到 500 以及遍历路径(事务) 数从 4000 增加到 12000 进行测试, 实验结果如图 6 所示. 从图 6 可以看出, 算法 GTWSPMiner 无论对于顶点数还是对于遍历事务数均具有良好的可伸缩性, 其执行时间与数据集规模呈近似线性关系; 在不同顶点数和遍历事务数情况下, GTWSPMiner 算法执行效率都优于 GGTWFPMiner, 而且随着顶点数和遍历数的增加, 二者的效率差别愈加明显. 这是因为, 尽管算法 GGTWFPMiner 提供了一种挖掘加权频繁遍历模式的方法, 但由于挖掘过程中需多次扫描整个遍历数据库, 而算法 GTWSPMiner 不需要多次扫描整个 TDB. 因此, 随着顶点数和遍历数的增加, TDB 规模不断变大, 算法 GGTWFPMiner 挖掘加权频繁模式的低效率愈加明显.



(a) 遍历事务数



(b) 顶点数

图 6　算法扩展性测试

6　结　　论

本文探讨了通过加权投影模式增长法挖掘 WDG 上频繁遍历模式的问题. 总结了 WDG 的分类, 提出了不同 WDG 间的转换模型, 根据遍历模式中项的连续性特点, 将遍历模式看作序列模式, 提出了一种加权频繁遍历挖掘算法))) GTWSPMiner. 算法通过对遍历模式权值的修订, 使得模式的加权支持度具有/ 反单调性0, 从而对挖掘任务采取/ 分而治之0的策略, 将挖掘原序列数据库的总任务分解成一系列挖掘局部投影数据库的小任务. 实验结果表明, 所提出的算法能快速有效地挖掘加权频繁遍历模式, 可应用在多种可以被建模为加权有向图的环境中. 现实世界存在大量可用加权有向图模拟的实例, 如何将文中提到的模型和算法付诸实践以及是否可以进一步优化算法等尚有待进一步研究.

参考文献(References)

[1] Chen M S, Park J S, Yu P S. Efficient data mining for path traversal patterns[J]. IEEE Trans on Knowledge and Data Engineering, 1998, 10(2): 209-221.

[2] Nanopoulos A, Manolopoulos Y. Mining patterns from graph traversals[J]. Data and Knowledge Engineering, 2001, 37(3): 243-266.

[3] Nanopoulos A, Manolopoulos Y. Finding generalized path patterns for Web log data mining[C]. Proc of the 4th East-European Conf on Advances in Databases and Information Systems. Heidelberg: Springer-Verlag, 2000: 215-228.

[4] Agrawal R, Srikant R. Mining sequential patterns[C]. Proc of the 11th Int Conf Data Engineering. Los Alamitos: IEEE Computer Society, 1995: 3-14.

[5] Zaki M Spade. An efficient algorithm for mining frequent sequences[J]. Machine Learning, 2001, 42(1/2): 31-60.

[6] Pei J, Han J, Asi B M, et al. Mining sequential patterns by pattern-growth: The prefixspan approach [J]. IEEE Trans on Knowledge and Data Engineering, 2004, 16(11): 1424-1440.

[7] Wang W, Yang J, Yu P S. Efficient mining of weighted association rules (WAR)[C]. Proc of the 6th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM Press, 2000: 270-274.

[8] Cai C H, Fu A W C, Cheng C H, et al. Mining association rules with weighted items[C]. Proc of the ISDEA. 98. Washington: IEEE Computer Society, 1998: 68-77.

[9] 邹力鹃, 张其善. 基于多最小支持度的加权关联规则挖掘算法[J]. 北京航空航天大学学报, 2007, 33(5): 590-593.
(Zou L K, Zhang Q S. Algorithm of weighted association rules mining with multiple minimum supports[J]. J of Beijing University of Aeronautics and Astronautics, 2007, 33(5): 590-593.)

[10] 段军, 戴居丰. 基于多支持度的挖掘加权关联规则算法[J]. 天津大学学报, 2006, 39(1): 114-118.
(Duan J J, Dai J F. Algorithm of mining weighted association rules based on multiple supports[J]. J of Tianjin University, 2006, 39(1): 114-118.)

[11] 陆建江. 加权关联规则挖掘算法的研究[J]. 计算机研究与发展, 2002, 39(10): 1281-1286.
(Lu J. Research on algorithms of mining association rules with weighted items[J]. J of Computer Research and Development, 2002, 39(10): 1281-1286.)

[12] 耿汝年, 董祥军, 须文波. 基于全局图遍历的加权频繁模式挖掘算法[J]. 计算机集成制造系统, 2008, 14(6): 1220-1229, 1235.
(Geng R N, Dong X J, Xu W B. Weighted frequent patterns mining algorithm based on global graph traversals[J]. Computer Integrated Manufacturing Systems, 2008, 14(6): 1220-1229, 1235.)

[13] 殷剑红, 吴开亚. 图论及其算法[M]. 合肥: 中国科学技术大学出版社, 2004.
(Yin J H, Wu K Y. Graph theory and its algorithm [M]. Hefei: University of Science and Technology of China Press, 2004.)