一种基于桶树的自动推理问题求解算法

袁 譽[1] 胡春玲[1] 胡学钢[2] 姚宏亮[2]
(合肥学院网络与智能信息处理重点实验室 合肥 230602)[1]
(合肥工业大学计算机与信息学院 合肥 230009)[2]

摘 要 桶消元和连接树推理算法是处理自动推理问题的两种常用的推理算法。针对连接树推理算法中消息传播效率问题,提出了一种能有效进行消息传播的连接树推理算法JTR。针对桶消元推理算法BE处理多任务的自动推理问题效率低下的问题,采用连接树结构和连接树推理算法JTR的消息传播方式对桶消元算法BE进行改进和扩展,提出了一种桶树推理算法BJTR。通过对算法BE、BTE和BJTR的时空性能分析发现:与同类算法BTE相比,算法BJTR在空间略有下降的情况下提高了时间性能;针对多任务的自动推理问题,与桶消元推理算法BE相比,BJTR算法的空间略有下降,时间性能得到明显提高;并通过实例和实验进一步验证了算法BJTR针对多任务的自动推理任务具有良好的时间性能。

关键词 自动推理,多任务,桶消元,连接树,桶-树
中图法分类号 TP181 **文献标识码** A

Bucket-Tree Based Algorithm for Automated Reasoning

YUAN Min[1] HU Chun-ling[1] HU Xue-gang[2] YAO Hong-liang[2]
(Key Laboratory of Network and Intelligent Information Processing, Hefei University, Hefei 230602, China)[1]
(School of Computer and Information, Hefei University of Technology, Hefei 230009, China)[2]

Abstract The bucket elimination algorithm and the join-tree reasoning algorithm are popularly used for automated reasoning. To improve the efficiency of message propagation in the join-tree reasoning algorithm, a new join-tree reasoning algorithm (called JTR) was proposed. Meanwhile, to handle the inefficiency of multi-task automated reasoning of the bucket elimination algorithm BE, a bucket-tree reasoning algorithm (named as BJTR), based on the join-tree structure and message propagation mode in JTR, was further developed from BE. Our study shows that in comparison with the BTE algorithm, the proposed algorithm BJTR improves the time performance while space performance decreases a little. Furthermore, as compared with BE, BJTR effectively reduces the demand on time-overhead while maintaining a slightly low space performance in the handling of multi-task automated reasoning. Meanwhile, both examples and experiments demonstrate that BJTR algorithm has an obvious advantage in the time performance for multi-task reasoning.

Keywords Automated reasoning, Multi-task, Bucket elimination, Join-tree, Bucket-tree

1 引言

自动推理可以解决不确定性推理和确定性推理领域的诸多问题,如:不确定性推理问题中概率推理领域的信度更新、最大概率解释、最大后验估计、最大效用估计等[1];确定性推理领域的约束满足及组合优化问题等[2,3]。自动推理问题的求解算法包括精确推理算法和近似推理算法,变量消元推理算法和连接树推理算法[4,5]是两大类主要的精确推理算法。

变量消元推理算法中最具代表性的是桶消元推理算法[6],所有桶消元推理算法在推理功能和推理效率两个方面都比较接近,该类算法的时间复杂度为推理过程中所产生导出图的导出宽度的指数级。桶消元推理算法的优点是:推理过程简单、规范、易于实现,该类算法基于统一的算法框架,算法的一致性使得对一种算法的改进和扩展易于移植同类算法中的其它算法。桶消元推理算法的缺点是:一次推理过程结束后,只能完成针对一个单变量的推理任务,若需要实现对一系列变量或变量组合相对于新证据的查询,则需要有多次推理过程,因此针对多任务的自动推理问题,桶消元推理算法的推理效率较低。

连接树推理算法中的代表算法主要包括LS结构、SS结构及Hugin结构等算法[7-9],该类算法的时间复杂度为连接树宽度的指数级。宽度最小的连接树为最优连接树,已经证

到稿日期:2012-04-15 返修日期:2012-08-10 本文受国家自然科学基金(60975034),安徽省教育厅自然科学一般项目(KJ2010B177),合肥学院人才科研基金(13RC01)资助。

袁 譽(1956-),女,硕士,副教授,主要研究方向为计算计网络、人工智能;胡春玲(1970-),女,博士,讲师,主要研究方向为自动推理、贝叶斯网络,E-mail:huchunling8787@yahoo.com.cn;胡学钢(1961-),男,博士,教授,博士生导师,主要研究方向为知识工程、数据挖掘;姚宏亮(1972-),男,博士,副教授,主要研究方向为人工智能、知识工程。

明构造一棵最优连接树是 NP 问题[10]，目前，已有一些方法可以构建近似最优的连接树[11]。连接树推理算法存在连接树构造困难问题，在连接树构造过程中需要借助图论中诸如三角化、有弦图、子团等的相关操作和概念，但处于一致状态下的连接树推理算法可以完成对所有单变量和属于某一子团的变量集相对于新证据的查询任务，因此针对多任务的自动推理需求，连接树推理算法的推理效率较高。

分析发现变量消元推理算法和连接树推理算法有很强的互补性，目前，已有学者通过不同的途径尝试将这两类算法进行融合，以达到取长补短的目的，其中具有代表性的工作是 Cozman 提出的一种针对贝叶斯网络概率推理问题的桶树消元算法[12]和 Kask 等人提出的 BTE 算法[13,14]。Cozman 提出的算法在桶消元推理算法的桶与桶之间建立一种树结构，对两类推理算法的融合做了初步尝试，但所提算法仅针对贝叶斯网络的推理问题，且没有对所提算法与连接树推理算法的内在关系以及时空性能进行分析。Kask 等人基于桶树结合的思想提出了一种面向自动推理问题的算法 BTE(Bucket Tree Elimination)，该算法能通过一次执行处理自动推理问题的多任务，但与桶消元推理算法通过多次执行处理多任务相比，BTE 算法的推理效率没有得到明显提高。

针对多任务的自动推理问题中的概率推理问题、约束满足问题和组合优化问题，本文基于桶树结合的思想，提出了一种桶树推理算法 BJTR(Bucket Join-Tree Reasoning)，该算法采用连接树结构和连接树推理算法的消息传播方式对桶消元推理算法进行改进和扩展。本文的工作主要包括以下 3 个方面：①基于图模型给出自动推理问题的一种定义，并针对该问题中连接树推理算法的推理效率问题，提出了一种连接树推理算法 JTR(Join-Tree Reasoning)；②针对多任务的自动推理问题，提出了一种桶树推理算法 BJTR，该算法通过在桶消元推理算法 BE(Bucket Elimination)中桶与桶之间建立连接树结构，提供了一种直观易实现的连接树构造方法，并采用连接树推理算法 JTR 对所建立的连接树进行推理；③分析并比较了桶消元推理算法 BE、桶树推理算法 BTE 和本文所提算法 BJTR 的时空性能。

以下描述中 n 表示变量集 X 中的变量个数，n^* 表示连接树中的节点个数，w^* 表示连接树的宽度，deg 表示连接树的入度，sep 表示连接树中最大的割集长度，w 表示消元过程所产生导出图的导出宽度，r 表示函数或关系表 F 的长度。

2 自动推理

文献[14]采用四元组按定义 1 定义了图模型。针对所处理的自动推理问题，本文在对定义 1 中的组合运算进行简化的基础上给出了定义 2，该定义通过七元组来描述自动推理问题。

定义 1 定义图模型为一个四元组 $G=\langle X,D,F,\otimes\rangle$，其中①$X=\{x_1,x_2,\cdots,x_n\}$ 表示变量集；②$D=\{D_1,D_2,\cdots,D_n\}$ 表示变量集 X 中各变量的论域，其中 D_i 对应变量 x_i 的论域；③$F=\{f_1,f_2,\cdots,f_r\}$ 表示定义在变量集 X 上的函数集或关系集，即：f_i 为定义在变量集 S_i 的函数或关系，变量子集 S_i 和其论域 D_{S_i} 满足：$S_i\subseteq X$；$D_{S_i}\subseteq D$；④$\otimes f_i\in\{\Pi f_i,\Sigma f_i,\infty f_i\}$ 表示不同的组合运算，其中 ∞f_i 表示关系的连接运算。

按此定义，可采用 F 中函数或关系的组合运算 $\otimes_{i=1}^{r} f_i$ 表示图模型。

定义 2 自动推理问题可以定义成一个七元组 $P=\langle X,D,F,\otimes,\tilde{\otimes},\Downarrow,\{Z_1,Z_2,\cdots,Z_t\}\rangle$，其中符号 X,D 的定义同定义 1，其余符号的定义为：①$F=\{f_1,f_2,\cdots,f_r\}$ 表示定义在变量集 X 上的函数集；②$\otimes f_i\in\{\Pi f_i,\Sigma f_i\}$ 表示定义在函数集 F 上的组合运算；③$\tilde{\otimes} f_i\in\{\div f_i,-f_i\}$ 表示该定义中组合运算 $\otimes$ 的逆运算；④$\Downarrow_Y f\in\{\mathrm{Max}_{S-Y}f,\mathrm{Min}_{S-Y}f,\Sigma_{S-Y}f\}$ 表示边缘化运算，其中 S 为 f 的定义域且 $Y\subseteq S$；⑤自动推理问题为：针对待查询变量集 $\{Z_1,Z_2,\cdots,Z_t\}$，计算 $\Downarrow_{Z_1}\otimes_i f_i$，$\Downarrow_{Z_2}\otimes_i f_i,\cdots,\Downarrow_{Z_t}\otimes_i f_i$，其中 $Z_i\in X,i\in\{1:t\}$。

本文给出的定义 2 是面向以下 3 类具体的自动推理问题的。

①不确定性推理问题中贝叶斯网络的概率推理问题。按照定义 2，概率推理问题中 X 对应于贝叶斯网络中的节点，D 对应于贝叶斯网络中每个节点的论域，F 中所包含的函数对应于网络中每个节点的条件概率分布表，组合运算为乘运算：$\otimes f_i=\times f_i$，逆组合运算为除运算：$\tilde{\otimes} f_i=\div f_i$。概率推理任务主要包括变量集信度更新、最大概率解释和最大后验假设，其中信度更新中的边缘化运算为求和运算，即：$\Downarrow_Y=\Sigma_{X-Y}$ 为在变量子集 X-Y 上的求和运算；最大概率解释的边缘化运算为求变量全集 X 上的最大值运算：$\Downarrow_\phi=\mathrm{Max}_X$；最大后验假设的边缘化运算为求变量子集 Y 上的最大值运算：$\Downarrow_\phi=\mathrm{Max}_Y$，其中 Y 为 X 的子集，X-Y 中变量在求解最大后验假设问题中为证据变量。

②确定性推理中约束满足问题 CSP(Constraint Satisfaction Problem)。该问题中 X 对应于有限变量集，D 对应于有限变量集 X 中每个变量的论域，F 表示变量之间的基本约束集，每条约束限制了变量集赋值的组合。约束满足问题的解是为变量集中每个变量赋一个对应论域上的值，使 F 中的所有约束得到满足。求解约束满足问题的目标是找到一个解或是全部解。约束满足问题存在不相容的情况，即在求解经典约束满足问题时，常常因为约束限制得太严格了，使得整个问题无解，这时通常转而求解使 F 中不能被满足的约束最少的解，即：求解能满足 F 中最多约束的解，相应的问题就称为 Max-CSP，又称为组合优化问题。

③组合优化问题 Max-CSP。该问题中 X 对应于有限变量集，D 对应于有限变量集 X 中每个变量的论域，表 F 对应于实值表示的代价函数，表示相应约束不被满足的代价，该问题组合运算为和运算：$\otimes f_i=+f_i$；逆组合运算为差运算：$\tilde{\otimes} f_i=-f_i$；边缘化运算为求最小值运算：$\Downarrow_\phi=\mathrm{Min}_X$，组合优化问题是求解具有全局最小代价的变量集 X 的一个完全赋值。

本文对约束满足问题和组合优化问题的描述进行了统一，将约束满足问题作为组合优化问题的特例进行求解，即：约束满足问题对应组合优化问题的目标代价函数值为零的情况，当组合优化问题的解满足 F 中所有的约束时，即：代价为零时，该解为相应约束满足问题的解。因此，定义 2 中，$F=\{f_1,f_2,\cdots,f_r\}$ 表示定义在变量集 X 上的函数集，在不确定性推理中，$f_i,i\in[1:r]$ 对应于不同的条件概率分布函数；在确定性推理中，$f_i,i\in[1:r]$ 对应于实值表示的相应约束不被满足的代价函数，进一步，定义 2 中定义在函数集 F 上的组合运算集 $\otimes f_i\in\{\Pi f_i,\Sigma f_i\}$ 中也就不再包含定义 1 中关系的

连接运算∞f_i。

3 自动推理问题的桶树推理算法

3.1 连接树推理算法 JTR

首先针对定义 2 所描述的自动推理问题，给出如下的一种连接树定义。

定义 3 定义自动推理问题 $P=\langle X,D,F,\otimes,\widetilde{\otimes},\Downarrow,\{Z_1,Z_2,\cdots,Z_t\}\rangle$ 对应的一种连接树为 $\langle T,\chi,\varphi,\psi,M\rangle$，其中 $T=(V,E)$ 为树结构，V 表示 T 中的节点集，E 表示 T 中的边集，对于 V 中的任一节点 v_i，$\chi(v_i)$ 为分配到 v_i 中的一个属于 X 的属性子集，$\varphi(v_i)$ 为分配到 v_i 中的一个属于 F 的函数子集，$\psi(v_i)$ 为 $\varphi(v_i)$ 中的所有函数的组合运算结果，即：$\chi(v_i)\subseteq X$；$\varphi(v_i)\subseteq F$；$\psi(v_i)=\otimes_{f_i\in\varphi(v_i)}f_i$，$M$ 表示在 E 中不同边上传播的消息；同时 $\langle T,\chi,\varphi,\psi,M\rangle$ 满足以下条件：

①对于 F 中的任一定义在变量集 S_i 上的函数 f_i，存在 V 中的唯一节点 v_i 与其对应，即满足：$S_i\subseteq\chi(v_i)$，$f_i\in\varphi(v_i)$；

②连接树满足连接性，即：若 X 中某一变量 x_i 同属于树结构 T 中节点 v_i 和 v_j，则 (v_i,v_j) 对应于连接树 T 中的一条路径，且该路径上的任一个节点 v_k 均包含节点 x_i，即满足：$x_i\in\chi(v_k)$；

③对于 $\forall Z_i\in\{Z_1,Z_2,\cdots,Z_t\}$，$\exists v_j\in V$，满足：$Z_i\subseteq\chi(v_j)$。

连接树 T 的宽度 w^* 为连接树中不同节点 v_i 所包含的变量集长度的最大值，即：$w^*=\max_{v_i\in V}|\chi(v_i)|$。连接树中连接相邻节点 (v_i,v_j) 的边称为连接树的超级边，该边对应于连接树的割集 $C_{ij}=\chi(v_i)\cap\chi(v_j)$。对应于问题 P 的符合定义 3 的树结构均为该问题的连接树，显然，对应于问题 P 的连接树不是唯一的。

图 2 为图 1 所示的贝叶斯网络对应的一种连接树结构，该图中 v_1 为连接树的根节点。连接树推理算法的推理过程就是以根结点为中心，通过自下而上和自上而下两个阶段的消息传播使连接树进入一致状态后，完成各种推理任务。在两个阶段的消息传播过程中，连接树中任一节点 v_i 在收到除第 k 个邻居节点之外的其它邻居节点传播过来的消息后，才能按式(1)计算向其第 k 个邻居节点传播的消息 M_{ik}。

$$M_{ik}=\Downarrow_{C_{ik}}\otimes_{f_i\in\varphi(v_i)}f_j\otimes_{m\neq k}M_{mi} \tag{1}$$

针对同一棵连接树，消息传播方式的不同会导致推理效率的不同，为了提高连接树推理算法的推理效率，本文针对满足定义 3 的自动推理问题提出了一种连接树推理算法 JTR，该算法通过对连接树推理算法的消息传播方式进行改进，以达到避免冗余计算的目的。算法 JTR 首先根据分配到 $\varphi(v_i)$ 中的不同函数 f_i 按组合运算对每个节点 v_i 的 $\psi(v_i)$ 进行初始化，即：$\psi(v_i)=\otimes_{f_i\in\varphi(v_i)}f_i$，然后在自下而上的消息传播过程中根据节点 v_i 的子节点 v_k 传播上来的消息 M_{ki} 按组合运算 $\psi(v_i)=\psi(v_i)\otimes M_{ki}$ 及时更新 $\psi(v_i)$，进而能够实现在自上而下的消息传播过程中，按式(2)仅通过一次逆组合运算即可完成父节点 v_i 对子节点 v_k 所传播的消息 M_{ik} 的计算。因为式(2)满足：

$$\begin{aligned}M_{ik}&=\Downarrow_{C_{ik}}\psi(v_i)\widetilde{\otimes}M_{ki}=(\Downarrow_{C_{ik}}\otimes_{f_i\in\varphi(v_i)}f_i\otimes_m M_{mi})\widetilde{\otimes}M_{ki}\\&=\Downarrow_{C_{ik}}\otimes_{f_i\in\varphi(v_i)}f_i\otimes_{m\neq k}M_{mi}\end{aligned}$$

故与式(1)在本质上是一致的。

$$M_{ik}=\Downarrow_{C_{ik}}\psi(v_i)\widetilde{\otimes}M_{ki} \tag{2}$$

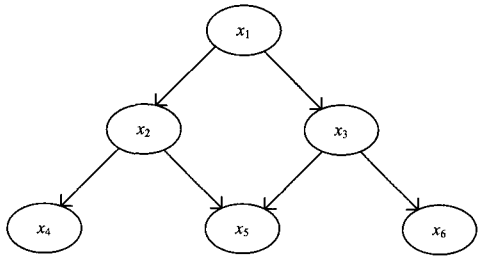

图 1 一个贝叶斯网络

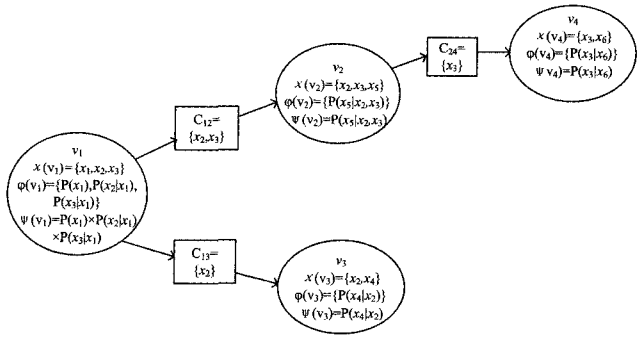

图 2 图 1 中贝叶斯网络对应的连接树

算法 JTR 的描述如下：

输入：自动推理问题 $P=\langle X,D,F,\otimes,\widetilde{\otimes},\Downarrow,\{Z_1,Z_2,\cdots,Z_t\}\rangle$ 及其对应的连接树 $\langle T,\chi,\varphi,\psi,M\rangle$。

输出：一致状态下的连接树 $\langle T,\chi,\varphi,\psi,M\rangle$ 及自动推理问题的求解。

①初始化：T 中每个节点 v_i 对应的 $\psi(v_i)$ 函数或关系初始化为：$\psi(v_i)=\otimes_{f_i\in\varphi(v_i)}f_i$；

②自下而上的消息传播阶段：从连接树的叶节点开始，每个节点 v_i 在收到其所有子节点传递过来的消息后按运算：$M_{ij}=\Downarrow_{C_{ij}}\psi(v_i)$；$\psi(v_j)=\psi(v_j)\otimes M_{ij}$ 向其父节点 v_j 传播消息，并根据传播过来的消息更新父节点 v_j 对应的 $\psi(v_j)$；叶节点直接向其父节点传播消息；

③自上而下的消息传播阶段：从连接树的根节点开始，每个节点 v_i 在收到其父节点 v_j 传播过来的消息后按运算：$M_{ji}=\Downarrow_{C_{ij}}\psi(v_j)$；$M_{ji}=M_{ji}\widetilde{\otimes}M_{ij}$；$\psi(v_i)=\psi(v_i)\otimes M_{ji}$ 向其子节点 v_i 传播消息，并根据传播过来的消息及时更新子节点 v_i 对应的 $\psi(v_i)$；根节点直接向其子节点传播消息；

④两次消息传播结束后，连接树进入一致状态，根据一致状态下的连接树进行问题求解：对于任一 Z_k，$k\in[1:t]$，I. 若 Z_k 属于 C_{ij}，则 $\Downarrow_{Z_k}M_{ji}\otimes M_{ij}$ 为问题的解；II. 若 Z_k 不属于任一 C_{ij}，则 Z_k 一定属于连接树中的某一节点 v_i，则 $\Downarrow_{Z_k}\psi(v_i)$ 即为问题的解。

图 3 为在图 2 所示连接树上采用 JTR 算法的消息传播方式进行两次消息传播后对应的一致状态下的连接树，然后按照算法 JTR 的运算步骤④可以完成不同的推理任务。JTR 算法在其消息传播过程中根据所传播的消息及时更新相应节点 v_i 的 $\psi(v_i)$ 函数，因此图 3 所示连接树节点 v_i 中 $\psi(v_i)$ 对应的是该函数在连接树处于一致状态下的取值，割集 C_{ij} 中 $\psi(v_i)$ 对应的是该函数在消息传播不同阶段的取值，割集 C_{ij} 中 $M_{ij}=\sum_{C_{ij}}\psi(v_i)=\sum_{C_{ij}}\prod_{f_i\in\varphi(v_i)}f_i\prod_{k\neq j}M_{ki}\ (i>j)$，其中 $\psi(v_i)$ 对应于自下而上的消息传播过程中节点 v_i 的 ψ 函数，割集 C_{ij} 中 $M_{ji}=\sum_{C_{ij}}\psi(v_j)\div M_{ij}=\sum_{C_{ij}}\prod_{f_j\in\varphi(v_j)}f_j\prod_{k\neq i}M_{kj}\ (i<j)$，其中 $\psi(v_j)$ 对应于自上而下的消息传播过程中节点 v_j 中的 ψ 函数，与节点

v_j 在一致状态下 ψ 函数是相同的。如：割集 C_{21} 中 $M_{21}=\sum_{x_5}\psi(v_2)=\sum_{x_5}P(x_5|x_2,x_3)\times M_{42}$，其中 $\psi(v_2)$ 对应自下而上的消息传播过程中节点 v_2 的 ψ 函数，与一致状态下连接树中节点 v_2 中的 $\psi(v_2)$ 函数是不同的；割集 C_{21} 中 $M_{12}=(\sum_{x_1}\psi(v_1))\div M_{21}=(\sum_{x_1}P(x_1)\times P(x_2|x_1)\times P(x_3|x_1)\times M_{21}\times M_{31})\div M_{21}=\sum_{x_1}P(x_1)\times P(x_2|x_1)\times P(x_3|x_1)\times M_{31}$，其中 $\psi(v_1)$ 对应自上而下消息传播过程中节点 v_1 的 ψ 函数，与一致状态下连接树中节点 v_1 中的 $\psi(v_1)$ 函数是相同的。

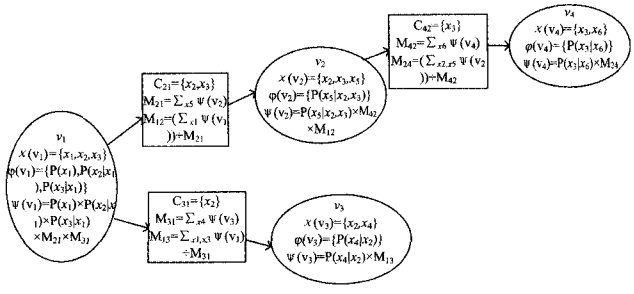

图 3　图 2 所示连接树的两次消息传播

定理 1　假设自动推理问题 P 中函数表 F 的长度为 r，该问题对应连接树 T 中的节点个数为 n^*，连接树的宽度为 w^*，则 JTR 算法的时间复杂度为 $O((n^*+r)\cdot\exp(w^*))$，空间复杂度为 $O(n^*\cdot\exp(w^*))$。

证明：①连接树 T 中所有节点的 ψ 函数初始化的时间复杂度：假设分配到节点 v_i 中的函数 f_i 的个数为 r_i，则节点 v_i 的 ψ 函数初始化的时间为 $r_i\cdot\exp(|\chi(v_i)|)$，所有节点的 ψ 函数初始化的时间为 $\sum_i r_i\cdot\exp(|\chi(v_i)|)$，又 $|\chi(v_i)|\leqslant w^*$，则初始化过程的时间复杂度为 $O(r\cdot\exp(w^*))$。

②连接树 T 中两次消息传播的时间复杂度：I. 计算存储连接树中每个割集 C_{ij} 中的消息所需要的时间为 $O(\exp(|\chi(v_i)\cap\chi(v_j)|))$；II. 计算连接树中每个节点 v_i 向其邻居节点 v_j 传播消息所需要的时间为 $O(\exp(|\chi(v_i)|)+\exp(|\chi(v_i)\cap\chi(v_j)|))$；III. 更新节点 v_j 中 ψ 函数所需要的时间为 $O(\exp(|\chi(v_j)|))$；假设连接树 T 中节点 v_i 的邻居节点的个数为 n_i，则两次消息传播过程所需要的时间为 $\sum_i n_i\cdot O(\exp(|\chi(v_i)|)+2\exp(|\chi(v_i)\cap\chi(v_j)|)+\exp(|\chi(v_j)|)$，又 $|\chi(v_i)|\leqslant w^*$，$i\in[1:n]$，两次消息传播过程的时间复杂度为 $O(n^*\cdot\exp(w^*))$。综合①和②，则 JTR 算法的时间复杂度为 $O((n^*+r)\cdot\exp(w^*))$。

③JTR 算法存储割集 C_{ij} 传播的消息 M_{ij} 和 M_{ji} 所需要的空间为 $2\exp(|C_{ij}|)$，JTR 算法存储节点 v_i 中 $\psi(v_i)$ 函数所需要的空间为 $\exp(|\chi(v_i)|)$，又连接树 T 中共有 n^* 个节点，则 JTR 算法所需空间为 $n^*\cdot\exp(|\chi(v_i)|)+2(n^*-1)\cdot\exp(|C_{ij}|)$，又 $|C_{ij}|\leqslant|\chi(v_i)|\leqslant w^*$，则 JTR 算法的空间复杂度为 $O(n^*\cdot\exp(w^*))$。故得证。

文献[14]提出了一种处理自动推理问题的连接树推理算法 CTE，分析发现该算法在消息传播的过程中和消息传播结束后推理计算中均存在冗余计算。假设连接树的入度为 deg，连接树中割集的最大长度为 sep，CTE 算法的时间复杂度和空间复杂度分别为 $O((n^*+r)\cdot deg\cdot\exp(w^*))$ 和 $O(n^*\cdot\exp(sep))$。与 CTE 算法相比，JTR 算法的时间复杂度与连接树的稠密程度无关，时间性能有明显改善，原因在于 JTR 算法在自上而下的消息传播中仅通过一次逆组合运算即可计算向子节点所传播的消息，并在消息传播结束后的推理任务的完成中也能有效地避免冗余计算。因 $sep<w^*$，相比于 CTE 算法，JTR 算法在改善时间性能的同时可增加空间开销，但这一问题在下文的桶树结构的连接树推理算法 BJTR 中并不突出，因为在 BJTR 算法中 sep 和 w^* 之间满足关系：$sep=w^*-1$。

3.2　桶消元推理算法 BE

针对定义 2 所描述的自动推理问题 $P=\langle X,D,F,\otimes,\widetilde{\otimes},\Downarrow,\{Z_1,Z_2,\cdots,Z_t\}\rangle$，桶消元推理算法 BE 根据给定的消元顺序 $d=\langle x_1',x_2',\cdots,x_n'\rangle$ 建立 n 个桶 $B_{x_i'}(i=1,2,\cdots,n)$，并在完成一次自下而上的消息传播后，在桶 $B_{x_1'}$ 进行关于 x_1' 的自动推理问题的求解。

算法 BE 的描述如下：

输入：自动推理问题 $P=\langle X,D,F,\otimes,\widetilde{\otimes},\Downarrow,\{x_1'\}\rangle$，变量消元顺序 $d=\langle x_1',x_2',\cdots,x_n'\rangle$。

输出：关于 d 中首变量 x_1' 的自动推理问题求解。

①初始化：将 F 中各个 f_i 分配到定义域 S_i 在 d 中序号最大的变量 x_i' 所对应的桶 $B_{x_i'}$ 中；

②自下而上的消息传播阶段：按 d 的逆序 $x_n'\rightarrow x_1'$ 依次进行消元，假设待消元变量 x_i' 所对应的桶 $B_{x_i'}$ 中的函数集为 $\{\lambda_1,\lambda_2,\cdots,\lambda_j\}$，相应的定义域为 $\{S_1,S_2,\cdots,S_j\}$，若记 $U_i=\bigcup_{l=1}^{j}S_l-\{x_i\}$，则有：$\lambda_i=\Downarrow_{U_i}\bigotimes_{l=1}^{j}\lambda_l$，并将 λ_i 传播到其定义域 U_i 在 d 中序号最大的变量 x_k' 所对应的桶 $B_{x_k'}$ 中；

③消息传播结束后，在桶 $B_{x_1'}$ 中进行关于 x_1' 的自动推理问题的求解，即：$\Downarrow_{x_1'}\otimes_{\lambda\in B_{x_1'}}\lambda$ 为所求自动推理问题的解。

图 4 为桶消元推理算法 BE 在图 1 所示的贝叶斯网络上按 $d=\{x_1,x_2,\cdots,x_6\}$ 的逆序进行消元所对应的一次消息传播过程，消息传播结束之后，可在桶 B_{x_1} 中完成变量 x_1 的信度更新、最大后验假设等概率推理任务。

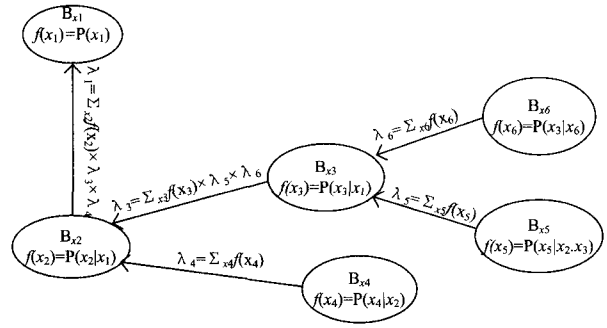

图 4　BE 算法的一次消元求解过程

图模型 $G=\langle X,D,F,\otimes\rangle$ 的初始图（Primal Graph）G' 定义为 G 对应的无向图，满足：对 F 中任一 f_i 的定义域 S_i 中任意的节点对 $\forall(x_j,x_k)\subseteq S_i$，$i=1,2,\cdots,r$，在 G' 中节点 x_j 和 x_k 之间均存在一条对应的无向边[14]。

BE 算法按给定消元次序 d 求解自动推理问题 P 的过程对应于导出图 G_d' 的产生，导出图 G_d' 的初始值为问题 P 对应的初始图 G'，在按 d 的逆序对 G' 的各节点 x_i 进行桶消元产生导出图 G_d' 的过程中，在 G_d' 中 x_i 的所有邻居节点之间添

加无向边。BE算法的时间复杂度取决于导出图 G_d' 的导出宽度[14]。假设自动推理问题 P 中变量集 X 包含的变量个数为 n，表 F 的长度为 r，桶消元推理算法 BE 求解问题 P 过程中产生的导出图的导出宽度为 w，则桶消元推理算法 BE 的时间复杂度为 $O((n+r)\cdot\exp(w+1))$，空间复杂度为 $O(n\cdot\exp(w))$。

3.3 桶树推理算法 BJTR

桶消元推理算法在一次消息传播过程结束之后只能完成针对单个变量的自动推理任务，因而该算法不适用于多任务的自动推理问题。连接树推理算法在一次消息传播过程结束，连接树进入一致状态之后可以有效地处理自动推理问题的多任务，但连接树推理算法存在连接树构造困难等问题。本文结合两类算法的优点，提出了一种桶树推理算法 BJTR。

桶树推理算法 BJTR 在根据算法 BE 按有序序列 $d=\langle x_1', x_2', \cdots, x_n'\rangle$ 进行消元所产生的桶与桶之间建立一种树结构 $\langle T, \chi, \varphi, \psi, M\rangle$。$T$ 中的每一个节点 v_i 对应于算法 BE 消元过程中一个桶 $B_{X_i'}$；假设 G_d' 为算法 BE 在其消元推理过程中产生的导出图，则 $\chi(v_i)$ 为 x_i' 和 x_i' 在 G_d' 中的所有邻居前驱节点构成的节点子集；$\varphi(v_i)$ 为 F 中元素 f_i 构成的函数子集，其中 f_i 满足其定义域在 d 中序号最大的变量为 x_i'；$\psi(v_i)$ 将被初始化为 $\psi(v_i)=\bigotimes_{f_i\in\varphi(v_i)} f_i$；$v_i$ 在连接树 T 中的父节点为 x_i' 在 G_d' 最邻近的邻居前驱节点 x_j' 在 T 中对应的节点 v_j，$C_{ij}=\chi(v_i)\cap\chi(v_j)$ 为连接树 T 中连接节点 v_i 和 v_j 的边对应的割集，M 表示在连接树边上传播的消息。

定理 2 桶树推理算法 BJTR 中建立的树结构是一种连接树结构。

证明：①BJTR 算法中桶树的每一个节点 v_i 对应的变量集 $\chi(v_i)$ 为 x_i' 和 x_i' 在导出图 G_d' 中的邻居前驱节点集；②BJTR 算法中桶树的每一个节点 v_i 对应的 $\varphi(v_i)$ 为 F 中满足其定义域在 d 中序号最大的变量为 x_i' 的 f_i 构成的子集；根据①和②，可知 BJTR 算法中所建立的树结构满足连接树定义 3 中的第①个条件，即：$\forall f_i\in F$，存在唯一 $v_i\in V$，满足 $f_i\in\varphi(v_i)$ 且 $S_i\subseteq\chi(v_i)$；③证明针对 BJTR 算法树结构 T 中的任一连接节点 v_i 和 $v_j(j<i)$ 的路径，如果变量 x_k' 同时属于 v_i 和 v_j，则 x_k' 属于树 T 中该路径上的任一节点和割集。因为 x_k' 属于节点 v_i 和 v_j，则 x_k' 一定不是在节点 v_i 中消元的，则 x_k' 属于 v_i 和其父节点 v_m 的割集 C_{im}，根据桶树的构造过程，则 x_k' 一定属于连接割集 C_{im} 的另一子团 v_m，如果 v_m 即为 v_j，则得证；如果 v_m 不为 v_j，则同理 x_k' 一定属于树结构中 v_m 的父节点 v_n 和相应割集 C_{mn}，一直进行下去，直到父节点为 v_j 为止，即：BJTR 算法中建立的树结构满足连接树定义 3 中的第②个条件；定义 3 中的第③个条件显然是满足的。故得证。

桶树推理算法 BJTR 在基于桶消元推理算法 BE 进行消元推理产生的桶之间建立一种连接树结构，并采用算法 JTR 的消息传播方式进行消息传播，使桶树结构进入一致状态后进行自动推理问题的求解。

算法 BJTR 的描述如下：

输入：自动推理问题 $P=\langle X, D, F, \otimes, \widetilde{\otimes}, \Downarrow, \{Z_1, Z_2, \cdots, Z_t\}\rangle$，变量集的一个有序序列 $d=\{x_1', x_2', \cdots, x_n'\}$ 及该序列对应的导出图 G_d'。

输出：一致状态下的连接树 $\langle T, \chi, \varphi, \psi, M\rangle$ 及自动推理问题的求解。

①初始化：根据算法 BE 按有序序列 d 消元求解过程中所产生的桶建立连接树 $\langle T, \chi, \varphi, \psi, M\rangle$；

②采用 JTR 算法的步骤②、步骤③进行两个阶段的消息传播，使连接树进入一致状态，并按 JTR 算法的步骤④进行自动推理问题的求解。

针对图 1 所示的贝叶斯网络，图 5 为 BJTR 算法按消元顺序 $d=\langle x_1, x_2, \cdots, x_6\rangle$ 进行两次消息传播的过程及消息传播结束之后一致状态下的连接树。根据一致状态下的连接树，在不同的连接树结点中可以完成对所有节点的信度更新、最大概率解释等推理任务。

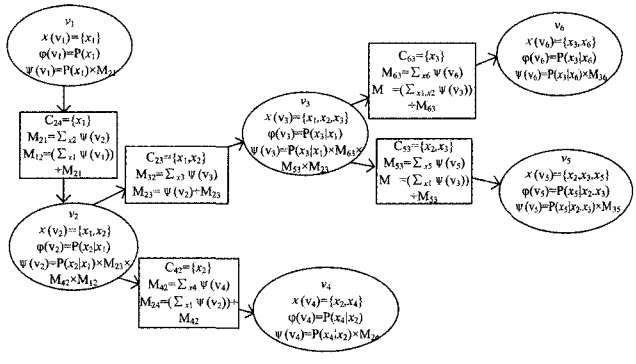

图 5 一致状态下的桶树

定理 2 证明了算法 BJTR 建立的树结构是一种连接树结构，又 BJTR 采用 JTR 算法的消息传播方式，故算法 BJTR 是连接树消元推理算法 JTR 的一个实例。

定理 3 假设自动推理问题 P 中变量集 X 包含的变量个数为 n，表 F 的长度为 r，桶树推理算法 BJTR 求解问题 P 过程中产生的连接树的节点个数为 n^*，连接树的宽度为 w^*，相应导出图的导出宽度为 w，则 BJTR 算法的时间复杂度为 $O((n+r)\cdot\exp(w+1))$，空间复杂度为 $O(n\cdot\exp(w+1))$。

证明：BJTR 算法为 JTR 算法的一个实例，而 JTR 算法的时间复杂度为 $O((n^*+r)\cdot\exp(w^*))$，空间复杂度为 $O(n^*\cdot\exp(w^*))$，则 BJTR 算法的时间复杂度为 $O((n^*+r)\cdot\exp(w^*))$，空间复杂度为 $O(n^*\cdot\exp(w^*))$。故得证。

针对同一自动推理问题和相同消元次序，连接树推理算法中的连接树宽度 w^* 和桶消元推理算法中导出图的导出宽度 w 之间满足 $w^*=w+1$，又桶树推理算法 BJTR 连接树中节点个数 n^* 与桶消元推理算法的桶个数是一致的，因此 BJTR 算法中连接树的节点个数和该推理问题 P 中变量集 X 包含的变量个数均为 n，即是一致的，满足：$n^*=n$，则 BJTR 算法的时间复杂度为 $O((n+r)\cdot\exp(w+1))$，空间复杂度为 $O(n\cdot\exp(w+1))$。故得证。

4 算法 BJTR 的性能分析

文献[14]提出了一种连接树推理算法 CTE，并基于该算法提出了一种桶树推理算法 BTE，算法 BTE 的时间复杂度为 $O((n+r)\cdot deg\cdot\exp(w+1))$，空间复杂度为 $O(n\cdot\exp(sep))$，又桶树推理算法中的割集最大长度 sep 和连接树宽度 w^* 之间满足 $sep=w^*-1=w$，则 BTE 算法的空间复杂度为 $O(n\cdot\exp(w))$。针对图模型表示的自动推理问题中的单

任务和多任务，表1给出了BE算法、BTE算法和BJTR算法的时空性能比较，从表1可以看出，针对自动推理问题中的单任务，算法BJTR的时间性能与BE算法相当，但明显优于BTE算法，空间性能略有下降；针对自动推理问题中的多任务，与BE和BTE算法相比，BJTR算法的空间性能略有下降，但其时间复杂度是BE算法时间复杂度的$1/n$，是BTE算法时间复杂度的$1/deg$，BJTR算法均具有明显的时间性能优势，因此针对多任务的自动推理问题，相对于BE和BTE算法，BJTR算法的时间效率能得到明显改善，这一改善在实时的针对所有变量的优化搜索问题中对搜索效率的提高是非常重要和明显的。

表1 BE、BTE和BJTR的时空性能比较

任务 / 算法	单节点的查询		所有节点的查询	
	时间复杂度	空间复杂度	时间复杂度	空间复杂度
BE算法	O((n+r)·exp(w+1))	O(n·exp(w))	O(n·(n+r)·exp(w+1))	O(n·exp(w))
BTE算法	O((n+r)·deg·exp(w+1))	O(n·exp(w))	O((n+r)·deg·exp(w+1))	O(n·exp(w))
BJTR算法	O((n+r)·exp(w+1))	O(n·exp(w+1))	O((n+r)·exp(w+1))	O(n·exp(w+1))

本文在标准贝叶斯网络Asia上[15]，根据证据节点B和E完成网络中所有非证据节点的信度更新需求，表2记录了BE、BTE和BJTR算法完成这一任务所需要的加、乘和除3种基本运算次数。表2中(×,÷,+)栏记录的是乘、除和加3种基本运算次数，Total栏对应的是3种基本运算次数之和。表2第一列中单节点对应以该节点为节点序的第一个元素的5个不同的节点序列，相应行中的运算次数为根据这5种不同的节点序列进行信度更新所需要的不同运算次数的均值；最后一行第一列的节点序列为该网络中当前所有非证据节点的一个序列。

表2 BE、BTE和BJTR的运算量比较

	BE算法		BTE算法		BJTR算法	
	(×,÷,+)	Total	(×,÷,+)	Total	(×,÷,+)	Total
X	(116,—,42)	158	(468,—,136)	604	(200,38,148)	386
S	(92,—,36)	128	(362,—,82)	444	(164,36,76)	276
A	(74,—,32)	106	(314,—,84)	398	(132,30,92)	254
D	(56,—,24)	80	(210,—,72)	282	(100,22,66)	188
T	(114,—,48)	162	(510,—,130)	640	(210,42,126)	378
L	(72,—,30)	102	(328,—,96)	424	(140,28,92)	260
	和值		均值		均值	
(X,S,A,D,T,L)	(514,—,212)	736	(365,—,100)	465	(158,33,100)	291

BE算法每次只能实现对节点序中第一个节点的信度更新，表中BE算法栏所对应不同节点序的各行记录的是完成节点序中第一个节点信度更新的基本运算量，该栏的最后一行为该栏中其它各行运算次数之和，即为对网络中所有非证据节点进行信度更新的基本运算量的统计。BTE和BJTE算法通过一次消息传播可以完成对网络中所有非证据节点的信度更新，表中BTE和BJTE栏的最后一行给出的是上面各行根据不同节点序对所有非证据节点进行信度更新的运算量的均值，故表2中最后一行记录的是不同算法完成所有非证据节点信度更新的基本运算量。

从表2可以看出，针对单个变量的信度更新，BE算法所需要的运算次数是最少的，但针对多节点的信度更新(此处为6个节点)所需要的运算次数是最多的，总运算次数为736。BJTR算法所需要的运算次数是最少的，总运算次数为291。从表2还可以看出，根据各种不同的节点序完成对所有非证据节点的信度更新，BJTR算法所需要的总运算次数总是少于BTE算法所需要的总运算次数。

为了进一步分析BJTR算法的时间性能，本文根据文献[16]中的贝叶斯网络生成算法BNGenerator生成不同规模不同结构的贝叶斯网络，并选择一种节点序生成桶树结构的连接树，然后在连接树中分别采用算法BJTR算法和BTE算法进行消息传播，算法BJTR和BTE均可在邻接树处于一致状态下完成对贝叶斯网络中所有非证据节点的概率查询任务，而BE算法则需要以不同非证据节点作为消元序列的首元素进行多次消元及消息传播来完成对贝叶斯网络中所有非证据节点的概率查询任务。表3中n表示贝叶斯网络中的节点个数，m表示贝叶斯网络中不同节点的最大状态数，Par表示贝叶斯网络中不同节点的最大父节点数，e表示贝叶斯网络中的证据节点数，针对完成相应贝叶斯网络中所有非证据节点的概率查询任务，TP_1表示算法BJTR和BE所需时间之比，TP_2表示算法BJTR和BTE所需时间之比，所有比值均为在10个相同规模和类似结构的贝叶斯网络上运行时间之比的平均值。从表3可以看出，网络规模越大，相比于算法BE，算法BJTR的时间性能优势越明显；网络规模越大，网络结构越复杂，相比于BTE，算法BJTR的时间性能优势越明显。实验测试环境是Pentium® CPU3.00 MHz 2GB内存的PC机，算法实现环境为Windows XP +Matlab7.0。

表3 BE、BTE和BJTR的时间性能比较

n	m	e	Par	TP_1	TP_2
20	2	4	2	10.2	1.8
			3	9.3	2.7
			4	8.7	2.9
30	2	4	2	18.2	2.6
			3	17.5	3.3
			4	16.9	3.8
40	2	4	3	25.7	3.6
			4	25.1	4.5
			5	24.7	5.3
50	2	4	3	28.6	3.9
			4	28.1	4.7
			5	27.7	5.8

结束语 针对特定领域自动推理问题中的多任务，结合桶消元推理算法和连接树推理算法的优点，提出了一种桶树推理算法BJTR，该算法根据桶消元推理算法BE在消元过程中产生的桶与桶之间建立一种连接树结构，并采用连接树推理算法JTR的消息传播方式进行消息传播。桶树推理算法BJTR既具有连接树推理算法JTR的自动推理功能和推理效率，又能解决连接树推理算法所面临的连接树构造困难问题，同时保留了桶消元推理算法的简单、易理解、易实现、易扩展等优点。通过对算法的时空复杂度的分析显示，相对于桶消元推理算法BE，桶树推理算法BJTR在多任务需求下能够有效地提高推理效率；与同类算法BTE相比，算法BJTR在空间性能略有下降的情况下提高了其时间性能；并通过实例和实验进一步验证了算法BJTR具有良好的时间性能。在后续的工作中，将进一步研究自动推理问题的统一、高效的算法框架，并尝试将算法BJTR应用到广泛的实际问题领域，用于求解自动推理问题中的多任务。

参 考 文 献

[1] Dempster A P. Studies in Fuzziness and Soft Computing [M]. Berlin: Springer, 2008: 73-104

[2] Gottlob G, Leone N, Scarello F. A Comparison of Structural CSP Decomposition Methods [C]// Proc of the 6th Int Joint Conf on Artificial Intelligence. California: Morgan Kaufmann, 1999: 394-399

[3] 季晓慧,张健. 约束问题求解[J]. 自动化学报, 2007, 2(33): 17-23

[4] Pearl J. Causality: Models, Reasoning, and Inference [M]. Cambridge: Cambridge University Press, 2000

[5] Butz C J, Yao H, Hua S. A join tree probability propagation architecture for semantic modeling [J]. Journal of Intelligent Information Systems, 2009, 33(2): 147-158

[6] Dechter R. Bucket elimination: A Unifying Framework for Reasoning [J]. Artificial Intelligence, 1999, 113(2): 41-85

[7] Lepar V, Shenoy P P. A comparison of Lauritzen-Spiegelhalter, Hugin and Shenoy-Shafer architectures for computing marginals of probability distributions [C]// Proc of the 14th Conf on Uncertainty in Artificial Intelligence. California: Morgan Kaufmann, 1998: 328-337

[8] Park J D, Darwiche A. Morphing the Hugin and Shenoy-Shafer Architectures [C]// Proc of the 7th European conference on symbolic and quantitative approaches to reasoning with uncertainty. LNCS 2711, Berlin: Springer, 2003: 149-160

[9] Fargier H, Vilarem M. Compiling CSPs into Tree-Driven Automata for Interactive Solving[J]. Constraints, 2004, 9(4), 263-287

[10] 田凤占,张宏伟,陆玉昌,等. 多模块贝叶斯网络中推理的简化[J]. 计算机研究与发展, 2003, 40(8): 1230-1237

[11] 汪荣贵. bayes网络理论及其在目标检测中应用研究[D]. 合肥: 合肥工业大学, 2004

[12] Cozman F G. Generalizing variable-elimination in bayesian networks[C]// Proc of the Workshop on Prob reasoning in Bayesian networks at SBIA/Iberamia. 2000: 21-26

[13] Kask K, Dechter R. A general scheme for automatic generation of search heuristics from specification dependencies [J]. Artificial Intelligence, 2001, 129(1/2): 91-131

[14] Kask K, Dechter R, Larrosa J, et al. Unifying Tree Decompositions for Reasoning in Graphical Models [J]. Artificial Intelligence, 2005, 166(1/2): 165-193

[15] http://www.norsys.com

[16] Ide J S. BNGnerator, Version 0.3[EB/OL]. http://www.pmr.poli.usp.br/ltd/Software/BNGenerator/, 2010-07-17

(上接第 186 页)

表 4 中,当最小支持度为 7%时,最大频繁项集长度为 7,故有 2 到 7 长度的规则;当最小支持度为 30%时,最大频繁项集长度为 5。'Ø'表示不存在该长度的频繁项集,'0'表示该长度下频繁项集产生的最小关联规则个数为零。

从以上实验不难看出,本文提出的 GMAR 方法能大大减少规则数目。相比 SAR 方法,若挖掘出的规则中后件包含一项的过多或者前件项数很多,且包含大量的冗余、重复项,那么 SAR 方法并不能有效地减少规则数目和进行规则项集的简化。GMAR 方法却克服了这一点,始终会输出最简的、无冗余的规则集合。

结束语 本文通过分析传统关联规则挖掘方法在规则提取子问题中的不足,提出基于项集依赖关系的最小关联规则挖掘方法。该方法用较少的规则实现了原始规则集的一种近似无损表述;并且,较 SAR 方法而言,本文提出的方法能更进一步地减少规则数目,从而找到更加简洁的、规则的前件和后件均不包含冗余项的关联规则。然而,由于目前生成的频繁项集仍然过多,接下来将着重研究该方法在闭包频繁项集挖掘模式下的应用和如何利用最小关联规则集推演原始规则集,实现规则集的无损表述等问题。

参 考 文 献

[1] Agrawal R, Imielinski T, Swami A N. Mining association rules between sets of items in large databases[C]// Proceedings of ACM SIGMOD International Conference on Management of Date. Washington DC, 1993: 207-216

[2] Loo K K, Yip C-I, Kao Ben, et al. A lattice-based approach for I/O efficient association rule mining [J]. Information Systems, 2002, 27(1): 41-74

[3] Lin T Y, Hu Xiao-hua, Louie E. A fast association rule algorithm based on bitmap and granular computing[C]// Proceedings of IEEE International Conference on Fuzzy Systems. 2003: 678-683

[4] 任永功,宋奎勇,寇香霞. 一种结合散列与位表挖掘频繁项目集算法[J]. 计算机科学, 2010, 37(12): 145-148

[5] Stanisic P, Tomovic S. Apriori multiple algorithm for mining association rules[J]. Information Technology and Control, 2008, 37(4): 311-320

[6] Aouad L M, Le-Khac N A, Kechadi T M. Performance study of distributed Apriori-like frequent itemsets mining[J]. Knowledge and Information Systems, 2010, 23(1): 55-72

[7] 陈茵,闪四清,刘鲁,等. 最小冗余的无损关联规则集表述[J]. 自动化学报, 2008, 34(12): 1490-1496

[8] Geng Li-qiang, Hamilton H J. Interestingness measures for data mining: a survey[J]. ACM Computing Surveys, 2006, 38(3): 1-33

[9] Christian H W. Statistical mining of interesting association rules [J]. Statistics and computing, 2008, 18(2): 185-194

[10] Li Gui-chong, Hamiltion H J. Basic association rules [C]// Proceeding of the 4th SIAM International Conference on Data Mining. Orlando, USA, 2004: 166-177

[11] Chen Guo-qing, Wei Qiang, Liu De, et al. Simple association rules(SAR) and the SAR-based rule discovery[J]. Computers and Industrial Engineering, 2002, 43(4): 721-733

[12] Mohammed J Z. Non-Redundant Association Rules [J]. Data Mining and Knowledge Discovery, 2004, 9(3): 223-248

[13] Louie E, Lin T Y. Finding association rules using fast bit computation: machine-oriented modeling[C]// Proceedings of International Symposium ISMIS2000. Charlotte, North, Lecture Notes in Artificial Intelligence, 2000: 486-494