基于元信息的粗糙集规则并行挖掘方法

苏 健 高 济

(浙江大学人工智能研究所 杭州 310027)

A Meta-information-Based Method for Rough Sets Rule Parallel Mining

SU Jian GAO Ji

(Institute of Artificial Intelligence, Zhejiang University, Hangzhou 310027)

Abstract Rough sets is one important method of data mining. Data mining processes such a great quantity of data in large database that the speed of Rough Sets Data Mining Algorithm is critical to Data Mining System. Utilizing network computing resources is an effective approach to improve the performance of Data Mining System. This paper proposes the concept of meta-information, which is used to describes the result of Rough Sets Data Mining in information system, and a meta-information-based method for rule parallel mining. This method decomposes the information-system into a lot of sub-information-system, dispatchs the task of generating meta-information of sub-information-system to some task performer in the network, and lets them parallel compute meta-information, then synthesizes the meta-information of sub-information-system to the meta-information of information system in the task synthesizer, and finally produces the rule according to the meta-information.

Keywords Meta-information, Data mining, Information system, Rough sets, Parallel computing

1. 引言

在当前的信息化时代,为从大量积累的历史数据中获取有用的知识,使得数据挖掘已成为研究热点[1]。Pawlak 教授提出粗糙集合理论[2],经过众多学者的研究和完善,已成为数据挖掘的重要手段[3~8]。在大数据环境下,数据挖掘方法的速度将直接影响整个数据挖掘系统的性能,如何有效地提高数据挖掘方法的速度,是迫切需要解决的问题。

与此同时,计算机网络存在大量的运算资源,充分利用这些资源是提高数据挖掘方法速度的有效途径。为此,本文提出基于元信息的粗糙集规则并行挖掘方法,该方法将信息系统分解为若干个信息子系统,从而信息系统的挖掘任务分割为网络中各个运算节点上的信息子系统的挖掘子任务,各子任务进行并行计算,最终将这些挖掘子任务的挖掘结果合成为信息系统的挖掘结果,并推导出粗糙集规则。显然,为达到上述目的,必须解决以下问题:如何描述各个信息子系统数据挖掘子任务的结果,并且该结果可合成为信息系统数据挖掘任务的结果。为此,本文提出以元信息描述信息子系统数据挖掘子任务的结果。元信息由等价类矩阵、条件属性等价类表示集合和决策属性等价类表示集合等构成:等价类矩阵描述信息系统中条件属性等价类与决策属性等价类的包含关系,条件与决策属性的等价类表示集合能够描述信息系统中条件与决策属性等价类族。后文将给出元信息的合成算法。元信息合成算法包括等价类表示集合合成算法与等价类矩阵的合成算法。各个信息子系统的元信息生成方法将采用渐增式生成算法[9]。

数据库中数据是动态增长的,粗糙集方法必须具备处理动态数据的能力。由于本文提出的方法通过将信息系统分为若干个信息子系统(即将数据分割为若干个部分),在并行计算所得到的元信息的基础上再进行合成,因而本质上支持动态数据的处理。这些后来动态增长的数据可视为新的信息子系统的数据集,只需将该信息子系统的元信息生成后再合成到已有的信息系统元信息中。

本方法通过将粗糙集数据挖掘方法分为两个阶段(即元信息生成阶段和规则生成阶段),将元信息生成任务与规则生成任务分离开,这样可以将现有的粗糙集规则处理方法(如规则约简等)加以改造,应用到元信息上,从而扩展了本方法的应用。这些将另文发表。

以本方法为基础,我们开发出粗糙集并行挖掘系统。该系统将信息系统分解为若干个信息子系统,将生成信息子系统元信息的任务分配给网络上的若干个任务执行器,然后在任务合成器上将这些元信息合成为信息系统的元信息,并根据该元信息生成规则。

本文首先介绍元信息等重要定义;再阐述元信息合成算法;然后在这些算法的基础上,构造规则并行挖掘系统,并分析该系统的实验结果;最后给出结论。

2. 信息系统元信息

首先介绍粗糙集合的若干重要定义,再分别阐述等价类矩阵和等价类表示的定义,然后在此基础上提出元信息的定义。

2.1 信息系统与信息子系统

定义 2.1 信息系统是一个四元组[4],$S=\langle U,A,V,F\rangle$,其中:I. $U=\{u_1,u_2,\cdots,u_n\}$,$n=|U|$,且 $n>0$,U 是对象集合;II. $A=C\cup D$。A 是属性集合,其中 C 是条件属性集合,D 是决策属性集合;III. $V=\bigcup_{P\in A}V_P$,V_P 是属性 P 的值域;IV. $F:U\times$

苏 健 博士研究生,主要从事数据挖掘、决策支持等领域研究。高 济 教授,博士生导师,主要从事人工智能、数据仓库、软件工程等领域的研究。

$A\to V$，对于$\forall\ x\in U, q\in A, f(x,q)\in V_q$。

Ziarko 提出可变精度粗糙集模型 VPRS[3]，该模型以集合间的部分包含关系取代经典粗糙集模型的绝对包含关系，以允许噪音数据的存在，从而拓广了粗糙集理论的应用范围。部分包含关系定义为：

$Y\overset{\beta}{\subseteq}X$，当且仅当 $c(X,Y)\leqslant\beta$

其中：$c(X,Y)=\begin{cases}1-|X\cap Y|/|X| & \text{如果}|X|>0\\ 0 & \text{如果}|X|=0\end{cases}$

若 $\beta<0.5$，则 $Y\overset{\beta}{\subseteq}X$ 为多数包含关系。

定义 2.2 信息系统 $S=(U,A,V,F)$，$U_1\subseteq U$，则称 $S_1=(U_1,A,V,F)$是 S 的信息子系统。

由信息系统的定义可知，属性集合 $A=C\cup D$，其中 C 是条件属性集合，D 是决策属性集合。R_C 和 R_D 分别为属性集合 C 和 D 的等价关系，若不指定下标，则关系 R 泛指某一属性集合上的等价关系。而等价类族 C^* 和 D^* 分别是基于 R_C 和 R_D 的等价划分，即 $C^*=U/R_C$，$D^*=U/R_D$。

2.2 等价类矩阵、等价类表示与元信息

定义 2.3 信息系统 $S=(U,A,V,F)$，$C^*=U/R_C$，$D^*=U/R_D$，记 $C^*=\{X_1,X_2,\cdots,X_n\}$，$D^*=\{Y_1,Y_2,\cdots,Y_m\}$，则 S 的等价类矩阵 $M=(m_{ij})$，其中 $m_{ij}=|X_i\cap Y_j|$，$1\leqslant i\leqslant|C^*|$，$1\leqslant j\leqslant|D^*|$。

等价类矩阵是衡量条件属性等价类与决策属性等价类交集的数字度量。

定义 2.4 论域 U，$U^*=U/R$ 是 U 的等价划分，$X\in U^*$，则称 $D_X=(x,|X|)$是 X 的等价类表示，其中 $x\in X$ 是 X 的特征元素。

等价类中，任何一个元素都可以代表其它元素，因而等价类表示取等价类的任何一个元素作为特征元素，同时记录等价类元素的数量以衡量等价类的大小。本文的并行生成方法主要是利用网络资源，减少网络通讯量可以提高系统的运行效率。用等价类表示来描述等价类，既减少了网络通讯量，又减少了系统处理时间，从而提高系统处理速度。

定义 2.5 信息系统 $S=(U,A,V,F)$，$C^*=U/R_C$，$D^*=U/R_D$，S 的元信息 I 是三元组，$I=(M,F_C,F_D)$，其中：M 是信息系统 S 的等价类矩阵；F_C 是条件属性等价类表示集合，即 $F_C=\{D_X|X\in C^*\}$，F_D 是决策属性等价类表示集合，即 $F_D=\{D_Y|Y\in D^*\}$。

信息系统的元信息是本文的核心。它是信息系统的描述，下面定理 1 证明从元信息能够推导出信息系统的 β 误差规则。

定义 2.6 信息系统 $S=(U,A,V,F)$，$A=C\cup D$，$C^*=U/R_C$ 是条件等价类族，$D^*=U/R_D$ 是决策等价类族，若$\exists\ X\in C^*$，$\exists\ Y\in D^*$，且 $X\overset{\beta}{\subseteq}Y$ 则称 X，Y 存在 β 误差规则：

$$(c_1,v_{c_1})\wedge(c_2,v_{c_2})\wedge\cdots\wedge(c_n,v_{c_n})\to(d_1,v_{d_1})\wedge(c_2,v_{d_2})\wedge\cdots\wedge(c_m,v_{d_m})。$$

其中，$n=|C|$，$m=|D|$。并称 $|X|/|U|$ 为规则的支持度，记为 $\text{support}(X,Y)=|X|/|U|$。

规则的支持度可以衡量规则的重要程度。

定理 1 信息系统 S 的元信息 $I=(M,F_C,F_D)$，$M=(m_{ij})$，$F_C=\{D_{X_i}|X_i\in C^*\}$，$F_D=\{D_{Y_j}|Y_j\in D^*\}$，对于给定的误差 β，若 $m_{ij}/|X_i|\geqslant1-\beta$，则 X_i，Y_j 确定一条 β 误差规则且 $\text{support}(X_i,Y_j)=|X_i|/|U|$。

其中：$X_i\in C^*$ 是 C^* 的第 i 个等价类，$Y_j\in D^*$ 是 D^* 的第 j 个等价类，$|X_i|$ 是 $D_{X_i}\in F_C$ 的第二元，$|U|=\Sigma|Y_j|$，$|Y_j|$ 是 $D_{Y_j}\in F_D$ 的第二元。

证明：记 $C^*=\{X_1,X_2,\cdots,X_m\}$，$D^*=\{Y_1,Y_2,\cdots,Y_n\}$。

$m_{ij}/|X_i|\geqslant1-\beta\Leftrightarrow|X_i\cap Y_j|/|X_i|\geqslant1-\beta\Leftrightarrow X_i\overset{\beta}{\subseteq}Y_j$，$\therefore m_{ij}$确定一条 β 误差规则，显然 $|U|=\Sigma|Y_j|=\Sigma|X_i|$，$\therefore\text{support}(X_i,Y_j)=|X_i|/|U|$。证毕。

由定理 1 可知，只要获取到元信息，就可以直接推导出规则，因此如何高效地生成元信息是提高规则挖掘的关键。本文提出的并行挖掘方法就是通过将各信息子系统生成的元信息合成为信息系统的元信息，并通过元信息推导出粗糙集规则。

3. 元信息的合成

元信息的合成包括等价类表示集合的合成和等价类矩阵的合成。前者将信息子系统元信息中的等价类表示集合合成为信息系统的等价类表示集合；后者是信息子系统元信息的等价类矩阵合成为信息系统的等价类矩阵。

当我们获得两个子信息系统的元信息时，就将得到这两个元信息的等价类表示集合，我们要合成元信息，就需要将两个元信息的等价类表示集合合成为信息系统元信息的等价类表示集合。设信息系统 $S=(U,A,V,F)$，它由 $S_1=(U_1,A,V,F)$，$S_2=(U_2,A,V,F)$两个信息子系统组成，$U_1\cap U_2=\phi$ 且 $U_1\cup U_2=U$。对某一等价关系 R，若已知 U_1 和 U_2 的等价类表示集合 $F_1=\{D_X\}$和 $F_2=\{D_Y\}$，需要将 F_1 和 F_2 合成为 U 的等价类表示集合 $F=\{D_Z\}$。

记 $D_X=(x,|X|)$，$D_Y=(y,|Y|)$，U_1 的等价划分 U_1/R，U_2 的等价划分 U_2/R，U 的等价划分 U/R。

命题 3.1 (1)对 $D_X\in F_1$，若$\exists\ D_Y\in F_2$，使得 xRy，则 $Z=X\cup Y\in U/R$，$D_Z=(x,|X|+|Y|)$；(2)对 $D_X\in F_1$，若$\forall\ D_Y\in F_2$，xRy 都不成立，则 $Z=X\in U/R$，$D_Z=(x,|X|)$；(3)对 $D_Y\in F_2$，若$\forall\ D_X\in F_1$，xRy 都不成立，则 $Z=Y\in U/R$，$D_Z=(y,|Y|)$。

证明：(1)$\because xRy$，$\therefore\exists\ Z\in U/R$，使得 $X\cup Y\subseteq Z$；因此，只需证明对$\forall\ z\in Z$，$z\in X$($z\in U_1$ 时)或者 $z\in Y$($z\in U_2$ 时)。若 $z\in U_1$，$\because X\in U_1/R$，$\therefore$对$\forall\ E\in U_1/R, E\neq X$，记 $D_E=(e,|E|)$，eRx 均不成立$\therefore eRz$ 不成立，$\therefore z\in X$。若 $z\in U_2$，同理可证 $z\in Y$，$\therefore Z=X\cup Y\in U/R$，$\therefore D_Z=(x,|X|+|Y|)$。故命题得证。

(2)、(3)易证得，证略。

算法 A (等价类表示集合合成算法)

已知：$F_1=\{D_X\}$，$F_2=\{D_Y\}$，求 $F=\{D_Z\}$

1. 任取 $D_X\in F_1$，记 $D_X=(x,|X|)$。
2. 对所有的 $D_Y\in F_2$，记 $D_Y=(y,|Y|)$，判断是否 xRy。

   若$\exists D_Y$，使得 xRy，则 $D_Z=(x,|X|+|Y|)$，$F=F\cup\{D_Z\}$，$F_2=F_2-\{D_Y\}$。

   否则，$D_Z=(x,|X|)$，$F=F\cup\{D_Z\}$。
3. $F_1=F_1-\{D_X\}$。
4. 若 $F_1=\phi$，则 $F=F\cup F_2$，结束，否则转 1。

该算法将 D_X 和 D_Y 进行比较，如果 xRy，则根据命题 3.1 的(1)，$Z=X\cup Y$ 是 U 的等价类；如果对所有的 D_Y 都不成立 xRy，则根据命题 3.1 的(2)，X 是 U 的等价类；步 4 中，当 $F_1=\phi$ 时，F_2 中剩余的等价类表示 D_Y，显然对 F_1 中的等价类表示都不成立 xRy，根据命题 3.1 的(3)，Y 是 U 的等价类。算法 A 实际上是将两个信息子系统的等价类表示集合中属于同一等价类的元素合并，并将两个等价类表示集合中其它的

元素添加进去，从而合成为信息系统的等价类表示集合，即 F 中的 D_Z 或者来自 F_1，或者来自 F_2，或者是 F_1 中 D_X 与 F_2 中 D_Y 的合成。

算法 A 可由命题 3.1 证得，证略。

算法 A 的时间复杂度是 $O(|F_1|\times|F_2|)$，即时间复杂度由 U_1 和 U_2 的等价类的数目之积决定。

设信息系统 $S=\langle U,A,V,F\rangle$，它由 $S_1=\langle U_1,A,V,F\rangle$，$S_2=\langle U_2,A,V,F\rangle$ 两个信息子系统组成，$U_1\cap U_2=\phi$ 且 $U_1\cup U_2=U$。对某一等价关系 R，U、U_1 和 U_2 的等价类表示集合分别记为 $F=\{D_Z\}$，$F_1=\{D_X\}$ 和 $F_2=\{D_Y\}$，记 $D_Z=(z,|Z|)$，$D_X=(x,|X|)$，$D_Y=(y,|Y|)$。

命题 3.2 $D_Z=(z,|Z|)$，$D_X=(x,|X|)$，$D_Y=(y,|Y|)$。存在如下关系：(1)对 $D_Z\in F$，若$\exists$ $D_X\in F_1$，使得 xRz，且$\exists$ $D_Y\in F_2$，使得 yRz 则 $Z=X\cup Y$；(2)对 $D_Z\in F$，若$\exists$ $D_X\in F_1$，使得 xRz，且$\forall$ $D_Y\in F_2$，yRz 都不成立，则 $Z=X$；(3)对 $D_Z\in F$，若$\exists$ $D_Y\in F_2$，使得 yRz，且$\forall$ $D_X\in F1$，xRz 都不成立，则 $Z=Y$。

本命题可由命题 3.1 证得，证略。

我们能够通过算法 A 将 U_1 和 U_2 的等价类表示集合 F_1、F_2 合成为 U 的等价类表示集合 F，同时，也可以通过命题 3.2 获得 F 中元素 D_Z 与 D_X、D_Y 的对应关系，即 Z 是由 X、Y 还是 $X\cup Y$ 组成的。命题 3.2 提供的对应关系有助于等价类矩阵的合成，命题 3.2 的作用将在下文阐述。

设信息系统 $S=\langle U,A,V,F\rangle$，它由 $S_1=\langle U_1,A,V,F\rangle$，$S_2=\langle U_2,A,V,F\rangle$ 两个信息子系统组成，$U_1\cap U_2=\phi$ 且 $U_1\cup U_2=U$。S_1 的元信息 $I_1=\langle M_1,F_C^1,F_D^1\rangle$，$S_2$ 的元信息 $I_2=\langle M_2,F_C^2,F_D^2\rangle$，S 的元信息 $I=\langle M,F_C,F_D\rangle$。算法 A 可以合成两个元信息的等价类表示集合，即可以将 F_C^1，F_C^2 合成为 F_C，将 F_D^1，F_D^2 合成为 F_D。根据等价类矩阵定义，记 $M=(m_{ij})$，$m_{ij}=|X_i\cap Y_j|$；$M_1=(m_{ij}^1)$，$m_{ij}^1=|X_i^1\cap Y_j^1|$；$M_2=(m_{ij}^2)$，$m_{ij}^2=|X_i^2\cap Y_j^2|$。下文将阐述如何将 M_1、M_2 合成为 M。

记 $C^*=U/R_C=\{X_1,X_2,\cdots,X_{|U/R_C|}\}$，$D^*=U/R_D=\{Y_1,Y_2,\cdots,Y_{|U/R_D|}\}$，$C_1^*=U_1/R_C=\{X_1^1,X_2^1,\cdots,X_{|U_1/R_C|}^1\}$，$D_1^*=U_1/R_D=\{Y_1^1,Y_2^1,\cdots,Y_{|U_1/R_D|}^1\}$，$C_2^*=U_2/R_C=\{X_1^2,X_2^2,\cdots,X_{|U_2/R_C|}^2\}$，$D_2^*=U_2/R_D=\{Y_1^2,Y_2^2,\cdots,Y_{|U_2/R_D|}^2\}$。根据命题 3.1 与命题 3.2 可知，将 C_1^* 与 C_2^* 合成为 C^* 后，对 $X_i\in C^*$，$X_i=X_k^1$ 或者 $X_i=X_l^2$ 或者 $X_i=X_k^1\cup X_l^2$，共 3 种情况，其中 $X_k^1\in C_1^*$，$X_l^2\in C_2^*$，同理，将 D_1^* 与 D_2^* 合成为 D^* 后，对 $Y_j\in D^*$，$Y_j=Y_p^1$ 或者 $Y_j=Y_q^2$ 或者 $Y_j=Y_p^1\cup Y_q^2$，共 3 种情况，其中 $Y_p^1\in D_1^*$，$Y_q^2\in D_2^*$。显然，$X_i\cap Y_j$ 共有 9 种组合。

命题 3.3 $M=(m_{ij})$ 与 $M_1=(m_{ij}^1)$，$M_2=(m_{ij}^2)$ 的关系如下：(1)$X_i=X_k^1$，$Y_j=Y_p^1$，则 $m_{ij}=m_{kp}^1$；(2)$X_i=X_l^2$，$Y_j=Y_p^1$，则 $m_{ij}=0$；(3)$X_i=X_k^1\cup X_l^2$，$Y_j=Y_p^1$，则 $m_{ij}=m_{kp}^1$；(4)$X_i=X_k^1$，$Y_j=Y_q^2$，则 $m_{ij}=0$；(5)$X_i=X_l^2$，$Y_j=Y_q^2$，则 $m_{ij}=m_{lq}^2$；(6)$X_i=X_k^1\cup X_l^2$，$Y_j=Y_q^2$，则 $m_{ij}=m_{lq}^2$；(7)$X_i=X_k^1$，$Y_p^1\cup Y_q^2$，则 $m_{ij}=m_{kp}^1$；(8)$X_i=X_l^2$，$Y_p^1\cup Y_q^2$，则 $m_{ij}=m_{lq}^2$；(9)$X_i=X_k^1\cup X_l^2$，$Y_p^1\cup Y_q^2$，则 $m_{ij}=m_{kp}^1+m_{lq}^2$。

证明：(1)$X_i=X_k^1$，$Y_j=Y_p^1$，则 $m_{ij}=|X_i\cap Y_j|=|X_k^1\cap Y_p^1|=m_{kp}^1$；

(2)$X_i=X_l^2$，$Y_j=Y_p^1$，则 $m_{ij}=|X_i\cap Y_j|=|X_l^2\cap Y_p^1|=|\phi|=0$；

(3)$X_i=X_k^1\cup X_l^2$，$Y_j=Y_p^1$，则 $m_{ij}=|X_i\cap Y_j|=|(X_k^1\cup X_l^2)\cap Y_p^1|=|(X_k^1\cap Y_p^1)\cup(X_l^2\cap Y_p^1)|=|(X_k^1\cap Y_p^1)\cup\phi|=m_{kp}^1$；

(4)$X_i=X_k^1$，$Y_j=Y_q^2$，则 $m_{ij}=|X_i\cap Y_j|=|X_k^1\cap Y_q^2|==|\phi|=0$；

(5)$X_i=X_l^2$，$Y_j=Y_q^2$，则 $m_{ij}=|X_i\cap Y_j|=|X_l^2\cap Y_q^2|=m_{lq}^2$；

(6)$X_i=X_k^1\cup X_l^2$，$Y_j=Y_q^2$，则 $m_{ij}=|X_i\cap Y_j|=|(X_k^1\cup X_l^2)\cap Y_q^2|=|(X_k^1\cap Y_q^2)\cup(X_l^2\cap Y_q^2)|=|\phi\cup(X_l^2\cap Y_q^2)|=m_{lq}^2$；

(7)$X_i=X_k^1$，$Y_p^1\cup Y_q^2$，则 $m_{ij}=|X_i\cap Y_j|=|X_k^1\cap(Y_p^1\cup Y_q^2)|=|(X_k^1\cap Y_p^1)\cup(X_k^1\cap Y_q^2)|=(X_k^1\cap Y_p^1)\cup\phi|=m_{kp}^1$；

(8)$X_i=X_l^2$，$Y_p^1\cup Y_q^2$，则 $m_{ij}=|X_i\cap Y_j|=|X_l^2\cap(Y_p^1\cup Y_q^2)|=|(X_l^2\cap Y_p^1)\cup(X_l^2\cap Y_q^2)|=|\phi\cup(X_l^2\cap Y_q^2)|=m_{lq}^2$；

(9)$X_i=X_k^1\cup X_l^2$，$Y_p^1\cup Y_q^2$，则 $m_{ij}=|X_i\cap Y_j|=|(X_k^1\cup X_l^2)\cap(Y_p^1\cup Y_q^2)|=|(X_k^1\cap(Y_p^1\cup Y_q^2))\cup(X_l^2\cap(Y_p^1\cup Y_q^2))|=|(X_k^1\cup Y_p^1)\cup(X_k^1\cap Y_q^2)\cup(X_l^2\cap Y_p^1)\cup(X_l^2\cap Y_q^2)|=|(X_k^1\cap Y_p^1)\cup\phi\cup\phi\cup(X_l^2\cap Y_q^2)|=m_{kp}^1+m_{lq}^2$

证毕。

命题 3.3 给出了等价类矩阵 $M=(m_{ij})$ 与 $M_1=(m_{ij}^1)$，$M_2=(m_{ij}^2)$ 的关系。根据这些关系，可以将 M_1、M_2 合成为 M。由于 $X_i\cap Y_j$ 共有 9 种组合，在应用命题 3.3 所提供的结论前，我们需要先判断 $X_i\cap Y_j$ 属于哪种组合，然后才能应用相应子命题的结论。前文命题 3.2 已经提供了判断的手段。根据命题 3.2，对 $X_i\in C^*$，通过等价类表示集合 F_C^1、F_C^2 和 F_C，可以判断 $X_i=X_k^1$ 或者 $X_i=X_l^2$ 或者 $X_i=X_k^1\cup X_l^2$；同理，对 $Y_j\in D^*$，通过等价类表示集合 F_D^1、F_D^2 和 F_D，可以判断 $Y_j=Y_p^1$ 或者 $Y_j=Y_q^2$ 或者 $Y_j=Y_p^1\cup Y_q^2$。

结合命题 3.2 和命题 3.3 就可以合成等价类矩阵，下面的算法 B 用于合成元信息，它首先利用算法 A 合成等价类表示集合，再合成等价类矩阵。设信息系统 $S=\langle U,A,V,F\rangle$，它由 $S_1=\langle U_1,A,V,F\rangle$，$S_2=\langle U_2,A,V,F\rangle$ 两个信息子系统组成，$U_1\cap U_2=\phi$ 且 $U_1\cup U_2=U$。S_1 的元信息 $I_1=\langle M_1,F_C^1,F_D^1\rangle$，$S_2$ 的元信息 $I_2=\langle M_2,F_C^2,F_D^2\rangle$，S 的元信息 $I=\langle M,F_C,F_D\rangle$。

算法 B(元信息合成算法)

已知：I_1，I_2，求 I。

1. 利用算法 A，将 F_C^1、F_C^2 合成为 F_C，将 F_D^1、F_D^2 合成为 F_D；
2. 将等价类矩阵 M_1、M_2 合成为 M。
   (1)i=0，j=0；
   (2)对 $D_{X_i}\in F_C$，考察 F_C^1，F_C^2，确定是否存在 $D_{X_k^1}\in F_C^1$，$D_{X_l^2}\in F_C^2$，使得 $X_i=X_k^1$ 或者 $X_i=X_l^2$ 或者 $X_i=X^1{}_k\cup X_l^2$
   (3)对 $DY_j\in F_D$，考察 F_D^1，F_D^2，确定是否存在 $D_{Y_p}^1\in F_D^1$，$D_{Y_q}^2\in F_D^2$，使得 $Y_j=Y_p^1$ 或者 $Y_j=Y_q^2$ 或者 $Y_j=Y_p^1\cup Y_q^2$
   (4)如果 $X_i=X_k^1$，$Y_j=Y_p^1$，则 $m_{ij}=m_{kp}^1$；否则如果 $X_i=X_l^2$，$Y_j=Y_p^1$，则 $m_{ij}=0$；否则如果 $X_i=X_k^1\cup X_l^2$，$Y_j=Y_p^1$，则 $m_{ij}=m_{kp}^1$；否则如果 $X_i=X_k^1$，$Y_j=Y_q^2$，则 $m_{ij}=0$；否则如果 $X_i=X_l^2$，$Y_j=Y_q^2$，则 $m_{ij}=m_{lq}^2$；否则如果 $X_i=X_k^1\cup X_l^2$，$Y_j=Y_q^2$，则 $m_{ij}=m_{lq}^2$；否则如果 $X_i=X_k^1$，$Y_p^1\cup Y_q^2$，则 $m_{ij}=m_{kp}^1$；否则如果 $X_i=X_l^2$，$Y_p^1\cup Y_q^2$，则 $m_{ij}=m_{lq}^2$；否则如果 $X_i=X_k^1\cup X_l^2$，$Y_p^1\cup Y_q^2$，则 $m_{ij}=m_{kp}^1+m_{lq}^2$。
   (5)j=j+1，如果 $j\leqslant|F_D|$ 转(3)
   (6)i=i+1，如果 $i\leqslant|F_C|$ 转(2)
3. 输出 I。

算法 B 可以通过命题 3.2 和命题 3.3 证得，证略。

算法 A 的时间复杂度为 $O(|F_C|*|F_D|)$，因为 $|F_C|\leqslant|F_C^1|+|F^2C|$ 且 $|F_D|\leqslant|F_D^1|+|F_D^2|$，所以算法 B 的时间复杂度可记为 $O((|F_C^1|+|F_C^2|)*(|F_D^1|+|F_D^2|))$。

元信息的合成算法主要用于将网络上各个运算节点对信

息子系统挖掘所产生的元信息合成为信息系统元信息。同时，该合成算法也可以用于信息子系统元信息的生成。令 $U=U_1 \cup U_2$，$U_2=\{x\}$，信息系统 $S=(U,A,V,F)$，S 的元信息 I，信息子系统 $S_1=(U_1,A,V,F)$，$S_2=(U_2,A,V,F)$，S_1 元信息 $I_1=(M_1,F_C^1,F_D^1)$，S_2 元信息 $I_2=(M_2,F_C^2,F_D^2)$。由于 S_2 是单个元素的集合，其元信息 I_2 可以直接获得。应用合成算法将 I_1、I_2 合成为 I 的过程实际上是 I 的渐增式生成过程。当然，我们另外还需研制出元信息渐增式生成算法，专门用于各信息子系统的元信息的生成[9]。

4. 系统设计和实验

基于元信息的粗糙集规则并行挖掘方法，首先将信息系统分解为若干个信息子系统，这些信息子系统的元信息生成之后，再将它们两两合成为信息系统的元信息，然后根据元信息推导出规则。本节首先给出信息系统的分解策略，再给出系统结构的描述与任务调度方法，然后在此系统的基础上给出实验数据和分析。

4.1 信息系统的分解策略

本文中，信息系统 $S=(U,A,V,F)$ 是数据挖掘的数据源，它是一个逻辑概念。具体到数据，它可以是多个同构或异构数据库的数据表或者特定格式的文件。在数据分布上，可以是集中式或者分布式。本方法所采取的分解策略是：(1)将处在不同物理位置的数据分割为不同的信息子系统；(2)将异构的数据分割为不同的信息子系统；(3)将数据量大的数据分割为若干个小颗粒的信息子系统。

策略(1)主要是为了减少信息子系统元信息生成的复杂度，使得每个元信息的生成只针对某一主机上的数据。策略(2)主要是为了增强异质数据的处理能力，由于同时处理多种异质数据会导致系统过于复杂，且会降低系统的可扩展性，我们针对各种不同类型的数据，采用不同的数据存取方式以便于元信息生成，并且每一种类型数据的处理程序都采用插件(plug-in)的形式集成到系统中，这样便于系统的扩展，以支持更多的数据类型。由于异构数据的挖掘结果采用的是统一的元信息格式，因而在元信息合成时，已经屏蔽了数据源的数据类型，可以有效地合成元信息。策略(3)将数据量大的数据分割为若干个小颗粒的信息子系统，主要是为了将大数据量的元信息生成任务分解为多个小数据量的生成任务，从而提高任务的并行度，减少任务完成所需要的时间。

4.2 系统结构与任务调度

定义 4.1 任务是信息系统的某个子系统的元信息的生成过程。其 BNF 定义如下：

⟨Task⟩::=⟨Task_Id⟩⟨Sub_Info_System_Id⟩{⟨Object_Space_Range⟩}*

其中：⟨Task_Id⟩是任务的唯一标识；⟨Sub_Info_System_Id⟩是信息子系统的唯一标识；⟨Object_Space_Range⟩是信息子系统的对象空间的范围。信息系统分解为信息子系统时，信息系统中的对象集合分割为几个子集，由于对象集合中每一个对象都有唯一标识(如主键)，该标识的范围可确定对象空间的范围，即决定对象空间是如何分割的。

从任务的角度，系统由 3 部分组成：任务调度器、任务执行器和任务合成器(见图 1)。

任务调度器的作用是将信息系统的对象空间分解为信息子系统对应的对象子空间，并将生成子元信息的任务分配给任务执行器。任务调度器拥有当前任务执行器列表，在任务调度器分配任务时，它首先轮询任务执行器，以确定执行器的当前转态。

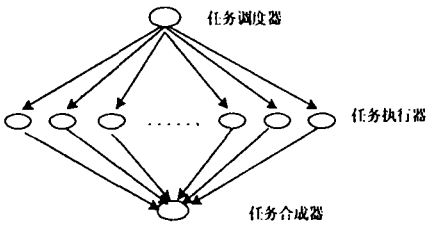

图 1 系统结构图

任务执行器的作用是应用元信息增式生成算法生成信息子系统元信息。任务调度器在初始化时将会在任务调度器中等记它的状态。任务执行器的状态有：就绪、执行和不可用。就绪状态下，任务执行器可以执行任务；执行状态表示正在执行任务；不可用状态表示出现异常情况。

任务合成器的作用是将各个信息子系统元信息合成为元信息，并生成规则。

任务执行器的数量有一个或一个以上，并行度随任务执行器的数量的增加而增加。任务调度器和任务合成器一般位于同一台主机上。任务调度器定时向任务执行器轮询状态，若任务执行器长时间没有响应，则认为任务执行器出现异常情况，此时若有任务调度器处于就绪状态，任务调度器就将任务重新分配给就绪的任务执行器。

4.3 实验与分析

本系统的实验以 Oracle 8 数据库为数据源，对 900k 条记录进行数据挖掘，共有 8 个条件属性和 1 个决策属性。网络是 10M/100M 以太网，以下是实验结果：

表 1 实验结果

任务执行器数	执行时间(s)	比例
1	341	100
2	223	65.4
3	146	42.8
4	119	34.9

从实验的结果可以看出，随着任务执行器数量的增加，执行时间呈下降趋势。然而，执行时间并没有与任务执行器数量成线性比例的下降。主要原因是任务执行器的运算能力并不一致，且任务执行器的增加导致数据库服务器的运算负荷的加重，使得数据库服务器成为运算的瓶颈。任务执行器的运算能力的不一致，可以通过以下方法改善系统性能：将任务分割为颗粒较小的任务，依次将小任务分配给就绪(空闲)的任务执行器。这样，运算能力大的执行器将分配较多的小任务，从而提高整体运算速度。可以采用高性能数据库服务器(如多处理器，集群等)，以缓解因任务执行器数增多而造成的数据库服务器运算压力。由于网络普遍采用 10M 或 100M 以上的快速网络，且多在局域网内计算，网络传输的延迟并不影响太多的性能。

获得元信息后，再设定相应的误差与支持度阈值，就可以从元信息中筛选出合适的规则，组成决策表。这样就可以利用规则约简技术来处理这些规则。

结论 基于元信息的粗糙集规则并行挖掘方法旨在利用计算机网络计算资源，以提高粗糙集数据挖掘速度。它通过将数据挖掘的任务分解为多个子任务，然后将这些子任务分配

给网络计算环境下的任务执行器，并令其进行并行计算，从而达到提高数据挖掘速度的目的。并且，由于本方法本质上支持动态增长的、分布式的、异质的数据，因此可适用于大数据库的数据挖掘。需要指出的是，本文提出的方法同样适用于单机系统。在具有多个处理器的单机系统中，本方法也能够有效地提高数据挖掘的速度。本方法经实验证明，利用丰富的计算机网络计算资源进行数据挖掘，是提高粗糙集规则挖掘系统性能的一个有效途径。

本文仅讨论粗糙集规则的并行挖掘方法，至于在这些规则基础上的规则约简等相关内容，将另文发表。

参考文献

1 Chen Ming-Syan, et al. Data Mining: An Overview from a Database Perspective. IEEE Trans. on Knowledge and Data Engineering. 1998,8(6):866～883

2 Pawlak Z. Rough sets. Intl. J. of Computer and Information Science, 1982,11(5):341～356

3 Ziarko W. Variable Precision Rough Set Model. J. of computer and system sciences. 1993,46:39～59

4 Pawlak Z. Rough set: Theoretical Aspects of Reasoning About Data. Kluwer Academic, Dordrecht, The Netherlands, 1991

5 Guan J W, Bell D A. Rough computational methods for information systems. Artificial Intelligence. 1998,105:77～103

6 Pawlak Z, et al. Rough Sets. Communication of The ACM. 1995, 38(11):89～91

7 Chan Chien-Chung. A rough set approach to attribute generalization in data mining. Journal of Information Sciences, 1998,107: 169～176

8 Bonikowski Z, et al. Extensions and intentions in the rough set theory. Journal of Information Sciences. 1998,107:149～167

9 苏健，高济．基于元信息的粗糙集规则增式生成方法．模式识别与人工智能，2001,14(4):428～433

（上接第 23 页）

方式。再有，每次最终的发送都要落实到发送单个 vbuf 的底部函数 viadev_post_send（除 RDMA 传输方式 RGET 和 RPUT 以外），调用宏定义 PACKET_SET_CREDITS 来设置流控数据，其中包括：把本地连接的域 local_credit（每次带过去远端的表示 credit 改变量，相当于 credit_update）赋值给发送包头的 vbuf_credit 成员，而后重置为 0，而且包头的 remote_credit 存贮本地 remote_credit 计数。最后，本地 remote_credit 自减 1。除此之外，流量控制和分析在处理接收过程时也有所涉及，获取发送过来的包以后，将包头里面携带过来的 vbuf_credit 叠加到接受端的 remote_credit 以更新。

为了避免由于一方不断大量发包，一方只接收但不反馈消息而造成的 credit 拥塞现象和程序死锁，在两种情况下会发送 noop 消息：

1. 所有完成队列里面的包都处理完毕，表示所有要到来或者要发出的消息已经完成。此情况下，本端发送 credit 的更新信息给远端，表示本方重新 post 进新的可用 vbufs。

2. 在我们正在处理接收描述子时。如果本地 credit 值足够大，但我们又不发送消息通知远端，连接的远端可能就会停止下来等待我们处理完所有接收任务从而发送一个 noop 使得它们能够继续工作。

5. MVICH 的现状和规划

比起一般的 TCP 通信，或是 LAM、MPICH 通信（建立在 TCP/TP 上）的性能，建立在 VIA 协议上的 MVICH 实现由于在传输数据上避免陷入系统核心，防止用户/核心间频繁切换，大大提高了网络速率[2]（例如在 Gigabit 以太网采用不同网卡，GNICII、Alteon 或 SysKonnect 进行比较），是将来高速局域网络应用程序的发展方向之一。用 MPI 与 VIA 的结合取代 MPI 和 TCP 或 UDP 的结合，将更能增强 MPI 执行性能，并且有进一步优化自身的开发潜力[1]。目前我们在 Th-net VIA 上调试通过的 MVICH1.0 版本经过改进，性能有了进一步提高，点到点的通信带宽达到 83MB/s（基于 33MHz/32bit 的 PCI 总线）。它和 MPICH 相互间配合，利用后者的 mpirun 脚本运行程序；在运行时动态调整参；可用 vipl 库点对点连接管理；采取更有效的流控算法，使得空消息的发送数目降至最低；和底层 VIA 协议相一致，使用动态（懒惰）注册内存机制（可降低系统负载）；vbuf 池的动态分配消除了资源不足问题[3]。

今后的工作旨在深入研究各方面因素。首先，拟将系统移植到 Linux2.4 内核上运行，实现 MVICH 的多线程版本，提高系统对于异步事件的敏感度和灵活性，其次，完善流控机制，支持不可靠的网络底层环境和异步通信，在提高稳定性的基础上提升传输速率，最后，进一步消除资源饥饿，从而优化 MVICH 性能。

总结 这篇文章里，结合我们清华大学高性能研究所自行开发的 VIA 网络接口卡和 TH-net VIA 底层环境，重点介绍了建立在其上，利用 VIA 传输协议的一种 MPICH 的设备层实现 MVICH。从其背景 MPI 标准入手，说明 MPICH 的核心机制 ADI 和 MVICH 在整个体系中的位置。然后深入到 MVICH 内部，较为详细地阐述其管理消息，实现消息传递的基础（vbuf 固定缓冲机制）。介绍四种传输协议 Eager，R3，RGET，RPUT，包括它们的模式和具体实现方法。接着描述 MVICH 里面的流控环节，依靠这个 MVICH 得以顺利发送大额数据而不造成拥塞。文章的最后概述了我们对提高现有 MVICH 性能所做工作和今后计划。

参考文献

1 Berkeley Lab VIA Software, the Regents of the University of California. M-VIA and MVICH, Virtual Interface Architecture Software for Low-Latency, High-Bandwidth, Inter-Process Communication, 2001

2 Ongg H, Farrelly P A. Performance Comparison of LAM/MPI, MPICH, and MVICH on a Linux Cluster connected by a Gigabit Ethernet Network. 2000

3 Lawrence Berkeley National Lab. M-VIA and MVICH Status and Future Plans, 2001

4 Compaq Computer Corp, Intel Corporation, Microsoft Corporation. Virtual Interface Architecture Specification Draft Revision 1.0, 1997

5 Speight E, Abdel-Shafi H, Bennett J K. Realizing the Performance Potential of the Virtual Interface Architecture, 1999

6 Brightwell R, Maccabe A B. Scalabilty Limitations of VIA-Based Technologies in Supporting MPI, 2000

7 Gropp W, Lusk E. A high performance, portable implementation of the mpi message passing interface standard, 1996

8 International J. for Supercomputing Applications. The Message Passing Interface(MPI)standard, 1995. & 都志辉，高性能计算之并行编程技术——MPI 并行程序设计

9 Gropp W, Lusk E. An Abstract Device Definition to Support the Implementation of a High-Level Point-to-Point Message-Passing Interface, 1995