面向自动文摘的多 Agent 系统的设计和仿真分析*)

Design and Simulation Analysis of Multiagent Systems for Automatic Abstracting

肖 明 胡舜耕 刘晓宇

(亚信科技(中国)有限公司)(北京邮电大学信息工程系 北京100876)

Abstract To build multiagent systems for automatic abstracting (MAS/ABS), two key problems must be resolved. Firstly, suitable number of abstracting agents for each domain must be given in some load. Secondly, what cooperation strategy should be used. In this paper, the model of multiagent system for automatic abstracting is given, three kinds of cooperation algorithms are set forth. We analyze the performance of the system based on the simulations, and get suitable number of abstracting agents for each domain in given load. Furthermore, we compare given cooperation algorithms.

Keywords Automatic abstracting, Multiagent system, Agent number, Cooperation algorithm, Simulation, Analysis

一、引言

建造面向自动文摘的多 Agent 系统是解决文摘系统适用领域的通用性和文摘质量的矛盾的一种可行的方案[1,2]。要设计和建造这样的多 Agent 系统，有两个关键问题：首先是确定在一定负载下，面向各个领域的合适的文摘 Agent 数目，其次就是选择什么样的协调算法。本文给出了在 Internet 环境下面向自动文摘的多 Agent 系统模型，提出了两种协调算法，在仿真的基础上分析了系统的性能，得到了在一定负载下面向各个领域的合适的文摘 Agent 数目，并对两个协调算法进行了比较研究。

二、系统模型

我们设计开发的面向自动文摘的多 Agent 系统(MAS/ABS)能向 Internet 用户提供自动文摘服务，如图1所示，系统包括一个动态变化的用户集，他们通过 Internet 向 MAS/ABS 请求自动文摘服务。从 MAS/ABS 的观点来看，用户的特征就是他所提交的文本的类型即文本所在的领域。MAS/ABS 中标识为 SA 的是协调者 Agent，它也是用户界面 Agent。标识为 IA 的是信息 Agent，它负责收集所有文摘 Agent 的相关历史信息和当前状态信息。标识为 $abs_{11},\cdots,abs_{1k_1},\cdots,abs_{n1},\cdots,abs_{nk_n}$ 的是文摘 Agent，它们在后台完成自动文摘服务。注意，所有的文摘 Agent 称为一个文摘 Agent 网络(记为 AAN)。$abs_{i1},\cdots,abs_{ik_i}$ 称为 AAN 中的第 i 类文摘 Agent，或第 i 类文摘 Agent 子网络。abs_{ij} 称为第 i 类文摘 Agent 子网络中第 j 个文摘 Agent，而且规定序号越小，文摘 Agent 做出的文摘质量越高。这里，$i\in\{1,2,\cdots,n\}$ 是文摘 Agent 网络所适用的领域标号，k_i 是面向领域 i 的文摘 Agent 数目。

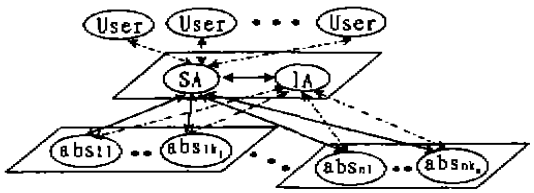


图1 系统模型

2.1 文摘 Agent 网络

文摘 Agent 网络由高度异质的、完全分布的文摘 Agent 组成。每个文摘 Agent 可以看作是 Internet 中的一个节点，它能为特定的领域提供高质量的文摘服务。因此，这些文摘 Agent 都是领域受限的。对任意 $i\in\{1,2,\cdots,n\}$，$abs_{i1},\cdots,abs_{ik_i}$ 可以由不同的人、基于不同的平台、使用不同的设计方法所开发的，可能是完全异质的。当然，它们中也可能有完全相同的，甚至于其中的一个是另一个的复制品。但是，一般来讲，这些文摘 Agent 应该是高度异质的，因此它们所作出的文摘质量有高低，服务时间有长短。

我们利用北京邮电大学智能研究中心研制的几个单机文摘系统做了统计试验，结果发现在同一硬软件

*)本项研究得到国家自然科学基金(项目编号69982001)资助。

平台上，在系统平稳工作的前提下，一个文本的长度和某一文摘 Agent 对该文本完成自动文摘所需时间或服务时间之间有确定的关系，可以用实函数来近似计算。假设 $abs_{11},\cdots,abs_{1k_1},\cdots,abs_{n1},\cdots,abs_{nk_n}$ 的服务时间分别由实函数 $f_{11}(x),\cdots,f_{1k_1}(x),\cdots,f_{n1}(x),\cdots,f_{nk_n}(x)$ 来计算。其中 x 是文本的长度。

2.2 用户

本文中，我们只关心在一定访问量或负载的情况下文摘 Agent 网络 AAN 的性能。AAN 中的文摘 Agent 对用户是透明的，即用户并不知道是哪一个文摘 Agent 在为它提供文摘服务。但是，为了提高系统的工作效率和稳定性，协调者 Agent SA 或协调策略需要来自用户的信息。SA 并不是去识别每一个用户，它只须知道用户所提交的文本类型，有时候，它还需要知道其文本的长度等其它信息。

我们用文本来模拟用户向 MAS/ABS 提交的请求，这些文本将根据其内容在 SA 中进行分类，以获取文本的领域信息。在本文中，文本属于标记为 $i\in\{1,2,\cdots,n\}$ 的 n 个不同领域。此外，我们将不区分用户、文本和数据包。

一个关键性问题是用户向 MAS/ABS 提交的文本是如何在不同领域之间分布的，即提交的文本属于某一个领域的概率有多大。我们采用纯 Zipf 函数来模拟文本在不同领域之间的分布。Zipf 形分布是这样定义的：对一个有 L 个元素的集合，选择第 i 个元素的概率 q_i 定义为 $q_i=c/i^{(1-x)}$。这里 $c=1/[\sum_{i=1}^{L}1/i^{(1-x)}]$ 是归一化因子[3,4]，$0\leqslant x\leqslant 1$ 为参数。所谓纯 Zipf 函数就是参数 $x=0$ 的情形。那么，为什么选择纯 Zipf 函数来模拟文本在不同领域之间的分布呢？纯 Zipf 函数通常被用来模拟人们所做的选择的分布，如模拟用户对 Web 页的请求次数[4,5]。而用户向 MAS/ABS 所提交的文本实际上反映了用户在不同领域之中的偏好或选择，所以用户提交的文本在不同领域之中的分布应该近似地符合 Zipf 分布，这也是符合人们的直觉的。

2.3 MAS/ABS 的工作原理

用户把文本通过适当方式（如 E-mail）提交给协调者 Agent SA，SA 根据预先确定的协调策略把文本提交给合适的文摘 Agent。该文摘 Agent 负责完成文摘任务以后，再把文摘传送给 SA。最后 SA 把文摘以适当方式提交给用户。若用户所提交的文本超出 MAS/ABS 的适用领域，系统会拒绝提供服务，并给出提示。

三、协调策略

当 SA 接收到一个用户提交的文本时，它如何决定把该文本提交给一个合适的文摘 Agent？SA 不仅利用用户或文本信息，而且也利用文摘 Agent 网络 AAN 的信息。完善的协调策略还将考虑文摘 Agent 的历史信息。用户或文本信息即文本所属的领域和文本长度等。AAN 信息就是 AAN 中各个文摘 Agent 的队列长度或服务时间等。文摘 Agent 的历史信息主要指在过去一个时间段内的利用率。下面我们将给出两种协调算法。

3.1 最小队列算法（MinQ）

第一种协调算法是根据文本所属领域和文摘 Agent 网络 AAN 中各个文摘 Agent 的队列长度来决定提供服务的文摘 Agent。协调者 Agent 首先确定用户提交的文本所属领域 TD，然后将文本提交给第 TD 类文摘 Agent 子网络中当前队列最短的文摘 Agent 中序号最小的 Agent。如果该类文摘 Agent 子网络中所有文摘 Agent 的队列均满，用户的文摘请求将被拒绝。可描述如下：令

$$sign(x)=\begin{cases}1, & x=0,\\ 0, & x\neq 0.\end{cases}$$

设 TD（Text Domain）表示文本所属领域，$N(abs_{ij})$ 表示 abs_{ij} 在第 i 类文摘 Agent 子网络中的序号，L_{ij} 和 Q_{ij} 分别表示 abs_{ij} 允许的最大队列长度和当前队列长度。为了简化算法，这里的 Q_{ij} 并不包括正在被服务的文本。其中 $i\in\{1,2,\cdots,n\}$，$j\in\{1,2,\cdots,k_i\}$。令 $MQS=\{abs_{ij}\mid(L_{ij}-Q_{ij})\cdot sign(TD-i)=\max_{p,l}[(L_{pl}-Q_{pl})\cdot sign(TD-p)\neq 0]\}$，$N(abs_{i_0j_0})=\min\{N(abs_{ij})\mid abs_{ij}\in MQS\}$。那么，文本将被提交给 $abs_{i_0j_0}$。注意，因子 $sign(TD-P)$ 确保了 $N(abs_{i_0j_0})$ 是合理定义的。另外，如果 $MQS=\phi$，则 $N(abs_{i_0j_0})=0$。这种情况表明第 TD 类文摘 Agent 子网络中的所有文摘 Agent 队列均已满，用户的文摘服务请求将被拒绝。

3.2 最小等待时间算法（MinWT）

一般而言，文摘 Agent 网络 AAN 中的文摘 Agent 是异质的，对相同文本的服务时间是不相同的。MinQ 算法没有考虑文摘 Agent 的服务时间。事实上，可能出现这样的情况，一个文摘 Agent 尽管其当前队列很长，但因为服务速度快，所以等待中的文本能很快被服务。相反，如果一个文摘 Agent 的当前队列很短，但服务很慢，那么文本等待服务时间就长。所以，在一个高度异质的多 Agent 系统中，我们还应该考虑服务时间问题。

设提交给 SA 一个文本时，文摘 Agent abs_{ij} 中队列长度为 Q_{ij}，队列中的文本长度依次为 $x_1,x_2,\cdots,x_{Q_{ij}}$。那么，被提交给 abs_{ij} 的下一个文本等待服务的时间为：

$$t_{ij}=f_{ij}(x_1)+f_{ij}(x_2)+\cdots+f_{ij}(x_{Q_{ij}})$$

这里 t_{ij} 没有包括正在被 abs_{ij} 服务的文本的剩余服务时

间，这纯粹是基于简化算法的考虑。其中 $i\in\{1,2,\cdots,n\}$，$j\in\{1,2,\cdots,k_i\}$。令

$MWT=\{abs_{i,j}\mid t_{i,j}\cdot sign(TD-i)=\min_{s,t}[t_{st}\cdot sign(TD-p)\neq 0]$且 $Q_{i,j}-L_{i,j}\neq 0\}$

$N(abs_{i_0j_0})=\min\{N(abs_{i,j})\mid abs_{i,j}\in MWT\}$

那么，SA 将把该文本提交给文摘 Agent $abs_{i_0j_0}$。这里，因子 $sign(TD-p)$ 确保 $N(abs_{i_0j_0})$ 是合理定义的。如果 MWT＝φ，则 $N(abs_{i_0j_0})=0$。这种情况表明第 TD 类文摘 Agent 子网络中的所有文摘 Agent 队列均已满，用户的文摘服务请求将被拒绝。

利用 MinWT 算法，协调者 Agent 首先确定用户提交的文本所属领域 TD，然后将文本提交给第 TD 类文摘 Agent 子网络中等待服务时间最短的文摘 Agent 中序号最小的 Agent。如果该类文摘 Agent 子网络中所有文摘 Agent 的队列均满，用户的文摘请求将被拒绝。

表1　AAN 中第一类文摘 Agent 的性能指标

领域1的 Agent 序号	协调算法	最大队列长度	平均队列长度	Agent 的利用率
Agent1	MinQ	2	.663269	.965435
	MinWT	2	.012142	.746078
Agent2	MinQ	2	.467560	.849677
	MinWT	2	.011079	.656887
Agent3	MinQ	2	256973	.593456
	MinWT	1	.011542	.537136
Agent4	MinQ	2	.101172	.291677
	MinWT	2	.013252	.409536
Agent5	MinQ	2	.028405	095682
	MinWT	2	.014271	.281551
Agent6	MinQ	2	.005697	.020470
	MinWT	2	.018604	.185420

表2　面向各个领域的文摘 Agent 数

领域编号	1	2	3	4	5
文摘 Agent 数	6	4	3	3	2
平均利用率(MinQ)	.4767	3554	.3184	.2355	.2835
平均利用率(MinWT)	.4694	.3505	.3128	.2319	.2800
最大队列长度	2	2	2	2	2
领域编号	6	7	8	9	10
文摘 Agent 数	2	2	2	2	2
平均利用率(MinQ)	.2366	.2000	.1785	.1607	.1416
平均利用率(MinWT)	.2335	.1982	.1766	.1575	.1397
最大队列长度	2	2	2	2	2

四、仿真参数和性能指标

我们在 Unix 支持下的 BONeS® 仿真环境下建立了离散事件的仿真模型。本节中，我们给出仿真模型中的参数值。在实际系统中，文摘 Agent 是异质的，解决问题的能力是各不相同的。但在我们的仿真模型中只考虑了一种特殊情况，即面向同一领域的文摘 Agent 能力是相同的，无差别的。具体表现在面向同一领域的文摘 Agent 所作出的文摘质量相同，对同一长度文本作出文摘的服务时间相同。

为了确定计算文摘 Agent 的服务时间的函数，我们利用北京邮电大学智能研究中心研制的一个单机自动文摘系统进行了实验。获取的每个数据可表示为(x，y)，其中 x 是文本长度(以字数记)，y 是该文摘系统其长度为 x 的文本作出摘要所需时间(单位为秒)。我们获取了九十组这样的数据，通过函数拟合，得到该文摘 Agent 的服务时间可用形如 $f(x)=cx^{\frac{1}{k}}+d$ 的实函数来近似计算。其中，c，d，k 是非负实数，且 k≠0，具体的实验结果是：c＝0.001，d＝1，k＝1.035。在仿真模型中，我们假设 AAN 中所有的文摘 Agent 的服务时间都可以用这一函数来近似计算。

在仿真模型中，文本长度服从正态分布，均值为 7000(字)，方差为 3000^2。用户向 MAS/ABS 提交文本的时间间隔服从均值为0.75的负指数分布。值得指出的是，均值为0.75秒的系统负载意味着每小时有4800人次向 MAS/ABS 提交文本并请求文摘服务，因为我们的多 Agent 系统被设计成向广大 Internet 用户提供自动文摘服务，为了保证服务效率，设计系统时考虑这样的负载是合适的。

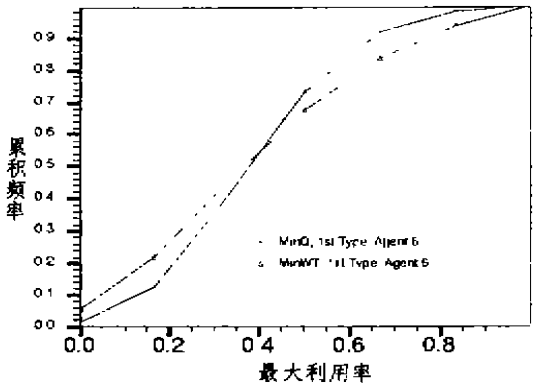
假定 MAS/ABS 的适用领域数 n＝10。我们要通过仿真确定面向每个特定领域的文摘 Agent 数目，因此把面向每一领域的文摘 Agent 数目分别设为1，2，…，10。根据这十种不同情况进行仿真，分析研究仿真结果，确定各类文摘 Agent 的合适数目。对于历史信息，我们考虑的是提交文本时刻之前240秒内各个文摘 Agent 的利用率。

我们选择的度量文摘 Agent 网络 AAN 性能的指标是：文摘 Agent 中最大队列长度；文摘 Agent 的平均队列长度；文摘 Agent 的利用率。最大队列长度表明文摘 Agent 必须预留的最大缓冲，平均队列长度反映文摘 Agent 之间任务分配的均衡性，文摘 Agent 的利用率表明其工作时间长短。除此之外，为了进一步考察整个文摘 Agent 网络 AAN 的性能，我们还采用了所谓的系统利用率。下面对此做一说明。为方便起见，将面向领域 $i\in\{1,2,\cdots,n\}$ 的所有文摘 Agent 记为 $Set_i=\{abs_{i1},\cdots,abs_{ik_i}\}$，称为 AAN 中第 i 类文摘 Agent 子网络。在仿真过程中，系统将接收到100,000个文本或数据包。系统以每0.03705秒的时间间隔对 Set_i 中的每一个文摘 Agent 进行抽样并记录在 Set_i 中有多少个文摘 Agent 在工作。根据仿真结果，我们可以得

到 Set_i 中有 j 个 Agent 在工作的频率，这就是所谓的系统利用率。进一步，还可以得到 Set_i 中最多有 j 个 Agent 在工作的累积频率，并且得到 Set_i 的最大利用率分布。这里，$j\in\{1,2,\cdots,k_i\}$。

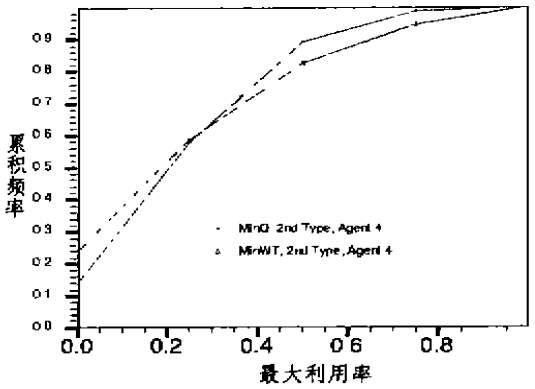
五、仿真结果和分析

通过仿真，我们得到了在前述两种协调算法下、Set_i 中文摘 Agent 数目从1到10的各种情况下的上述AAN 的性能指标、部分仿真结果见表1。同时还得到了各种情况下 Set_i 的最大利用率分布。图2表示 $|Set_1|=6$时，在两种不同协调算法下 Set_1 的最大利用率分布、图3表示 $|Set_2|=4$时，在两种不同协调算法下 Set_2 的最大利用率分布，这里 $|Set_1|$ 和 $|Set_2|$ 分别表示集合 Set_1 和 Set_2 中元素个数。限于篇幅、我们省略了所有其它的仿真结果和最大利用率分布。



（第一类，六个文摘 Agetn）

图2 最大利用率分布



（第二类，四个文摘 Agetn）

图3 最大利用率分布

为了确定面向各个领域的合适的 Agent 数目，我们给出如下的原则：

(1)在两种协调算法下，每个 Set, 的平均利用率都不超过0.5；

(2)每个文摘 Agent 中的最大队列长度为2。

依据上述原则，得到面向各个领域的合适的文摘 Agent 数目，见表2。原则(1)确保 AAN 的负载不会过大，平均说来，在任何时刻，每类文摘 Agent 中正在工作的文摘 Agent 数目不会超过一半。原则(2)确保 AAN 能有效应付突发性的大量文摘请求。这里，每个文摘 Agent 中允许的最大队列长度仅为2，这是为了适当控制 AAN 对文本的处理时间。如果对实时性要求很高，就应该增加各类文摘 Agent 的数目。另外，MAS/ABS 是一个开放系统、开放系统的性能是很不稳定的。我们所给出的每类文摘 Agent 的数目可以说是为确保 AAN 的性能所必需的最小配置。更多的 Agent 无疑是允许的，它将加快 AAN 对用户请求的响应速度，从而有效地改善 MAS/ABS 的性能。

下面比较一下两种不同协调算法。从表1可以看出，利用两种算法的最大队列长度基本相同，而平均队列长度和 Agent 的利用率相差较大。限于篇幅，表1中只给出了第一类文摘 Agent 的性能指标。从表2看到，不管利用哪种算法，AAN 中的各类文摘 Agent 的系统平均利用率都不超过50%，而且利用 MinWT 时，各类文摘 Agent 的系统平均利用率几乎一致。进一步分析表明，利用 MinQ 的 MAS/ABS 中面向同一领域的文摘 Agent 中的平均队列长度和 Agent 的利用率随着文摘 Agent 序号的增加而减小，且相差悬殊。利用 MinWT 的 MAS/ABS 中面向同一领域的文摘 Agent 中的平均队列长度相差不大，而 Agent 的利用率随着文摘 Agent 序号的增加而减小，且相差比较悬殊，只是这种不均衡性比利用 MinQ 算法时稍好。从最大利用率分布来看，利用算法 MinQ 时，各文摘 Agent 子网络中有一半左右文摘 Agent 在工作的概率较大，全部空闲和全都在工作的概率很小，利用算法 MinWT 时，基本情况和算法 MinQ 相似，只是全部空闲和全都在工作的概率较利用算法 MinQ 时稍大。在第二节中我们指出了每一类文摘 Agent 的序号都是按它们作出的文摘质量从高到低而递减的，因此，如果只考虑文摘质量而不关心任务分配的均衡性，利用算法 MinQ 较好、它所需系统信息也是最少的。如果既要考虑文摘质量又要任务分配较均衡，算法 MinWT 较好，但它所需系统信息也较多。

结论 多 Agent 系统协调方法分为显式协调和隐式协调两大类[6]。以为多 Agent 系统制定社会规则为代表的隐式协调方法还不够成熟。显式协调又可以分成三类：完全集中的协调，集中与分布结合的协调以及完全分布的协调。完全集中的协调不适合于动态、开

放的环境，很少用于多 Agent 系统中。完全分布的协调方法主要有两种类型：(1)不须交互，但需要其它 Agent 的许多信息和知识。(2)需要各 Agent 直接交互。显然，第一类协调法不适合动态、开放环境中的多 Agent 系统，我们重点分析第二类协调法。应用第二类协调法的多 Agent 系统结构实际上就是 Agent 网络结构。在这样的系统中，当 Agent 数目较大时，有明显的几个缺陷：一是通信代价太大，导致系统的低效率；二是这样的 Agent 网络的实现太复杂；三是每个 Agent 都具备协调管理能力，在系统开发上将产生大量重复劳动，不符合软件重用的思想；四是这样的体系结构不能满足系统动态、开放性的要求，因为 MAS 中的每个 Agent 都要拥有大量其它 Agent 的信息和知识，这在动态，开放的系统中是做不到的；五是这样的协调方法不符合人类社会组织的协调原理，在一个大型社会组织里一定有一个或多个协调管理中心，这种组织的工作才是高效的。因此，一般的多 Agent 系统都使用协调管理 Agent，这将大大提高系统性能和效率，减少系统实现的复杂度，有利于软件重用。综上所述，完全分布的协调方法不适合于动态、分布环境中的多 Agent 系统，特别是大型多 Agent 系统。最后，我们考察一下集中与分布相结合的协调方法。这类协调方法是指系统中的 Agent 组成层次结构，上层的协调管理 Agent 对下层的 Agent 有部分的控制能力，和完全集中的协调相比，既提高了系统整体性能和效率，更接近人类社会的特征，也体现了系统中 Agent 的自主性，适合应用于动态、开放的多 Agent 系统中[6~8]。正是基于上述考虑，我们的协调策略都是集中与分布相结合的。另外，我们根据 Internet 环境的特点，建立了用户模型、业务模型和协调模型，并在此基础上建立了系统的仿真模型，通过仿真分析解答本文开头提出的问题。用仿真的方法进行多 Agent 系统的性能分析尚不多见。

本文的工作为在 Internet 环境下设计和开发面向自动文摘的多 Agent 系统提供了依据。而且，只要对诸如 MAS/ABS 适用的领域数量、系统的访问量或负载等参数作适当的修改，我们可以得到更多具有指导意义的结果。正是在上述工作的基础上，利用北京邮电大学智能研究中心研制的几个基于单机的自动文摘系统，我们在 Internet 环境下建造了一个面向自动文摘的多 Agent 系统。目前，系统采用的是 MinQ 协调算法。

参 考 文 献

1 Hu Shungeng, Zhong Yixin, Wei Chaocheng. An automatic abstracting architecture based on multiagent technologies. In: Proc. of Intl Conf on MT & CLIP. Beijing, China, 1999
2 胡舜耕，钟义信，魏超成. 基于多 Agent 技术的自动文摘研究. 计算机工程与应用(已录用)
3 Zipf G K. Human Behaviour and the Principles of Least Effort. Cambridge, Mass.: Addison-Wesley, 1949
4 Colajanni M, et al. Analysis of task assignment policies in scalable distributed web-server systems. IEEE Trans. on Parallel and Distributed Systems, 1998, 9(6): 585~600
5 Cunha C, et al. Characteristics of WWW Client-Based Traces: [Technical Report BU-CS-96-010]. Computer Science Dept., Boston Univ., Apr. 1995
6 李建民，石纯一. DAI 中多 Agent 协调方法及其分类. 计算机科学，1998, 25(2)
7 Miao X, et al. A Normative-Descriptive Approach to Hierarchical Team Resource Allocation. IEEE Trans on Sys. Man. and Cyb., 1992, 32(3)
8 Genesereth M R, et al. Software Agents. Communications of the ACM, 1994, 37(7): 48~53

(上接第68页)

数其最小支持度为3/8，可以满足最小支持度，因此应该在大项集中加入这个集合(如图4)。图中 * 表示预测要发生的事件。

图4

小结 本文把时间序列预测用于关联规则的维护过程，分析了用时间序列来预测事件的发生，把事件与数据集对应起来。分析了关联规则的维护在基于动态变化的数据集时的不足，并加以改进，得到了新的对大项集的判别算法，用实例证明了对关联规则维护的改进过程，得到了更有意义的关联规则，这对关联规则的准确维护起到了重要作用。由于预测的准确性影响着规则的可靠性，这还需要以后不断地改进和探索。

参 考 文 献

1 Ester M. Algorithm For Characterization And Trend Detection. In: Spatial Database Process of 4th Intel Conf. On KDD, 1998. 1~4
2 Povinelli R J. Time Series Identify temporal patterns for characterization and prediction of time series events. 1999, 12: 15~40
3 陆玉昌. 数据挖掘与知识发现. 中国计算机用户，1999, 10: 29~32
4 钱治欣，陈奇. 关联规则采掘综述. 计算机应用研究，2000, 1: 1~5
5 冯玉才，冯剑林. 关联规则的增量式更新算法. 软件学报，1998, 9(4): 301~306
6 董力. 现代经济管理预测与决策. 地震出版社，1999. 66~99
7 时间序列用于经济预测的方法. Available at: http://mba.netbig.com/teach/course/973/2000628/ts/d064.htm