基于数据交叠分区的并行 DBSCAN 算法*

宋　明, 刘宗田
(上海大学 计算机工程与科学学院, 上海 200072)

摘　要: DBSCAN 是基于密度的聚类算法的一个典型代表, 它对空间数据库聚类有很好的性能。然而, 在对大规模数据库聚类时, DBSCAN 需要大量内存支持并伴随着 I/O 开销。随着高性能计算机的发展, 特别是集群式计算机的出现, 提供了一种解决 DBSCAN 算法缺陷的方法。测试表明, 它极大地降低了 DBSCAN 对时间和空间的需要。

关键词: 大规模数据库; 聚类; 数据交叠分区; DBSCAN 算法; 并行计算
中图法分类号: TP301.6　　**文献标识码**: A　　**文章编号**: 1001-3695(2004)07-0017-04

A Data-overlap-partitioning-based Parallel DBSCAN Algorithm

SONG Ming, LIU Zong-tian
(School of Computer Engineering & Sciences, Shanghai University, Shanghai 200072, China)

Abstract: As an outstanding representative of clustering algorithms, DBSCAN algorithm show good performance in spatial data clustering. However, for large spatial databases, original DBSCAN requires large volume of memory support and could incur substantial I/O costs. With the rapid development of high performance computer, especially appearance of the cluster computers give a means to overcome the defect of original DBSCAN algorithm. Experimental results show that the new algorithm is superior to the original DBSCAN in efficiency.

Key words: Massive Database; Clustering; Data-overlap-partition; DBSCAN Algorithm; Parallel Computing

0　引言

随着科学技术的发展, 大规模空间数据库中的知识发现变得越来越普遍和重要。但随着数据库规模的扩大, 单处理机无论是运算能力还是内存都不能满足数据挖掘的需要, 于是人们渐渐把目光投向并行计算和多处理机进行数据挖掘。

在数据挖掘中聚类是最广泛的应用之一, 聚类技术在统计数据挖掘、模式识别、图像处理等领域有广泛的应用。尽管人们在过去几十年研究中已经提出了许多聚类算法, 但是依然没有一个有效的对大规模数据库和高维数据聚类的算法。

本文提出了一个并行数据挖掘算法——基于数据交叠分区的并行 DBSCAN 算法。它在对大规模数据库和高维数据聚类时有很好的性能表现。它的实现环境是集群式高性能计算机(Cluster Computers), 是基于并行计算的消息传递模型来实现的。算法实现环境也可以是互连的几台普通计算机上, 性能也能较串行算法有较大提高。

1　关于DBSCAN 算法

Ester Martin 等人[2] 提出的 DBSCAN 算法是一个基于高密度连接区域的密度聚类方法, 它能够发现任意形状簇, 并能有效地处理噪音点。下面给出关于 DBSCAN 算法的一些主要定义(对于给定的对象集 D 和参数 Eps, minpts):

(1) 核心对象。如果对象 p 以 Eps 为半径的邻域内的对象数大于或等于 minpts, 则称 p 为核心对象。

(2) 直接密度可达。在对象集 D 中, 对象 p 在对象 q 的以 Eps 为半径的邻域内, 而 q 是一个核心对象, 则称对象 p 从对象 q 关于 Eps 和 minpts 直接密度可达。

(3) 边界对象。如果对象 p 不是核心对象, 但是从其他核心对象直接密度可达, 则称 p 为边界对象。

(4) 密度可达。如果存在对象链 $p_1, p_2, \cdots, p_n$, 其中 $p_1 = q$, $p_n = p$, 对于 $p_i \in D$, $(1 \leq i \leq n)$, p_{i+1} 是从 p_i 关于 Eps 和 minpts 直接密度可达, 则称对象 p 是从对象 q 关于 Eps 和 minpts 密度可达的。

(5) 密度相连。如果对象集 D 中存在一个对象 o, 使对象 p 和 q 是从 o 关于 Eps 和 minpts 密度可达, 那么对象 p 和 q 是关于 Eps 和 minpts 密度相连的。

(6) 簇(或类)。一个基于密度可达性的最大密度相连对象的集合。

(7) 噪音点。对象集合 D 中不属于任何簇的对象就是噪音点。

DBSCAN 算法大体步骤为: 从对象集 D 的任何一个对象 p 开始, 检索整个数据对象集 D, 找出所有关于给定的 Eps, minpts 从对象 p 密度可达的所有对象, 如果 p 是核心对象, 则创建一个新簇, 给簇的所有成员加上簇标签; 如果 p 不是核心对象, 暂时把它认为是噪音点, 处理数据库的下一个对象, 直到所有对象处理完毕。

由于没有作任何预处理而直接对数据库进行操作, 在 DB-

收稿日期: 2003-07-13; 修返日期: 2003-09-18
基金项目: 国家自然科学基金资助项目(60275022); 上海市科委重点基础研究项目(01JC14022)

SCAN 算法对大规模数据库聚类时,需要大量的内存和 I/O 开销。

2 并行DBSCAN 算法与数据分区

对于 DBSCAN 的缺点,周水庚等人[5]提出了一种基于数据分区的 DBSCAN 算法。其主要思想是根据数据库在某一维或多维的分布特性,将整个数据库划分为若干个交集为空的局部区域;然后依次对每个局部数据库计算局部 K-dist 图,并得出各自的 Eps 值,用 DBSCAN 依次对各数据分区进行局部聚类;最后是局部聚类合并。DBSCAN 算法的优点是:解决了数据规模过大、内存需求紧张局面,减少了算法耗时量,并能在数据密度分布不均匀时提高聚类质量。只是该 DBSCAN 算法实现在单机上,依然是串行对各数据分区聚类,提高算法运算速度能力有限。但它的数据分区思想可为并行算法所借鉴。

因为 DBSCAN 算法是核心对象反复对它的邻域查询来实现的,也就是说一个对象与其邻域外的对象在聚类时并没有直接联系。所以数据分区有利于把一个大任务划分成多个小任务,适合并行计算的特点。数据分区是把整个数据库划分成一些规则分区,一般是矩形分区。

基于数据分区并行 DBSCAN 实现步骤很简单:在数据分区后,把每个数据分区发送到一个处理机上,并行进行局部聚类,最后合并局部聚类。但是根据上述的分区办法,把整个数据库分成若干交集为空的局部分区,就割断了一些对象间的本来存在的联系,造成边界分区对象的邻域分别处于两个局部分区,使得一些本来是核心对象变成噪音点。一些全局类被拆分为几个类,这样的局部聚类合并算法必须对边界区域对象重新查询它的邻域,导致合并算法非常复杂、耗时。

不难发现,受影响的对象仅仅是局部分区边界上对象集。我们就试图在局部聚类时把相邻局部分区边界部分包括进来,以保证所有对象的邻域在局部聚类时完整地分布在一个局部分区内。这样在局部聚类合并时只需搜索一遍边界区域对象集,而且不需要把对象本身发送到相邻节点上,只需发送簇标签、是否是核心对象等聚类信息,这就大大减少了算法的通信耗时。

鉴于上述考虑,本文提出基于数据交叠分区(Data-overlap-Partitioning)的并行 DBSCAN 算法,称之为 OPDBSCAN。它有两个特点:①交叠分区;②并行计算。算法流程图如图 1 所示。

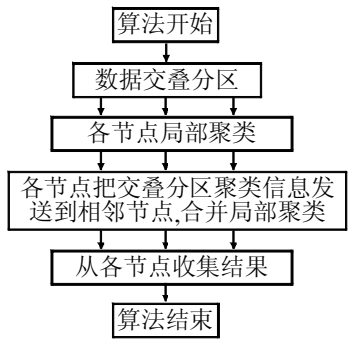

图 1 算法流程图

3 基于数据交叠分区的DBSCAN(OPDBSCAN)

大规模的数据库聚类时,数据分区是一个有效的方法。本文中因为是并行地实现 DBSCAN 算法,所以数据分区不仅仅缓解内存紧张的局面,还起到把整个任务划分成多个分任务的作用。为了使各分任务间耦合性少,且每个任务运算量相当,把整个数据库划分成规则的、数据量大致相同的几个数据分区。

3.1 交叠分区

对于一个给定数据库 DB 和参数 Eps(如果用户没给出,可用 K-dist 图得到)、minpts,数据库中任意元素 X 有 m 维(X_1, X_2, …, X_m),DB 中共有 N 个元素,把它分成 n 个数据量大致相同局部分区,步骤如下:

(1) 取任意整数 i($1 \le i \le m$),把数据库 DB 投影到第 i 维上,即$\forall X \in DB$, $X->X_i$,则数据库 DB 映射为一个一维数据集 DB_1,即 $DB_1 = \{X_i \mid X(X_1, X_2, \cdots, X_m) \in DB\}$。

(2) 令 A, B 分别为数据集 DB_1 的下限和上限,则数据集分布在区间[A, B]内。

(3) 找到数列 $a_0, a_1, \cdots, a_n$,满足:

①$A = a_0 < a_1 < a_2 < \cdots < a_n = B$;

②区间$[a_i, a_{i+1}]$内分布的数据个数为 $\lfloor N/n \rfloor$,$0 \le i \le n-1$。

(4) 如果分区结果满足条件 1,则结束分区。

(5) 如果分区结果不满足条件 1,则转到步骤(1)重新分区。

这样,数列 $a_1, \cdots, a_{n-1}$ 把区域[A, B]划分成 n 个区间$[a_i, a_{i+1}]$,i = 0, 1, 2, …, n;数据集 DB_1 内数据近乎均量地分布在每个区间,也就是说把 DB 内对象集近乎均等地分成 n 个分区。但是,这并不是最后的分区结果,为了保证每个对象在聚类过程中每一个对象的邻域至少被完整包含一次,这里提出交叠区域定义。

定义 1:区间$[a_i - Eps, a_i + Eps]\ \bigcup_{n-1}^{i=1}$称为交叠区域,区间$[a_i - Eps, a_i + Eps]$称为第 i 个交叠区域。

定义 2:交叠区域中包含数据的个数与整个数据库元素个数比值称为交叠比。

条件 1:事先给定一个阈值,交叠比必须小于该值。

条件 1 是控制交叠区域的数据量,如果数据量过大就会导致并行算法性能低下。把上面的分区加上交叠区域,得到最后的 n 个交叠分区为$[a_0, a_1 + Eps]$, $[a_i - Eps, a_{i+1} + Eps]$ (i = 1, 2, …, n - 2), $[a_{n-1} - Eps, a_n]$。

为了便于描述,用对象集列 $A_1, A_2, \cdots, A_n$ 分别表示上面 n 个交叠分区内对象的集合。

3.2 算法的实现与讨论

3.2.1 局部聚类及其聚类合并

算法实现环境是集群式高性能计算机,根据上面分区,用 n 个处理机节点来实现算法。算法的步骤为:

(1) 把对象集 A_i 发送到第 i 个处理机内存中,i = 1, 2, …, n;

(2) 每一个处理机节点用 DBSCAN 算法对本地对象集聚类;

(3) 聚类合并,对于处理机 i($1 \le i < n-1$):把第 i 个交叠区内对象的聚类信息发送到处理机 i + 1,与处理机 i + 1 相对应交叠区内对象集聚类信息比较,按合并规则合并局部聚类;

(4) 从各点收集结果,算法结束。

以上不难发现,处在交叠区内任意对象都聚类了两次,具

体合并规则就是根据其两次聚类信息比较获得。对于任意一个属于交叠区的对象 p, 在其两次聚类中, 有以下几种情况: ①分别属于类 A 和类 B, 如对象 p 至少有一次为核心对象, 类 A 和类 B 合并能为一个类, 如果都是边界点, 则对象 p 可以属于类 A 或类 B, 但不能确定类 A 和类 B 能合并; ②有一次属于类 A, 而另外一次是噪音点, 则对象 p 属于类 A; ③两次都为噪音点, 则对象 p 在全局聚类中就是噪音点。

从这些规则看来, 局部聚类合并时, 不需要把所有交叠分区内对象聚类信息发送给转载相邻分区的处理器, 而只要把交叠区域内核心对象发送出去, 就将减少算法通信量。

3.2.2　算法有效性讨论

这些规则是否与 DBSCAN 算法等价呢? 规则②, ③比较明显, 下面对于规则①有效性进行分析。规则①有三种情况:

(1) 对象 p 的两次聚类都是核心点, 根据核心点定义, 有 $\forall m \in A$, $m \neq p$, m 是从对象 p 密度可达的。同样有 $\forall n \in B$, $n \neq p$, n 是从对象 p 密度可达的。根据密度相连定义, $\forall p$, 使得类 A 与类 B 中任何非 p 对象 m, n 都是从 p 密度可达的, 那么 m 和 n 是密度相连的, 即类 A 与类 B 中任何非 p 对象之间都是密度相连的, 类 A 和类 B 属于同一个类。

(2) 对象 p 的两次聚类一次是核心点, 一次是边界点。不妨假定 p 是类 A 的核心点, 是类 B 的边界点, 因为对象 p 在类 A 内是核心对象, 有 $\forall m \in A$, $m \neq p$, m 是从对象 p 密度可达; 在类 B 中, 对象 p 是边界点, 则 $\exists n \in B$, $n \neq p$ 使对象 p 从对象 n 密度可达。根据密度可达定义, 得出对象 m 从对象 n 密度可达。又有 $\forall q \in B$, $q \neq n$, q 从对象 n 密度可达。根据密度相连定义, $\exists n$ 使对象 m、对象 q 都从对象 n 密度可达, 则对象 m 与对象 q 密度相连。又因为 m, q 在类 A、类 B 中任意性, 所以类 A 与类 B 是同一个类。

(3) 对象 p 的两次聚类都是边界点。假设在 $(A \cap B) = p$ 条件下, 类 A 和类 B 能合并为一个类, 有 $\forall m \in A$, $m \neq p$, 对象 m 与对象 p 密度相连, 同样 $\forall n \in B$, $n \neq p$, 对象 n 与对象 m 密度相连, 则有一个核心对象 q, $q \in (A \cup B)$, 使对象 m, n 都从对象 q 密度可达。又因为 $(A \cap B) = p$, 而 p 不是核心对象, 显然 $q \neq p$, 不妨假定 $q \in A$, 因为 n 从 q 密度可达, 则存在对象链 P_1, P_2, …, P_n, $P_1 = q$, $P_n = n$, P_{i+1} 从 P_i 直接密度可达。因 $q \in A$, $n \in B$, $A \cap B = p$, 则 p 一定在对象链中, 但这与对象 p 不是核心对象矛盾, 故在 $(A \cap B) = p$ 条件下, 不能确定类 A 和类 B 能合并。

可以看出, 局部聚类的合并规则是有效的, 而且操作也很简单。

3.2.3　算法的复杂性讨论

DBSCAN 算法时间复杂度为 $O(n^2)$, 其中 n 为数据库对象数目。当 n 很大时, 可以假定对象规模为 n 的数据库执行 DBSCAN 算法耗时 T_{DB} 为 $n^2 t$。OPDBSCAN 算法耗时主要在局部聚类及聚类合并时对交叠区点集聚类信息传输上, 而数据传输是并行执行的。

设 k 为数据分区数, ϕ 为交叠比, 因为数据库是近乎均匀划分, 每个节点上对象数目为 $[n(1+\phi)]/k$, 又设传输一个整数(对象聚类结果用整数表示)的时间为 T, 则数据传输所花时间为 $n\phi T/k$, 不记聚类合并耗时, OPDBSCAN 算法耗时 T_{OP} 为 $[n^2 t]/\{[n(1+\phi)/k]^2 t + n\phi T/k\}$, OPDBSCAN 的加速比 α 为

$$\alpha = \frac{T_{DB}}{kT_{OP}} = \frac{n^2 k}{n^2 (1+\Phi)^2 + n\Phi k \frac{T}{t}} \quad (1)$$

如果采用空间索引, DBSCAN 算法时间复杂度为 $O(n\log n)$, OPDBSCAN 加速比 α 为

$$\alpha = \frac{n\log n}{nk(1+\Phi)[\log n + \log(1+\Phi) - \log k] + n\Phi \frac{T}{t}} \quad (2)$$

对于同一个系统, T/t 值是确定的, 从式(1)可以看出控制交叠比并选择一个合适的数据分区数 k, 可以得到一个好的加速比。

4　性能评估

OPDBSCAN 算法在上海大学“自强 2000”上实现, 该系统采用 SMP(多对称处理器), 节点通过高速网络(100Mbps)互连方式。目前可用节点数为 100 个, 内存总量为 20GB, 硬盘总量为 20TB, 每个节点内存为 512MB, 硬盘 15 ~30GB, CPU 为 PⅢ 500MHz, 操作系统为 Linux(Readhat)。在 Martin Easter 等人开发的 DBSCAN 软件包提供的 DBSCAN 软件包基础上, 用 MPI 和 C++ 实现了 OPDBSCAN。测试数据由一个公开的聚类测试数据生成算法“Clusgen.c”产生, 该算法来自于 Dave Dubin (http://alexia.lis.uiuc.edu/~dubin)[9]。OPDBSCAN 算法把数据分区作为预处理, 不记入算法总的运行时间。

本文的测试数据都是二维, 每一维定义域都是 [0, 100000]。表 1 给出 OPDBSCAN 与 DBSCAN 对一组数据集的聚类结果。测试结果表明, OPDBSCAN 算法的聚类质量与 DBSCAN 算法是等价的。

表 1　OPDBSCAN 算法与 DBSCAN 算法聚类质量比较

数据规模	参数 (Eps, minpts)	DBSCAN		OPDBSCAN	
		类数	噪音点	类数	噪音点
5 000	19, 3000	3	680	3	680
10 000	13, 2150	3	1310	3	1310
20 000	19, 2000	4	2607	4	2588
50 000	19, 1225	3	6817	3	6750
100 000	19, 1000	5	13430	5	13339

图 2 给出 OPDBSCAN 算法对大小为 100 000 的数据集聚类时通信时间与总的运算时间比较。因为 OPDBSCAN 算法中每一个节点只与其相邻节点通信, 所以算法的通信耗时不会随计算的节点增大而无限增大, 而是节点数达到一定个数之后通信时间趋于稳定。

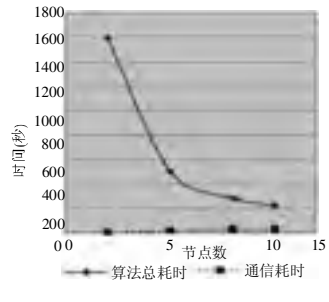

图 2　算法运行时间与通信时间比较

图 3 是 OPDBSCAN 算法用不同数据集(数据规模分别为 10 000, 40 000, 160 000, 640 000)相对加速比曲线图(相对加速比就是串行算法运行时间与并行算法运行时间的比值)。如图 2 所示, 在数据库规模加大情况下, OPDBSCAN 算法的时间性能优势越明显, 加速比趋近于 1。但并不是节点用得越多越好, 应根据数据库规模来确定最佳节点数。OPDBSCAN 算法

耗时与节点间通信速度有直接关系,提高通信速度将提高它的时间性能。

5 算法优缺点和日后工作

此算法有以下优点:①缓解了串行 DBSCAN 算法因数据规模过大内存需求紧张局面;②局部聚类合并算法简单,通信耗时少,通过使用多处理器,极大地提高了算法的运算速度;③该算法在聚类方式上与 DBSCAN 算法完全等价,为以后高质量聚类算法提供了基础。

为了提高局部聚类的速度,局部聚类可以采用空间索引的方法[1]或快速 DBSCAN 算法[7]。因为在 OPDBSCAN 聚类方式上与 DBSCAN 算法完全等价,而 DBSCAN 算法因使用全局的 Eps,只能发现密度相仿的簇,对参数非常敏感。本课题最终目的是建立一个数据挖掘系统,图 4 就是此算法在该系统上的运行流程图。系统对每次聚类结果的每一个簇进行评估,聚类结果满足用户需求则输出,否则重新执行基于交叠数据分区 DBSCAN 算法,直到满足条件为止。聚类结果评估和选择合适节点数是日后工作重点。

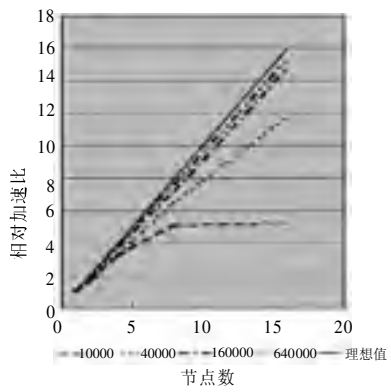

图 3 OPDBSCAN 相对加速比曲线图

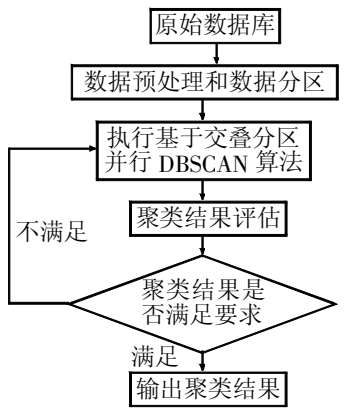

图 4 运行流程图

参考文献:

[1] eachmann N, et al. The R* - tree: An Efficient and Robust Access Method for Points and Rectanggles[C]. Proc. of ACM SIGMOD Int'l Conf. on Management of Data, Atlantic: ACM Press, 1998. 73-84.

[2] ster M, et al. A Densith-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise[C]. Proc. of 2nd Int'l Conf. on Knowledge Discovering in Databases and Data Mining (KDD-96), Portland: AAAI Press, 1996.

[3] uha S, Rastogi R, Shimk. CURE: An Efficient Clustering Algorithm for Large Databases[C]. Proc. of the ACM SIGMOD Int'l Conf. on Morgan Kaufmann, 1997. 186-195.

[4] aul Stolorz, et al. Scalable High Performance Computing for Knowledge Discovery and Data Mining[M]. Kluwer Academic Publishers, 1997.

[5] 周水耿,周傲英,等. 基于分区的 DBSCAN 算法[J]. 计算机研究与发展, 2000, (10): 1153-1159.

[6] aul Stolorz, Ron Musick. Scalable High Performance Computing for Knowledge Discovery and Data Mining[M]. Kluwer Academic Publishers, 1997.

[7] HOU Shuigeng, JIN Wen. FDBSCAN: A Fast DBSCAN Algorithm [J]. Journal of Software, 2000, 11(6): 735-744.

[8] Zhang, R Ramakrishnan. Birch: An Efficent Data Clustering Method for Very Large Databases[C]. Proceedings of the ACM SIGMOD Conference on Management of Data, Montreal, Canada, 1996.

[9] Milligan. An Algorithm for Creating Artificial Test Clusters[J]. Psychometrika, 1985, 50(1): 123-127.

[10] aul Stolorz, Ron Musick. Scalable High Performance Computing for Knowledge Discovery and Data Mining[M]. Kluwer Academic Publishers, 1997.

作者简介:

宋明(1978-),男,硕士研究生,研究方向为基于高性能计算机数据挖掘;刘宗田(1946-),男,教授,博士生导师,研究方向为人工智能和软件工程。

(上接第 16 页)智能体系统基础上,结合移动智能体技术,将使系统更高效,更富有柔性,更加实用化。

参考文献:

[1] 杨肖鸳. 敏捷制造[M]. 昆明: 云南科技出版社, 1997. 1-3.

[2] 杨肖鸳,等. 面向敏捷制造的虚拟企业管理信息系统框架研究[R]. 2001. 40-49.

[3] 赵春明. 虚拟企业[M]. 杭州: 浙江人民出版社, 1999. 1-30.

[4] 毕波. 试论虚拟企业[J]. 中国工业经济, 2001, (5): 75-77.

[5] 赵龙文,侯义斌. 多 Agent 系统及其组织结构[J]. 计算机应用研究, 2000, 17(7): 12-14, 25.

[6] 焦文品,等. 构造 MAS 的动态体系结构的模型[J]. 计算机学报, 2000, (7): 732-737.

[7] 瞿有甜. 基于 Agent 的虚拟组织的信息基础结构[J]. 计算机科学, 2000, (8): 29-32.

[8] 但斌,等. 虚拟供应链体系结构和运作模式研究[J]. 工业工程与管理, 2000, (5): 46-48.

[9] 杨晓春,等. 面向虚拟企业信息集成的扩展联邦模式[J]. 计算机集成制造系统—CIMS, 2001, (3): 19-25.

[10] 仲智刚,等. 敏捷制造集成策略研究[J]. 中国机械工程, 2000, (7): 767-770.

[11] 林东豪,等. 支持敏捷制造的企业新型信息系统体系结构研究[J]. 浙江大学学报, 2001, (1): 77-82.

[12] 庄晖,等. 支持供应链管理的多主体系统[J]. 工业工程与管理, 2000, (5): 49-51.

[13] 高济,等. 基于 Agent 技术的虚拟组织集成框架[J]. 计算机研究与发展, 1999, (12): 1417-1423.

[14] 史殿习,王怀民,等. 基于 Agent 的分布式计算环境[J]. 计算机学报, 1996, (3): 197-201.

[15] 周水银,等. 基于多 Agent 的供应链研究[J]. 工业工程与管理, 2001, (6): 28-31.

[16] 王丹,等. 基于 Agent 的面向虚拟企业信息系统的研究与实现[J]. 小型微型计算机系统, 2001, 22(2): 184-186.

作者简介:

杨肖鸳(1944-),女,云南人,教授,研究方向为虚拟企业与敏捷制造、管理信息系统;朱萍(1971-),女,山西人,讲师,硕士,研究方向为敏捷虚拟企业、电子商务;曹好顺(1976-),女,广东人,在读硕士,研究方向为虚拟企业与敏捷制造、管理信息系统;毕贵红(1968-),男,云南人,副教授,硕士,研究方向为敏捷虚拟企业、智能体技术。