基于剪枝的海量数据离群点挖掘

杨茂林 卢炎生
(华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 基于距离的离群点挖掘通常需要$O(N^2)$的时间进行大量的距离计算与比较,这限制了其在海量数据上的应用。针对此问题,提出了一个带剪枝功能的离群点挖掘算法。算法分为两步:在对数据集进行一遍扫描后,剪枝掉大量的非离群点;然后对余下的可疑数据实施一种改进的嵌套循环算法,以每个数据点与其k个最近邻点的平均距离作为离群度,确定前n个离群点。在真实数据和合成数据集上的实验结果均表明,该算法在获得高命中率的同时仍保持低误警率。与相关算法相比,其具有较低的时间复杂性。

关键词 离群点,数据挖掘,基于距离
中图法分类号 TP311 **文献标识码** A

Pruning-based Outlier Mining from Large Dataset

YANG Mao-lin LU Yan-sheng
(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract Distance-based outlier detection approach typically requires $O(N^2)$ time of distance computation and comparison. This quadratic scaling restricts the ability to apply this approach to large datasets. To overcome this limitation, a novel distance-based outlier mining approach with pruning rules was proposed. The approach consists of two phases. During the first phase, the original input data are scanned and the majority of non-outliers are pruned. During second phase, an improved nested loops approach is applied to compute the average K-nearest distance which measures the degree of being an outlier and finally reports the top-n outliers. Experiments on both synthetic data and real-life data show that the proposed approach achieves a high hit rate with a low false alarm rate. Compared with related approaches, the proposed approach has a lower time complexity.

Keywords Outlier, Data mining, Distance-based

1 引言

离群点检测是从大量的数据中发现少量偏离大多数的异常个体的过程。作为数据挖掘的一个分支,离群点检测在诸多领域有着广泛的应用,例如金融领域的欺诈、网络安全中的恶意攻击、视频监控、天气预报、医学诊断等。近年来,离群点挖掘取得了不少重要成果,研究人员提出了大量的离群点检测算法。

基于统计的方法[1]是出现得最早的离群点检测方法,它假设数据集服从某种分布或概率模型,通过进行不一致检验把那些严重偏离分布曲线的数据标记为离群点。这种方法主要适用于单变量,且分布模型与参数已知的数据集。但现实数据的分布模型大多是未知的,或根本不符合任何标准的分布,故其应用有局限性。

为克服基于统计离群点检测方法的缺点,Knorr 等人提出了基于距离的离群点检测方法[2,3]。该方法将数据点与其近邻点之间的距离作为离群与否的度量标准。离群点被定义为与大多数点的距离都大于某个阈值的点,记为 $DB(p,D)$离群点。数据集 S 中的某个对象 o 被称为 $DB(p,D)$离群点当且仅当 S 中至少有 p 部分的对象与 o 的距离大于 D。

基于距离的离群点检测方法的优点是它不必知道数据集的分布模型和分布参数,并且适用于任何维度的特征空间,只要该特征空间上存在某种距离度量手段。其缺点是需要进行大量的距离计算与比较,其时间复杂性一般为 $O(N^2)$。这限制了其在海量数据挖掘中的应用。

本文提出了一种带剪枝规则的基于距离的离群点检测方法,它在对数据集进行一次扫描后,即剪裁掉大量的正常数据,只需考察余下的少数数据的离群度,确定最靠前的 n 个(TOP n)离群点。由于数据集中的正常数据总是多数,因此大部分数据在第一步后即被剪枝,这大大减小了数据的规模,使得该算法能适用于高维海量数据集。

2 相关工作

近年来,研究人员提出了多个基于距离的离群点的定义及其检测算法。除 $DB(p,D)$离群点以外,较为流行的还有以下3种:

到稿日期:2011-12-12 返修日期:2012-03-06
杨茂林(1968-),男,博士生,讲师,主要研究方向为数据挖掘、软件工程,E-mail:kylin@hust.edu.cn;卢炎生(1949-),男,教授,博士生导师,主要研究方向为数据库系统、软件工程。

• 离群点是在距其距离为 r 的邻域空间内，数据点的个数少于某个阈值 k 的点[3]。

• 离群点是离其第 k 个最近邻点的距离最大的前 n 个点[4]。

• 离群点是离前 k 个最近邻点的平均距离最大的前 n 个点[5]。

上述第一个定义显然是 $DB(p,D)$离群点的另一种表述。这 3 个定义密切相关，但又略有区别。与第一个不同的是，后两者不再视离群为二值属性，而是赋予每个数据点一个离群的程度值，离群程度最强的 n 个点被视为离群点。两者的区别仅在于离群程度的计算方法不同。第二个定义只抽样考察第 k 近的点，但却忽略比它更近点的信息。第三个定义考察了所有 k 个最近邻点的信息，但需更多的计算。三者的共同点是都考察邻近点的距离，只要有足够多的邻点分布在足够近的空间内，该点就被视为正常点。否则，考虑其为离群点。

在基于距离的离群点检测方法的基础上，Brenunig[6] 等人提出了基于密度的离群点检测方法，它不仅考察数据点与其邻近点的距离，还考察给定范围内数据点的个数，两者结合得到“密度”的概念。以此为基础，计算每个数据点的局部离群因子 LOF(Local Outlier Factor)，将其作为离群程度的度量依据。LOF 值越大，离群程度越高。基于密度的离群检测方法能够检测出基于距离的方法所不能识别的一类异常数据——局部异常。但本质上，基于密度的方法是基于距离方法的扩展，其距离的计算量比基于距离的方法更甚。

所有以距离计算为基础的离群点检测方法都有一个共同的缺点，即需要大量的距离计算与比较，其时间复杂性一般为 $O(N^2)$，因其需要计算每对数据点间的距离，以确定最邻近的 k 个数据点，如循环嵌套(NL)算法。为减少距离的计算，研究人员提出了许多新的方法，如利用空间索引结构 KD-树[7]、R-树[8] 和 X-树[9] 等来查找数据点的 k 个近邻[3,4,10]。在低维空间，这些方法可以获得接近 $O(N\log N)$的时间复杂度，但当维度增加到一定程度时，方法失效。Breunig 等人采用一种 X-树查找 K 近邻，结果发现这种索引结构只在维度低于 5 的情形下才有较好性能[6]。

对数据进行空间划分可以更快确定对象的近邻点，从而成为减少距离计算的另一种手段，如基于网格(Cell-Based)的方法[10]。该方法的时间复杂性与数据集的规模 N 呈线性关系，却与维度呈指数关系。因此，其也不适用于高维数据集。当数据集的维度大到一定程度后，嵌套循环算法反而优于基于索引和基于网格的检测算法[10]。

现实数据往往具有高维和海量的特点，开发适合这类数据的离群点挖掘算法成为研究的热点。值得注意的是，一些研究人员转而寻求不以距离作为数据点之间关系度量标准的检测方法，也取得一定的成果[11]。但本文仍侧重于研究基于距离的离群点检测方法。

Bay 等人提出了一种改进的基于距离的离群点检测算法 ORCA[12]，它在 NL 算法的基础上增加了一条剪枝规则，且在考察离群度之前，先对数据集的数据排列顺序进行了随机打散处理。该算法可以取得接近线性的时间复杂度，但在最差情况下，仍需要 $O(N^2)$时间[13]。

本文在 Bay 等人的研究基础上，提出一个新的离群点检测算法。该算法在考察数据点的离群度之前，先对数据集进行一次剪枝，目的是尽早、尽可能多地剪掉确信为“非离群”的数据点。幸运的是，数据集中的正常数据总是多数，经一次剪枝后，数据集的规模被大大降低。与 ORCA 算法相比，该算法的时间复杂性显著降低，并且可以有效地挖掘高维海量数据集中的离群点。

3 相关定义

在以下的讨论中，设数据对象集为 $S=\{s_1,s_2,\cdots,s_i,\cdots,s_N\}$，$dist(p,q)$是度量数据对象 p 和 q 距离的函数。S 的属性集 $A=\{A_1,A_2,\cdots,A_d\}$。对象 p 的第 i 个属性值记为 $p.A_i$。

定义 1(r-邻域) 对任意实数 $r\geqslant 0$，对象 $p\in S$ 的 r-邻域 $N(p,r)$定义为：

$$N(p,r)=\{q\in S\mid dist(q,p)\leqslant r\}$$

定义 2(k-距离) 对任意正整数 k，数据对象 p 的 k-距离 $D^k(p)$定义为对象 p 和某个对象 $o\in S$ 之间的距离 $dist(p,o)$，使得：

(1) $|N(p,dist(p,o))|\geqslant k$；

(2) 对任意的实数 $r<dist(p,o)$，都有 $|N(p,r)|\leqslant k-1$。

定义 3(k-近邻) 对任意的正整数 k，对象 p 的 k-近邻 $N_k(p)$由所有距 p 的距离不超过 k-距离的对象组成，即

$$N_k(p)=\{q\in S\mid dist(q,p)\leqslant Dk(p)\}$$

定义 4(平均 k-距离) 对任意正整数 k，对象 p 的平均 k-距离 $LD_k(p)$定义为 p 与 k-近邻中所有对象的平均距离，即

$$LD_k(p)=\frac{\sum_{q\in N_k(p)} dist(q,p)}{|N_k(p)|}$$

定义 5(离群点与正常点) 对象 o 被称为 $DB(k,r)$离群点，当且仅当 o 的 r-邻域的基数小于 k，即 $|N(p,r)|<k$。否则为正常点(“非离群点”)。

注意，$DB(k,r)$是 $DB(p,D)$的等价定义，这里 $k=N*(1-p)$，N 为数据集的大小，$r=D$。

4 算法

算法 RAP(Ranking After Pruning)的目标是报告数据集 S 中的前 n 个离群点。它由两步组成：

第一步(剪枝步) 扫描数据集 S 一遍，并将数据划分到不同的簇中。首先随机指定一个数据点为候选簇中心，凡离某个簇中心的距离小于等于阈值 $r/2$ 的数据点被划分到该簇。与任何簇中心的距离都超过阈值 $r/2$ 的点则成为新的候选簇中心。在划分的同时记录每个簇的基数。在一遍扫描结束后，所有的数据对象都被划分到不同的簇中。接下来，凡是基数不小于 k 的簇被整体剪枝。

第二步(离群度计算步) 对余下的数据进行嵌套循环扫描，计算每个数据点 p 的平均 k-距离，即 $LD_k(p)$，将其作为离群度。离群度最高的 n 个点，被视为 TOP n 离群点。在计算离群度的时候，设定一个阈值 $CUTOFF$(默认为 0)，当某个数据点 p 已探测到的平均 k-距离小于该阈值时，结束关于该点的 k-近邻的查找，因为其不可能为 TOP n 离群点。否则该点将作为候选离群点，加入到 TOP n 离群队列。TOP n 队列中离群度最低的点被移出，次低的离群度晋升为新的离群度阈值。

算法 RAP 用到一种特别的数据结构，称为 KN-Vector。

它是一个由具有 KN-Element 结构的节点组成的队列，并带有一个容量参数 k。KN-Element 包含以下信息：

(1) *id*：数据对象的标识；

(2) *score*：一个记录距离或离群度的数据域。

KN-Vector 可以根据数据域 *score* 的值按指定的顺序(升序或降序)管理列队，自动记录数据域最小(或最大)的 k 个节点，超出的部分被自动移出队列。KN-Vector 还能处理数据域值相等(并列名次)的情况。

图 1 给出了 RAP 算法的轮廓。图 2 描述了其中剪枝的过程。图 3 给出了离群度计算的过程。

```
Algorithm   RAP(Ranking top n outliers after pruning)
Input:      数据集 S,
            近邻个数 k,
            距离参数 r
            返回的离群点个数 n
            离群度阈值参数 cutoff
Output:     前 n 个离群点 O
begin
1
2           P = First-match-pruning(S,k,r)
3           O = Ranking(P,k,n,cutoff)
end         return O
```

图 1　RAP 算法

```
Procedure   First-match-Pruning
Input:      近邻个数 k
            数据集 S,
            距离参数 r
Output:     剪枝后的数据集 P
Begin
1
2           P = S , C = ∅
3           C = C.addcluster(RandomSample())
4           for each object p∈P  {
5               cluster-matched = false;
6               for each cluster c ∈C  {
7                   if  (dist(p,c.center) ≤ r/2 ) {
8                       p.cluster = c
9                       Cluster-matched = true
10                      break
11                  }
12              }
13              if  (not cluster-matched) {
14                  C = C.addcluster(p)
15              }
16          }
17          for each object p∈P  {
18              if  (p.cluster.size() ≥ k ) {
19                  P.remove(p)
20              }
21          }
22          Return P
end
```

图 2　剪枝过程

```
Procedure   Ranking(标记 TOP n 离群点)
Input:      数据集 P
            近邻个数 k
            返回的离群点个数 n
            离群度阈值 cutoff(默认值为 0)
Output:     前 n 个离群点 O
Begin
1
2           O = new KN-Vector(n,Descending)
3           for each object p∈P {
4               KDIST= new KN-Vector(k,Ascending)
5               is-Topn = true
6               for each object q∈P {
7                   KDIST.add(q.id,dist(p,q))
8                   if (rank(p) ≤ cutoff ) {
9                       is-Topn  =  false
10                      break
11                  }
12              }
13              if (is-Topn)  {
14                  O.add(p.id,rank(p))
15                  cutoff = O.last().score
16              }
17          }
18          Return O
end
```

图 3　离群度计算过程

在剪枝步，确定对象 p 的簇归属时，可能会遇到可以划分到多个簇的情况，即它距多个簇中心的距离都不超过 $r/2$。这时，有两种选择：将 p 划分到距其最近的簇或者将其划分到第一个匹配的簇。为减少计算量，这里选取了后者。簇划分完毕后，凡基数大于等于 k 的簇被直接剪枝掉，因为这样的簇不可能包含离群点。

引理 1　如果依算法 RAP 定义的簇 C 的基数 $|C| \geq k$，则该簇不包含任何 $DB(k,r)$ 离群点。

证明：设 p 和 q 为簇 C 中的任意两个数据点，簇 C 的中心点记为 o，则必有：

$dist(p,o) \leq r/2$，且

$dist(q,o) \leq r/2$

根据三角不等式定理，有：

$dist(p,q) \leq dist(p,o) + dist(q,o) \leq r$

于是，对于簇 C 内的任意数据点 p，有 $|N(p,r)| \geq k$，即 p 为正常点而非离群点。证毕。

5　实验结果与分析

算法在合成数据集和真实数据集上都作了实验，并与 LOF 和 ORCA 等算法进行了比较。为保证比较的公平性，所有算法都在同一个平台上实现并运行，各算法直接从文件中读取数据，均未使用任何索引结构。算法用 JAVA 语言编写，在 Eclipse 下编译，在一台配置为 Intel® Core™ 2. 33G 双 CPU、2G 内存的 PC 机上运行。

5.1　合成数据集

图 4 是一个二维合成数据集的分布图。该数据集由 4 个簇组成：100 和 200 个稠密高斯分布的簇各一，200 个密度相异的均匀分布的簇各一。数据集中人为添加了 7 个离群点。它正是文献[5]中采用的一个数据集，我们根据文献[5]的描述，重新生成了这个数据集，用于跟该文献描述的算法进行比较。

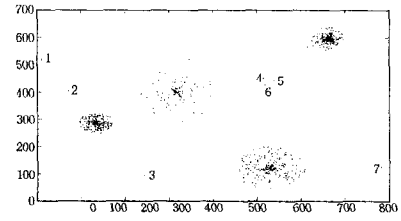

图 4　合成数据集

在对数据集进行了归一化预处理后，分别运用 LOF、NL、ORCA 和本文提出的 RAP 算法对其进行离群点检测。程序记录了各算法运行期间距离计算的次数和运行时间。算法运行时参数取 $k=5$(即 LOF 算法中的 MinPts)，$r=0.05$，$n=7$。实验结果显示，NL、ORCA 和 RAP 算法均在 TOP 7 离群点中准确列出了 7 个离群点。LOF 算法未能在 TOP 7 离群点中列出 7 个离群点中的任何一个。但修正参数 k 为 7 后，LOF 亦能将 7 个离群点准确报告为 TOP 7 离群点。这印证了 LOF 算法对参数的敏感性。表 1 给出了各算法运行期间进行距离计算的次数和运行时间。结果表明，LOF 算法最耗时，RAP 算法用时最短。运行时的记录还显示，在经过一次扫描后，RAP 算法剪掉了 700 个数据点，余下的 7 个数据点正好全部是离群点，故其运行时间最短。在这个小数据集上，ORCA 算法未能体现其剪枝优势，其与 NL 算法所用时间事实上相差无几。

表 1　合成数据集上的运行结果比较

算法	距离计算次数	运行时间
LOF	422063	281 ms
NL	330379	203 ms
ORCA	310339	187 ms
RAP	770	15 ms

5.2　KDD CUP 99 数据集

KDD CUP 99 数据集[14]源自美国国防部高级规划署(DARPA)在 MIT 林肯实验室进行的一项网络入侵检测评估中所采集的真实的网络数据。1999 年的数据挖掘与知识发现(KDD)竞赛因采用此数据而得名。该数据集有 42 个属性，其中 34 个为连续属性，7 个为离散属性，其中包括 1 个类别属性，类别属性值为'normal'的表示正常连接，其它的为各种网络攻击。该数据集一共有 4898431 条记录，其中 972871 条为正常连接，其余的标记为各种攻击。由于该数据集主要用于训练学习攻击模型，因此它收录了很多攻击数据，正常数据只占 19.8%，不符合离群的定义。所以，必须对该数据集进行必要的筛选，以保证正常连接为大多数。为测试数据规模对算法的影响，构建了记录数分别为 500、1000、2000、5000、10000 和 50000 的 6 个数据集(分别称为 D500、D1K、D2K、D5K、D10K 和 D50K 数据集)，每一个数据集都从 KDD CUP 99 数据集中随机抽取，并尽量保持各种攻击类型的比例。数据集中正常连接数保持在 98%～99%左右，网络攻击占 1%～2%，作为离群数据用以检验算法的有效性。42 个属性中，除类别标号未参与离群检测，仅用于与检测的结果进行比较外，其它 41 个属性全部参与了离群检测。在挑选、预处理和检测过程中未使用任何领域知识对数据集中的任何属性进行任何特别的对待。

5.2.1　数据预处理与距离度量

由于该数据集中既有离散属性，又有连续属性，因此需对其进行预处理。所有的离散属性都被映射成正整数。所有的连续属性都作了归一化处理，被映射到[0.0，1.0]区间。同时，由于混合属性的存在，针对连续变量的欧几里德距离并不适用于该数据集。结合文献[15]提出的方法，采用了以下距离度量函数：

$$dist(p,q)=\sqrt{\sum_{i=1}^{d} f(p.A_i,q.A_i)}$$

d 为维度。

当 A_i 为连续属性时，$f(p.A_i,q.A_i)=(p.A_i-q.A_i)^2$。

当 A_i 为离散属性时，如果 $p.A_i=q.A_i$，则 $f(p.A_i,q.A_i)=1$；否则 $f(p.A_i,q.A_i)=0$。

5.2.2　数据规模对运算时间的影响

在接下来的所有实验中，参数 r 的取值均为 1.0。k 和 n 的取值则与数据集的大小成比例。当数据集大小为 50000 时，k 取值为 200，n 取值为 2000。当数据集大小为 500 时，k 取值为 2，n 的取值为 20。图 5 绘出了在 500～10000 5 个不同规模数据集上，LOF、ORCA 和 RAP 算法所用时间的对比图。其中 ORCA 算法对数据集作随机打散的预处理时间未计入在内。图中的 3 条曲线从上至下依次为 LOF、ORCA 和 RAP。结果表明，LOF 算法的时间复杂性接近 $O(N^2)$，ORCA 的时间复杂性接近线性，而 RAP 的时间复杂性为 $O(N)$，远低于其它算法。

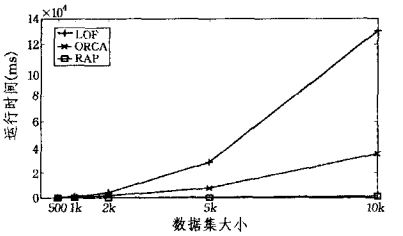

图 5　数据集大小对运行时间的影响

5.2.3　离群检测的结果评价

在机器学习领域，受试者工作特征曲线(Receiver Operating Characteristic，简称 ROC)是度量分类模型好坏的标准[16]。图 6 绘出了在大小为 10000 的这个数据集上 3 个算法的 ROC 曲线。实验结果表明，ORCA 和 RAP 有着相近的性能曲线，在误警率小于 10%时，已能检出 90%以上的离群点。两者的性能明显优于 LOF。

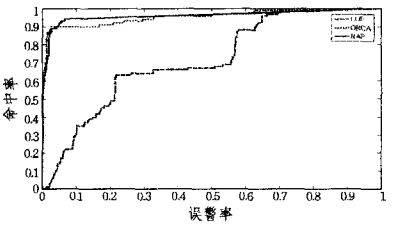

图 6　3 个算法的性能比较

ROC 曲线下的面积 AUC(Area Under roc Curve)用于比较检测模型的好坏。表 2 给出了 3 个算法在全部 6 个数据集上的性能指标 AUC。实验结果表明，RAP 算法在不同规模的数据集上性能表现稳定。而 LOF 算法则随着数据规模的增大，性能略有下降。由于 LOF 算法在 D50K 数据集上未能在有限时间(24 小时)内运算出结果，因此其对应的性能指标栏空缺。

表 2　3 个算法在不同数据集上的 AUC 性能比较

算法	D500	D1K	D2K	D5K	D10K	D50K
LOF	0.884	0.860	0.836	0.832	0.715	
ORCA	0.991	0.990	0.985	0.939	0.955	0.957
RAP	0.978	0.986	0.945	0.957	0.960	0.957

结束语　本文研究了基于距离的高维海量数据集的离群点挖掘问题，提出了一个基于剪枝的离群点挖掘算法。该算法先对数据集进行划分，并剪枝大部分仅由正常数据组成的簇，再对余下的少数数据实施离群度计算。实验结果表明，所提出的算法针对高维海量数据集是有效的，能取得较高的离群点命中率且保持较低的误警率，特别是效率得到明显的提高。进一步改进该算法，使其适用于数据流中的离群点挖掘，是下一步研究的内容。

参 考 文 献

[1] Barnett V, Lewis T. Outliers in Statistical Data (3rd Ed)[M]. New York(New York City): John Wiley & Sons, 1994: 1-2

[2] Knorr E M, Ng R T. Algorithms for mining distance-based outliers in large datasets [C]// Proc of the 24th int'l conf on VLDB. New York (New York City): ACM, 1998: 392-403

[3] Knorr E M, Ng R T, Tucakov V. Distance-based outliers: algorithms and applications [C]// Proc of 26th Int'l conf on VLDB.

Egypt (Cairo): ACM, 2000:. 237-253

[4] Ramaswamy S, Rastogi R, Shim K. Efficient algorithms for mining outliers from large data sets [C]// Proc of ACM SIGMOD Int'l conf on Management of Data. Texas (Dallas): ACM, 2000: 427-438

[5] Angiulli F, Pizzuti C. Fast outlier detection in high-dimensional spaces [C]// Proc of Principles of Data Mining and Knowledge Discovery. 6th European Conf. Finland (Helsinki): ACM, 2002: 15-26

[6] Breunig M M, Kriegel H P, Ng R T. LOF: Identifying density-based local outliers [C]// Proc of ACM SIGMOD Int'l conf on Management of Data. Texas (Dallas): ACM, 2000: 93-104

[7] Bentley J L. Multidimensional binary search trees used for associative searching [J]. Communications of the ACM, 1975, 18 (9): 509-517

[8] Guttmann R. A dynamic index structure for spatial searching [C]// Proc of ACM SIGMOD int'l conf on Management of Data. New York (New York City): ACM, 1984: 47-57

[9] Berchtold S, Keim D, Kreigel H P. The X-tree: an index structure for high-dimensional data [C]// Proc of the 22nd int'l conf on VLDB. Mumbai (Bombay): Morgan Kaufmann, 1996: 28-39

[10] Knorr E M, Ng R T. Finding intentional knowledge of distance-based outliers [C]// Proc of 25th Int'l conf on VLDB. Scotland (Edinburgh): Morgan Kaufmann, 1999: 211-222.

[11] 张净,孙志挥,宋余庆,等. 基于信息论的高维海量数据离群点挖掘[J]. 计算机科学, 2011, 38(7): 148-151

[12] Bay S D, Schwabacher M. Mining distance-based outliers in near linear time with randomization and a simple pruning rule [C]// Proc of 9th ACM SIGKDD int'l conf on KDD. Washington DC: ACM, 2003: 29-38

[13] Angiulli F, Fassetti F. Very efficient mining of distance-based outliers[C]// Proc of ACM 6th conf on Information and Knowledge Management. Portugal (Lisboa): ACM, 2007: 791-800

[14] Hettich S, Bay S. KDD Cup 1999 Data[OL]. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 2011-09-01

[15] Kaufman L, Rousseeuw P J. Finding Groups in Data: An introduction to Cluster Analysis [M]. New Jersey (Hoboken): John Wiley & Sons, 2005: 32-37

(上接第 142 页)

```
16.    Pop(ω);
17.    如果 γ 不是并发动作且动作 ω∈hot(γ)是状态更新操作
18.      S=S∪ω.I−ω.O; //撤销 ω 的执行
19.    如果 hot(γ)集合中还有未被执行过的原子动作
20.      φ=γ;
21.    else {
22.      如果 hot(γ)集合中原子动作是并发关系, ∀ω∈hot(γ)
23.        S=S∪ω.I−ω.O;
24.      Pop(ω);
25.      如果 λ 不是并发动作且动作 ω∈hot(λ)是状态更新操作
26.        S=S∪ω.I−ω.O;
27.  φ=λ;
28.    }
29.  }
30. }while(栈不为空)
31. End
```

算法时间复杂度分析:基于深度优先的回溯算法时间复杂度不高,通常可在线性时间 $O(n+m)$ 内完成。采用的数据结构不同,算法的时间复杂度也不相同。由于服务组合过程(构造服务调度执行的并发和顺序关系)实际上就是在对应于公式 ϕ 的控制流图上的深度优先搜索过程,因此可以借助控制流图来分析算法的时间复杂度。假设控制流图的深度为 n,宽度为 m,原子动作数为 e,则算法的时间复杂度为多项式级别,最坏情况下为:$m*n+e$。

结束语 本文阐述了基于并发事务逻辑 Horn 子集的语义 Web 服务组合方法。从 OWL-S Web 服务功能和行为两个方面对 Web 服务进行了表示与推理。重点分析了 Web 服务的并发和同步并发行为的建模方法。采用并发 Horn 公式作为组合推理形式,提出将服务组合方案的生成归结为并发 Horn 目标公式可满足性的证明,并给出了对约束的预处理方法。本文最后提出一个多项式时间的服务组合算法,并分析了算法的实践复杂度。仅用服务功能描述的基本输入输出变量作为表示服务行为的动作执行的效果,没有充分利用 OWL-S 提供的丰富语义描述,这是本方法的不足之处。下一步工作将把该算法用于具体网络应用中,对算法的实际运行效率进行实验,并考虑结合描述逻辑推理丰富 Web 服务的功能描述来组合 Web 服务。

参 考 文 献

[1] Rao J, Su X. A survey of automated Web service composition methods[C]// Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition. San Diego, USA, 2004: 43-54

[2] Bonner A J, Kifer M. Concurrency and communication in transaction logic[C]// Proceedings of the Joint internal Conference and Symposium on Logic Programming. Bonn, Germany, MIT Press, 1996: 142-156

[3] OWL-S 1.2 Release [OL]. http://www.ai.sri.com/daml/services/owl-s/1.2/, 2011-11-20

[4] Horrocks I. Ontologies and the semantic Web [J]. Communications of the ACM, 2008, 51(12): 58-67

[5] Bonner A J, Kifer M. Transaction logic programming[R]. Computer Systems Research Institute Technical Report CSRI-323. University of Toronto

[6] Rao Jing-hai, Küngas P, Matskin M. Composition of Semantic Web services using Linear Logic theorem proving [J]. Information Systems, 2006, 31(4/5): 340-360

[7] Baader F, Lutz C, Milicic M, et al. A Description Logic Based Approach to Reasoning about Web Services[C]// Proceedings of the WWW Workshop on Web Service Semantics: Towards Dynamic Business Integration. Chiba, ACM Press, 2005: 636-647

[8] 史忠植. 常亮. 基于动态描述逻辑的语义 Web 服务推理[J]. 计算机学报, 2008, 31(9): 1600-1611