ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*, Vol.21, No.4, April 2010, pp.632−643
doi: 10.3724/SP.J.1001.2010.03565

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 基于监控对象动态聚簇的高效 RFID 数据清洗模型*

谷 峪[1,2], 于 戈[1,2+], 胡小龙[2], 王 义[2]

[1](医学影像计算机教育部重点实验室(东北大学),辽宁 沈阳 110004)

[2](东北大学 信息科学与工程学院,辽宁 沈阳 110004)

## Efficient RFID Data Cleaning Model Based on Dynamic Clusters of Monitored Objects

GU Yu[1,2], YU Ge[1,2+], HU Xiao-Long[2], WANG Yi[2]

[1](Key Laboratory of Medical Image Computing (Northeastern University), Ministry of Education, Shenyang 110004, China)

[2](School of Information Science and Engineering, Northeastern University, Shenyang 110004, China)

+ Corresponding author: E-mail: yuge@mail.neu.edu.cn

**Gu Y, Yu G, Hu XL, Wang Y. Efficient RFID data cleaning model based on dynamic clusters of monitored objects. *Journal of Software*, 2010,21(4):632−643.** http://www.jos.org.cn/1000-9825/3565.htm

**Abstract**: Because wireless radio frequency signal is adopted during the RFID (radio frequency identification) communication, many false negative and false positive readings may be produced leading to inaccurate event detection. In RFID-based applications, monitored objects often progress in the form of group. In this paper, by analyzing the group changes based on the defined association degree and dynamic clusters, a series of models and algorithms about association degree maintenance and data cleaning are proposed. Especially, by compressing the graph model, an association degree maintenance strategy based on splitting and recombining list model is discussed to improve the time and space performance. Simulated experiments have demonstrated the efficiency and accuracy of the proposed data cleaning model.

**Key words**: RFID (radio frequency identification) technology; data cleaning; dynamic cluster; association degree; false negative data

**摘 要**: 由于 RFID(radio frequency identification)技术采用无线射频信号进行数据通信,漏读和多读现象时有发生,降低了其在事件检测中查询结果的准确性.在很多 RFID 监控应用中,监控物体都是以动态变化的小组为单位进行活动的.通过定义关联度和动态聚簇对各个 RFID 监控物体所在的小组进行动态的分析,并在此基础上定义了一套关联度维护和数据清洗的模型和算法,通过对图模型进行压缩,提出了基于分裂重组思想的链模型关联度维护策略,提高了维护的时空效率.模拟实验结果表明,该数据清洗模型可以获得较好的效率和准确性.

**关键词**: RFID(radio frequency identification)技术;数据清洗;动态聚簇;关联度;漏读数据

**中图法分类号**: TP311 **文献标识码**: A

RFID(radio frequency identification)技术融无线通信、嵌入式计算、数据管理和信号处理等技术为一体,受到人们的广泛关注.该技术主要通过无线射频信号在阅读器与标签间进行数据通信,从而探测到贴有标签的

* Supported by the National Natural Science Foundation of China under Grant Nos.60773220, 60873009, 60933001 (国家自然科学基金)

Received 2008-05-15; Revised 2008-11-27; Accepted 2009-01-15

物体的逻辑位置.RFID 技术的最早应用就是识别对象.但是,随着对 RFID 技术的深入研究,其应用领域逐渐扩大.目前,RFID 技术已经可以支持基于事件监控等更高级的应用.RFID 技术应用的广泛性使其成为国内外各高等院校和科研机构研究的焦点.然而,RFID 阅读器的读数具有很大的不可靠性,如果将源数据直接投入到实际应用中,得到的结果一般都没有应用价值,所以在对 RFID 源数据进行处理之前,对数据进行清洗[1−5]是非常重要的.而数据的不可靠性通常是指下面两种情况:(1) 漏读(false negative)是指,当一个标签在一个阅读器阅读范围之内时,该阅读器没有读到该标签.(2) 多读(false positive)是指,当一个标签不在一个阅读器阅读范围之内时,该阅读器仍然读到该标签.经调查表明,在最差情况下,阅读器只能读到在其感应范围内 60%~70%的数据,即有30%的数据在读取时被遗漏,而数据多读现象发生的概率相对较小.

值得注意的是,在很多 RFID 监控应用中,组的概念是广泛存在的.例如在一个博物馆参观监控场景中,来自一个班级的学生和一个单位的同事可能会志愿组成小组进行参观.再例如在一个战场模拟场景中,战士可能要分组完成任务.这种分组存在两个典型的特点:(1) 动态性:整个小组的成员可能会不断地变化.例如,参观人员根据兴趣找到新的伙伴一同参观;战士们为了执行新任务分成更小的组或者进行重组.(2) 干扰性:不同部门的人可能也会在某些时刻汇聚到同一地点,在物理上构成了一个组,但在逻辑上属于不同的组.例如,参观场景中每个展台都可能混杂着不同单位的人员;在某个战场双方的士兵遭遇等.这些特点对组的判断提出了挑战.不难发现,组的概念其实可以渗透到很多重要的 RFID 应用模型中,例如对各批货物、各队车辆、各群牛羊的监控.显然,这种概念上的小组为数据清洗提供了很好的参照,可以根据分析出来的小组其他成员的读数信息来填补某个成员的漏读数据,去除多读数据.

为了解决上述问题,本文提出了基于对监控对象动态聚簇概念的 RFID 数据清洗策略,通过有效的聚簇建模和高效的关联度维护来估算真实的小组,在此基础上进行有效的清洗.由于引入了新的维度,在有小组参与的情况下,无论数据量的大小和组变化的程度,与只考虑时间维的现有方法相比,本文提出的模型可以有效地利用组间成员的关系来提高清洗的准确性.本文的贡献主要有下面几点:

- 对 RFID 动态聚簇和相应清洗问题的理论建模.
- 提出了一系列 RFID 监控对象关联度维护的模型和算法,特别是提出了一套基于分裂重组的链模型.
- 用大量模拟实验证明了各种模型和算法的高效性和有效性.

本文第 1 节介绍相关工作.第 2 节对 RFID 动态聚簇和清洗进行理论建模.第 3 节描述高效维护监控对象关联度的算法体系.第 4 节对一些相关的重要问题进行讨论和分析.第 5 节通过实验对相关的模型和算法进行评估.最后总结全文并对未来工作进行展望.

## 1 相关工作

RFID 数据管理已经得到了越来越多的关注,主要工作集中在 RFID 管理框架研究[6]、数据清洗研究、以数据为中心的管理建模[7]和以事件为中心的高效维护算法上[8,9].其中,对 RFID 应用中的数据清洗技术的研究已经取得了一定的进展.文献[1]提出了一种基于管道框架进行过滤的技术,主要解决了漏读和多读(即噪声数据)问题.文献[2]主要通过概论统计方法对漏读的数据进行了填补,使之尽可能地接近真实数据,但文章中只考虑了单个阅读器读取数据的情况,不适合大数据量和多逻辑区域参与的 RFID 监控场景.文献[3]讨论了多读、漏读、冗余和乱序的综合清洗机制,框架的实现相对简单,针对大数据量下的数据漏读问题,准确性和效率较差.文献[4]提出了以机器学习为背景的清洗算法来分析各个已有时序清洗策略对应的清洗开销,但该模型的候选算法是已有的基于时序的平滑技术.因此,上述清洗算法本质上都是根据自身历史数据的特点进行平滑,没有考虑到监控对象之间的关系和时空的闭合性,在有小组参与的漏读率较高的环境下,准确性较差.例如,某物体连续通过 *A*,*B*,*C* 位置,在 *B* 位置所有数据漏读,如果没有其他对象作参照物,仅仅通过对自身历史读数的分析是无法得出正确的结果的.此外,由于 RFID 应用的特点,定义的是不同标签物体的关联度为聚簇的准则,模型和算法都完全不同于传感器网络中基于空间关联的清洗方法[10,11],从概念上也与数据流上的聚簇问题[12]完全不同.综上所述,本文是通过动态监控 RFID 标签对象的分组信息对漏读和多读数据进行清洗的开创性研究.

## 2 RFID 动态聚簇和清洗建模

### 2.1 动态聚簇建模

本节首先定义一些重要的概念和性质.当被附着标签的对象进入阅读器的感知范围内时,就会产生一个RFID 的读数.将标签对象标记为 $o_i, i\in\mathbb{N}$,并将读数的元数据建模为 $r_i=(r\sim EPC, o\sim EPC, \tau)$,其中 $r\sim EPC$ 和 $o\sim EPC$ 分别表示阅读器和标签对象的 EPC(electronic product code)编码,$\tau$ 表示该对象被感知的时间戳.由于RFID 现在还不能有效地支持跨区域的应用,因此,EPC 编码目前还仅仅是一个理论上的标准.在实际的小规模应用中,一般都采用预先设定的连续编码来标记监控对象和阅读器,这相当于将 EPC 编码映射成为一个逻辑的id.例如 $S_o$ 表示 $o\sim EPC$ 集合,可以得到一个单射 $f:\mathbb{N}\to S_o$,这样就可以用 $o_i, i\in\mathbb{N}$ 来表示一个标签对象.此外,一个或多个阅读器对应一个逻辑区域,表示实际应用中的某一个指定位置或者步骤,例如博物馆的一个展台.用 $\Upsilon_i, i\in\mathbb{N}$ 来标记一个逻辑区域,同理,$S_r$ 表示 $r\sim EPC$ 集合可以指定映射 $g:\mathbb{N}\to\rho(S_r)$ 和 $g':S_r\to\mathbb{N}$,将多个 $r\sim EPC$ 映射成为某个 $\Upsilon_i$.将 $t$ 时刻在整个监控场地内(例如整个博物馆)标签对象的集合标记为 $\Omega(t)$,$t$ 时刻某个逻辑区域 $\Upsilon_i$ 探测到的物体定义为 $\Upsilon_m(t)=\{o_k \mid \forall k.\exists i.(g(m)=r_i.r\sim EPC\wedge f(k)=r_i.o\sim EPC\wedge t=r_i.\tau)\}$.需要说明的是,阅读器一般都会预先布置在指定的逻辑区域,位置不会在应用中发生变化;而标签对象 $o_i$ 将动态地穿梭于各个 $\Upsilon_m$,因此 $\Upsilon_m(t)$ 是动态变化的.此外,还会存在新的 $o_i$ 进入和原先的 $o_i$ 退出整个监控场地的情况,因此 $\Omega(t)$ 也是动态变化的,即 $\Omega(t)=\Omega(t-1)\bigcup\Omega_+(t)-\Omega_-(t)$,其中 $\Omega_+(t)$ 和 $\Omega_-(t)$ 分别表示在 $t$ 时刻进入和退出监控场地的标签对象.在实际应用中,可以通过统计监控场地入口和出口位置的读数信息来直接确定 $\Omega_+(t)$ 和 $\Omega_-(t)$.

对于进入监控场地的标签对象,在逻辑上可能存在一定的联系(例如来自同一个单位的参观者).这样从逻辑上将所有的标签对象抽象为一些大组 $\Sigma_i$,$\Omega(t)=\bigcup_i\Sigma_i$,但从读数信息上,并不会知道具体的 $\Sigma_i$ 信息.在整个监控过程中,每个 $\Sigma_i$ 会不断地分裂和重组成许多小组 $g_m$,某个小组表示在某一时刻一起行进的 $\Sigma_i$ 中的成员.明显地,$\forall i,k,\exists m,o_i,o_k\in g_m(t)\Rightarrow\exists n,o_i,o_k\in\Upsilon_n(t)$,但是因为此时的 $o_i$ 和 $o_k$ 可能来自不同的 $\Sigma$,所以反之并不成立.要通过合理地建模,动态聚簇出 $g'_m(t)$,作为对 $g_m(t)$ 的有效估计,进一步根据 $g'_m(t)$ 成员作为参照物来进行数据清洗.下面将定义一些重要的概念.

**定义 1(全覆盖区域 EOR(entire overlay region))**. 对于一个监控场地,如果所有的位置都能被阅读器探测到,将其称为一个全覆盖区域.即 $\Omega(t)=\bigcup_m\Upsilon_m(t)\cup\Phi(t)$.其中 $\Upsilon_m(t)\cap\Upsilon_n(t)=\varnothing, m\neq n$,$\Phi(t)$ 表示 $t$ 时刻的漏读数据集合.

**定义 2(部分覆盖区域 POR(partial overlay region))**. 对于一个监控场地,如果有些区域并没有部署阅读器,将其称为一个部分覆盖区域.没有部署阅读器的位置可能代表不同 $\Upsilon_i$ 之间的路径和一些没有特定含义的地点,将时刻出现在这些位置的 $o_i$ 集合定义为 $\Upsilon'(t)$.在 POR 场景中,$\Upsilon'(t)=\Omega(t)-\bigcup_m\Upsilon_m(t)-\Phi(t)$,其中 $\Upsilon_m(t)\cap\Upsilon_n(t)=\varnothing, m\neq n$.

**定义 3(标签对象关联度 $\delta$)**.根据 RFID 应用的特点,$t$ 时刻标签对象 $o_i$ 和 $o_k$ 的关联度 $\delta_t(o_i,o_k)$ 定义为

$$\delta_t(o_i,o_k)=\begin{cases}\delta_{t-1}(o_i,o_k)+1, & \text{iff } \exists l, o_i,o_k\in\Upsilon_l(t)\\ \delta_{t-1}(o_i,o_k), & \text{iff } o_i,o_k\in\Upsilon'(t)\\ 0, & \text{otherwise}\end{cases}.$$

$\delta_t(o_i,o_k)$ 表示了 $o_i$ 和 $o_k$ 在相同的逻辑区域被连续探测到的次数,反映了 $o_i$ 和 $o_k$ 在一个聚簇 $g'_m(t)$ 中的可能性.此外.在 EOR 场景中,不需要考虑 $o_i,o_k\in\Upsilon'(t)$ 的情况.为了更好地说明问题,首先基于 EOR 场景给出 $\delta_t(o_i,o_k)$ 的一些重要性质.POR 场景的特殊问题将会在第 4.2 节的讨论中给出.

**性质 1**. $o_i\in\Upsilon_m(t)\wedge o_k\in\Upsilon_n(t)\wedge m\neq n\Rightarrow\delta_t(o_i,o_k)=0$.证明略.

**性质 2**. $\delta_t(o_i,o_k)=\delta_t(o_k,o_i)$.证明略.

**性质 3**.

$\forall i,k,l,(\delta_t(o_i,o_k)\geq\delta_t(o_i,o_l)=\delta_t(o_k,o_l))\vee(\delta_t(o_i,o_l)\geq\delta_t(o_k,o_l)=\delta_t(o_i,o_k))\vee(\delta_t(o_k,o_l)\geq\delta_t(o_i,o_k)=\delta_t(o_i,o_l))$.

证明:假设 $o_i,o_k,o_l$ 在同一逻辑区域 $\Upsilon_m(t)$,如果 $o_i,o_k$ 第 1 次在这个周期内出现在同一逻辑区域 $\Upsilon_m(t')$ 的时刻

为 $t'$,即 $\forall t' \le \tau < t, \delta_{\tau+1}(o_i,o_k)=\delta_\tau(o_i,o_k)+1 \wedge \delta_{t'}(o_i,o_k)=1$, 如果 $o_i,o_l$ 第 1 次在这个周期内出现在同一逻辑区域 $\Upsilon_m(t'')$ 的时刻为 $t''$, 同理,即 $\forall t'' \le \tau < t, \delta_{\tau+1}(o_i,o_l)=\delta_\tau(o_i,o_l)+1 \wedge \delta_{t''}(o_i,o_l)=1$. 如果假设 $t''>t'$, $\forall t'' \le \tau < t$, $\delta_{\tau+1}(o_k,o_l)=\delta_\tau(o_k,o_l)+1 \wedge \delta_{t'}(o_k,o_l)=1$,即 $\delta_t(o_i,o_k) \ge \delta_t(o_i,o_l)=\delta_t(o_k,o_l)$. $t''<t'$ 和 $t''=t'$ 的其他情况同理可证. □

性质 3 是本文维护算法进行数据压缩的重要性质.

**定义 4(强聚簇关系Δ).** Δ定义为一个$\Omega(t)$上的二元关系,表示 $o_i$ 与 $o_k$ 之间存在最大的关联度.如果 $i=k, o_i\Delta o_k; i\ne k, o_i\Delta o_k \Leftrightarrow (\delta_t(o_i,o_k)=\max\{\forall l,\delta_t(o_i,o_l)\}) \wedge (\delta_t(o_i,o_k)=\max\{\forall l,\delta_t(o_k,o_l)\})$.

**定义 5(弱聚簇关系∇).** ∇为一个$\Omega(t)$上的二元关系,表示$\delta_t(o_i,o_k)$是 $o_i$ 或者 $o_k$ 参与的最大的关联度.如果 $i=k, o_i\nabla o_k; i\ne k, o_i\nabla o_k \Leftrightarrow (\delta_t(o_i,o_k)=\max\{\forall l,\delta_t(o_i,o_l)\}) \vee (\delta_t(o_i,o_k)=\max\{\forall l,\delta_t(o_k,o_l)\})$. 弱聚簇关系使得任何对象 $o_i$ 都能找到一个包含其他对象的聚簇,这样在概念上才能为数据清洗提出参照的对象.如果 $o_i\Delta o_k \Rightarrow o_i\nabla o_k$.

**定义 6(弱最大相似集$\Theta_t$).** 一个对象 $o_i$ 的弱最大相似集定义为与 $o_i$ 有着最大关联度的对象集合,即 $\Theta_t(o_i)=\{o_k \mid \delta_t(o_i,o_k)=\max\{\forall l,\delta_t(o_i,o_l)\}\}$.可以证明, $\forall o_k \in \Theta_t(o_i) \Rightarrow o_i\nabla o_k$.

**定义 7(弱最大代表元$\theta_t$).** 一个对象 $o_i$ 的弱最大代表元定义为与 $o_i$ 有着最大关联度的任何一个对象,即 $\theta_t(o_i)=o_k, o_k \in \Theta_t(o_i)$.

**定义 8(强最大相似集 $\Theta_t^{\alpha-\max}$).** 将 $\Theta_{t-1}(o_i)$ 按照 $t$ 时刻的Δ关系进行划分,每一个等价类为 $\Theta_t^{\alpha}(o_i)$,则 $\Theta_t^{\alpha-\max}(o_i)=\max|\Theta_t^{\alpha}|$.

**定义 9(强最大代表元 $\theta_t'$).** $\theta_t'(o_i)=o_k, o_k \in \Theta_t^{\alpha-\max}(o_i)$. $\theta_t'$ 将会在 $o_i$ 发生漏读时提供更准确的估计.

本节所定义的模型和性质为后面的相关算法提供了理论基础.

### 2.2 基于动态聚簇的数据清洗建模

本节分别定义了发生漏读和多读情况下的数据清洗模型.每种情况下都会考虑全覆盖区域 EOR 和部分覆盖区域 POR 的情况,并根据采用强最大代表元或弱最大代表元,分为随机选取策略和最大划分策略.

- $o_i$ 发生漏读,在全覆盖区域 EOR 的情况下,不考虑 $\Upsilon'(t)$ 的影响,则计算 $o_i \in \Omega(t)-\bigcup_m \Upsilon_m(t)$.如采用随机选取策略,需要计算 $o_k=\theta_{t-1}(o_i)$,采用最大划分策略,需要计算 $o_k=\theta_t'(o_i)$,之后添加 $(o_k.r \sim EPC, o_i.o \sim EPC, t)$.在部分覆盖区域 POR 的情况下,需要考虑 $\Upsilon'(t)$ 的影响.这个时候通过计算 $\Omega(t)-\bigcup_m \Upsilon_m(t)$ 无法直接确定 $o_i \in \Phi(t)$ 或是 $o_i \in \Upsilon'(t)$,需要根据组员的情况来加以判断.如采用随机选取策略,计算 $o_k=\theta_{t-1}(o_i)$,如果 $o_k \in \Upsilon'(t)$,则认为 $o_i \in \Upsilon'(t)$,不需要进行漏读数据的填补.相反地,如果 $o_k \in \Phi(t)$,则认为 $o_i \in \Phi(t)$,此时需要填补 $(o_k.r \sim EPC, o_i.o \sim EPC, t)$.如果采用最大划分策略,则求得 $o_k=\theta_t'(o_i)$,按类似的方法进行操作.
- $o_i$ 发生多读,无论在全覆盖区域 EOR 还是在部分覆盖区域 POR 的情况下,都可以计算 $o_i \in \Upsilon_m(t) \wedge o_i \in \Upsilon_n(t) \wedge m \ne n$.采用随机选取策略,需要求出 $o_k=\theta_{t-1}(o_i)$,保留 $(o_k.r \sim EPC, o_i.o \sim EPC, t)$.采用最大划分策略,需要求出 $o_k=\theta_t'(o_i)$,保留 $(o_k.r \sim EPC, o_i.o \sim EPC, t)$.

可以看出,最大划分策略因为考虑了当前时刻的分组情况,比随机选择策略有着更高的可靠性,但也带来了额外的开销.在实际应用中,一般采用随机选取策略.此外,当数据发生漏读或者多读现象时,根据代表元的情况来维护关联度,可以保证性质 3 的正确性.不难理解,如果动态聚簇的目的是为了高效的清洗,并不需要直接得到聚簇关系,而只需要高效而增量地维护各个对象在每个时刻的最大代表元$\theta_t$,并且在任意对象发生漏读或者多读数据的情况下快速地得到对应的$\theta_t(o_i)$,而$\theta_t(o_i)$又依赖于任意两个对象 $o_i$ 和 $o_k$ 的关联度$\delta_t(o_i,o_k)$.因此一个最关键的问题是如何高效地增量维护$\delta_t(o_i,o_k)$.在下一节中,首先以 EOR 场景下漏读情况的随机选取策略为背景,来说明增量维护$\delta_t(o_i,o_k)$的模型和算法.之后在第 4 节讨论其他数据清洗情况.

## 3 关联度增量维护算法

本节将讨论关联度增量维护的算法,这些算法是进行聚簇和数据清洗的基础.假设有一个循环使用的连续的 id 标识,当对象退出监控场景时,收回对应的标签 id,并循环发送给新进入场馆的标签对象.因此,理论上只要

维护监控场地所能容纳的最大标签对象数目或者参与监控的最多标签对象数目 $N_{\max}$ 即可,从而简化了模型的构建.如果采用 EPC 编码,需要进行相应的逻辑 id 的哈希操作.$N$ 表示实际在$\Omega(t)$中的对象数目,$N_r$ 表示逻辑区域的数目,$N_a=N/N_r$ 表示每个逻辑区域 $\Upsilon_m(t)$ 包含的标签对象个数的平均值.图 1 描述了整个关联度维护的模型转化过程,下面将分小节来逐步说明相关算法的优化.

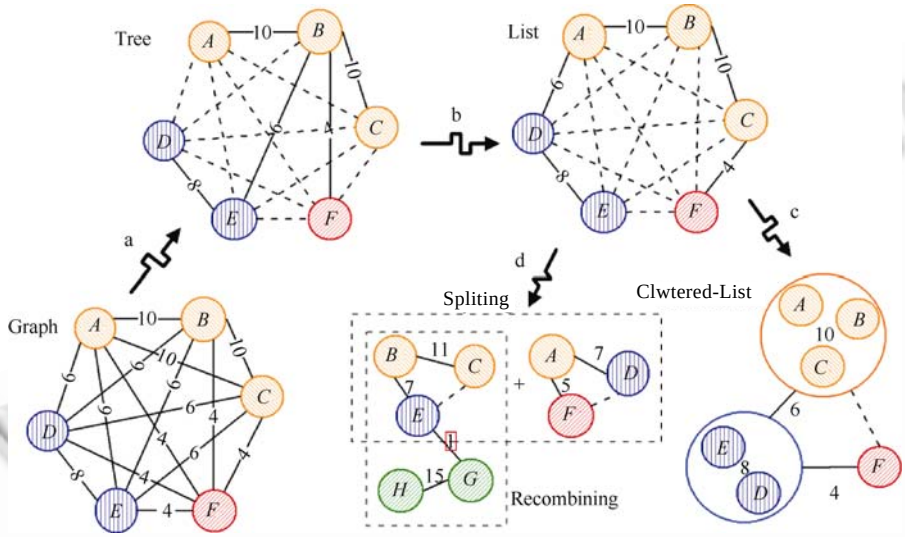


Fig.1 Illustration of the transformations of different association degree maintenance strategies

图 1 各种关联度模型的转化示意图

### 3.1 基于$\delta$-图模型

因为什么时刻发生漏读,哪个对象发生漏读是无法事先预知的,每个时刻的$\theta_t(o_i)$到下一时刻可能会换成别的对象.因此,为了快速得到$\theta_t(o_i)$,必须高效地维护各个时刻任意两个对象的$\theta_t(o_i,o_k)$,显然,图模型直观地存储了需要的数据.每一个 $\Upsilon_m(t)$ 对应一个图 $G_m(t)=(V,E)$,$V$ 表示顶点的集合,$E$ 表示边的集合. $G_m(t).V=\Upsilon_m(t)$, $G_m(t).E=\{(o_i,o_k,w)\mid \forall o_i,o_k\in\Upsilon_m(t)\wedge i<j,w=\delta_t(o_i,o_k)\}$.可以看出,这是一个带权重的全联通图, $\forall o_i,o_k\in G_m(t)\Rightarrow\delta_t(o_i,o_k)\neq 0$.相反地,对于 $o_i\in G_m(t),o_k\in G_n(t),m\neq n\Rightarrow\delta_t(o_i,o_k)=0$.图 1 中的 Graph 图示给出了某个 $\Upsilon_m(t)$ 对应的 $G_m(t)$,边上的数字对应于指定的权重.

为了利用连续的 id,可以直观地将$\bigcup_m G_m(t)$用一个大的矩阵 $M_t$ 来表示,并且 $M_t[i][k]=\delta_t(o_i,o_k)$,如果 $o_i,o_k$ 位于同一个逻辑区域 $\Upsilon_m(t)$,将 $M_t[i][k]$加 1;反之,将 $M_t[i][k]$清 0.这样就可以得到对应的 $M_t$.基于矩阵的算法直观地反映了任意两个对象的关联度,算法易于实现.在维护空间代价上,二维数组的设计需要 $O(N^2)$的复杂度;在维护时间代价上,只需维护一个三角矩阵,也即只需维护一半的数据位;还可以将每个 $\Upsilon_m(t)$ 内的标签对象 id 进行排序,按照顺序进行操作位增量和标志位修改的操作,以简化操作次数,但维护时间复杂度依然为 $O(N^2)$.当探测出每个漏读数据 $o_i$ 时,需要通过对 $o_i$ 所在行的排序来找到$\theta_t(o_i)$,因此查找代价为 $O(N\log N)$.本文将基于矩阵的图维护方法称作$\delta$-Graph$^M$ 方法.

基于矩阵的图结构虽然实现简单,但由于采用二维数组进行存储,当 $N$ 取较大值时,算法将无法运行,而且随着 $N$ 的增加,维护时间代价会变得很大.我们可以采用哈希链结构存储邻接表来压缩矩阵,提高在 $N_a$ 较小情况下图结构的查询效率.一个邻接链表相当与一个哈希链表 $H_t$,其中 $H_t[i]$连接了一个链表结构,包含了与 $o_i$ 在 $t$ 时刻位于同一逻辑区域 $\Upsilon_m(t)$ 的对象 $o_k$ 和对应的$\delta_t(o_i,o_k)$.$H_t[i]$对应链表的每项(item)数据结构为 $(id,\delta_t(o_i,o_{id}))$.一般来说,该策略需要 $O(NN_a^2)$ 的维护时间代价,空间复杂度同样为 $O(N^2)$,但由于每个 Item 需要存储 id 和$\delta$两个元素,因此空间代价要高于$\delta$-Graph$^M$.因为每次都将$\delta$=1 的新 Item 插入对应的链表,而且在维护过程中并没有改变原有 Item 的顺序,因此可以证明 $\theta_t(o_i)=H_t[i].head$.所以,当发现漏读数据时,相应的时间复杂度为 $O(1)$.本文将该维护策略称作$\delta$-Graph$^L$ 方法.

### 3.2 基于压缩的$\delta$-树模型

以上基于图的方法并没有很好地考虑关联度的一些性质,特别是性质 3,因此,存储的过程中包含一些冗余的数据,下面首先通过定理 1,证明压缩$\delta$-树模型的可行性.

**定理 1**. 任意一个基于图模型的 $G_m(t)$都可以无损地用一个树模型 $T_m(t)$来表示.

证明:采用数学归纳法来证明.$|G_m(t)|=1$或者$|G_m(t)|=2$,本身就是一棵树.假设$|G_m(t)|=n$能用树 $T_m(t)$来表示 $G_m(t)$中所有的边,则对于一个新加入的对象 $o_i$,一定能够找到一个 $o_k \in T_m(t)$,使得 $\delta_t(o_i,o_k)=\max\{\forall o_l \in T_m^n(t),\delta_t(o_k,o_l))$,即 $T_m^n(t)$ 中与 $o_k$ 有着最大关联度的某个顶点 $o_k$,并在 $o_i$ 和 $o_k$ 间加入一条边和对应的权重,这样可以得到一棵树 $T_m^{n+1}(t)$.对于 $\forall o_l \in T_m^{n+1}(t)$,因为 $\delta_t(o_i,o_k) \geq \delta_t(o_i,o_l)$,根据关联度性质 3,可得 $\delta_t(o_i,o_l)=\delta_t(o_k,o_l)$,因为 $\delta_t(o_k,o_l)$ 在 $T_m^n(t)$ 中可得,因此,$o_i$ 与 $T_m^n(t)$ 中任意对象的关联度都可得,即$|G_m(t)|=n+1$也能用一棵树 $T_m^{n+1}(t)$来表示.根据数学归纳法,证毕. □

例如,对于一个有 4 个对象结点的图,图 2 图示了这种转化的过程.

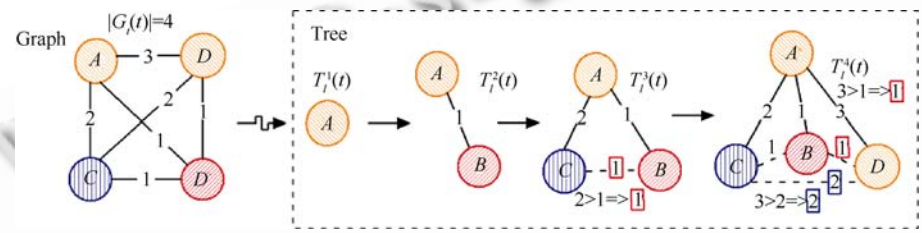


Fig.2 Transformation from $\delta$-Graph to $\delta$-Tree

图 2 $\delta$-Graph 与$\delta$-Tree 的转化

需要说明的是,在本文中,仅将树模型作为一种更好的说明从图到链的过渡模型,在实际应用中,采用时空效率更高的链模型来设计.通过对下面章节的比较可以证明,采用树模型进行维护,平均和最好的时间和空间代价都不会超过基于分裂和重组的链模型.具体细节这里不再详述.

### 3.3 基于压缩和分裂重组的$\delta$-链模型

#### 3.3.1 模型建立

**定理 2**. 任意一个基于图模型的 $G_m(t)$都可以无损地用一个链模型 $L_m(t)$来表示.

证明:采用数学归纳法.$|G_m(t)|=1$或者$|G_m(t)|=2$本身就是一个链,假设$|G_m(t)|=n$能用一个链 $L_m^n(t)$ 来表示 $G_m(t)$中所有的边,则对于一个新加入的对象 $o_i$,一定能够找到 $L_m^n(t)$ 中与 $o_i$ 有着最大关联度的某个结点 $o_k$,并在 $o_i$ 和 $o_k$ 间加入一条边和对应的权重,这样可以得到一棵树 $T_m^{n+1}(t)$.根据性质 3,$\delta_t(o_i,o_k) \geq \delta_t(o_i,o_k.neighbor)$.如果 $\delta_t(o_i,o_k) > \delta_t(o_i,o_k.neighbor)$,断开 $o_k$ 和 $o_k.neighbor$,连接 $o_i$ 和 $o_k.neighbor$,可得链表 $L_m^{n+1}(t)$.如果 $\delta_t(o_i,o_k)=\delta_t(o_i,o_k.neighbor)$,断开 $o_i$ 和 $o_k$,连接 $o_i$ 和 $o_k.neighbor$,按照上面的步骤递归,直至到达链表的头或者尾,也一定能获得一个链表 $L_m^{n+1}(t)$.根据数学归纳法,证毕. □

具体地,$L_m(t)$可以用一个单向链表来实现,其中某一数据项(item)包括 $(id,\delta=\delta_t(o_{id}.previous,o_{id}))$,$\delta$域表示了该结点对象与链表中前驱结点对象的关联度.特别地,因为头结点没有直接前驱结点,设 $L_m(t).head.\delta=0$.

**定理 3**. 在一个$\delta$-链 $L_m(t)$中,任意两个对象 $o_i$ 和 $o_k$ 的关联度$\delta_t(o_i,o_k)$等于它们之间所有$\delta$域的最小值 min$\delta$.

证明:根据性质 3,可以推出 $\delta_t(o_i,o_i.next.next)=\min\{o_i.next.\delta,o_i.next.next.\delta)$, 依此类推,直到 $o_i.next^*=o_k$,可得 $\delta_t(o_i,o_k)=\min\{o_i.next.\delta,...,o_i.next^*)$, 得证. □

定理 3 为实现高效的分裂算法提供了依据,$L_m(t)$在 $t$+1 时刻分裂成 $m$ 个子链,这些子链中$\delta$的计算可以通过对 $L_m(t)$ 的直接遍历获得.

**定理 4**. 两个子链 $L_m^a(t)$ 和 $L_m^b(t)$,如果 $L_m^a(t)$ 与 $L_m^b(t)$ 在 $t$–1 时刻不属于同一个逻辑区域,而在 $t$ 时刻属于同一个逻辑区域,则只需将两个链首尾相连,并将 $L_m^b(t).head.\delta$ 置 1,即可构成一个新链 $L_m(t)$.

证明:由于 $L_m^a(t)$ 和 $L_m^b(t)$ 中对应的数据项相互位置并没有改变,$L_m^a(t)$ 和 $L_m^b(t)$ 中各自对象的关联度可以按定理 3 计算得到,而因为任意的 $\delta$ 域必然不小于 1,因此根据定理 3,可以由添加的边求出 $\forall o_i \in L_m^a(t)$ 和 $\forall o_k \in L_m^b(t)$, $\delta_t(o_i,o_k)=1$,这样,$L_m(t)$中任意两个对象的关联度都可以正确地得到.证毕. □

定理 4 为实现高效的重组算法提供了理论依据,在 $t$ 时刻 $n$ 个子链重新构成 $L_m(t)$,可以通过直接相连这些子链得到.

**定义 10**. 分裂度 $N_p$.定义某一 $L_m(t)$在 $t$+1 时刻分裂成的子链的个数称为分裂度,所有 $L_m(t)$分裂度的平均值用 $N_p$ 表示.分裂度反映了应用中聚簇的规模和程度、聚簇动态变化的频度、应用场景中干扰特性存在的程度.如果组的规模越大,变化频度越小,干扰性越小,分裂度就越小.不难理解,$1 \le N_p \le \min(N_a, N_r)$.

特别地,因为所有子链的 $\delta$ 域都是从 1 开始逐渐增加的,这就不需要每一时刻将图转化为树,再转化为链.而是基于分裂重组的思想,从开始就构成链,并设计高效的增量维护的链算法.

3.3.2 基于多遍子链扫描策略 $\delta$-List$^{\alpha}$

上述模型和定理证明为设计高效的链维护算法提供了理论基础.在本节中,首先提出一个基于定理 3 和定理 4 的经过优化的维护策略 $\delta$-List$^{\alpha}$.该策略体现了分裂重组的思想,对每个子链进行多次扫描,直到所有分裂全部完成.该算法增量地将 $L(t-1)$ 转换为 $L(t)$,算法的主要步骤描述如下:

Step 1. 依次遍历每个 $\delta$ 子链,在每个子链中,循环执行 Step 2~Step 5;

Step 2. 循环执行 Step 3~Step 5,直到该子链 $L_m(t-1)$为空;

Step 3. 遍历该子链,取出与当前头结点逻辑位置相同的结点形成一个链 $L'$,$\delta$ 按定理 3 增量维护并加 1;

Step 4. 遍历的同时将子链 $L'$中的结点从 $L_m(t-1)$中增量地删除;

Step 5. 将 $L'$与对应的 $t$ 时刻代表相同逻辑区域的子链 $L_n(t)$连接,连接处的 $\delta$ 按定理 4 维护.

通过对算法进行分析,可以得出该算法的处理代价与分裂度 $N_p$ 有关,例如当 $N_p$=1 时,维护每个 $L_m(t)$的代价只须 $N_a$;而当 $N_p = N_a (N_a < N_r)$ 时,维护每个 $L_m(t)$的代价为等差数列 $N_a + N_a - 1 + \ldots + 1 = (1+N_a)N_a/2$,随着 $N_p$ 的增加,处理代价会逐渐地增大,因此,修改算法具有从 $O(N_rN_a)$到 $O(N_rN_a^2)$的时间复杂度.在空间代价上,由于采用链表进行了有效的压缩,存储每个 $L_m(t)$的代价为 $O(N_r)$.显然,由于引入了压缩下的分裂重组思想,该算法比起基于图的策略,在时空性能上都有大幅的提高.因为可以证明,$\theta_t(o_i)$一定是 $o_i$ 的邻居结点,因此,当探测出每个漏读数据 $o_i$ 时,需要遍历所有的结点并记录直接前驱结点,直到找到相应的数据,需要 $O(N)$的响应时间.

3.3.3 基于单遍子链扫描策略 $\delta$-List$^{\beta}$

$\delta$-List$^{\beta}$算法在 $N_p$ 较大的情况下,可能会达到 $O(N_rN_a^2)$的时间复杂度,本节给出一个通过连续记录每个位置的最小值,只遍历一遍链表的增量维护策略,该策略与 $N_p$ 无关,是一种稳定的算法.该算法同样增量地将 $L(t-1)$转换为 $L(t)$,算法的主要步骤描述如下:

Step 1. 依次遍历每个 $\delta$ 子链,在每个子链中,循环执行 Step 2~Step 5;

Step 2. 依次遍历该子链 $L_m(t-1)$的每个结点,循环执行 Step 3~Step 5;

Step 3. 取出该结点的 $\delta$ 域,遍历所有子链的 min$\delta$,如果 $\delta$<min$\delta$,该子链的 min$\delta$ 赋值为 $\delta$;

Step 4. 如果该结点是 $L_m(t-1)$中第 1 个插入到 $t$ 时刻对应逻辑区域子链的,对应子链的 min$\delta$ 置 0;

Step 5. 将该结点插入对应的逻辑区域子链,$\delta$ 域设为 min$\delta$+1,此后将该 min$\delta$ 置极大值.

通过进一步的算法分析可以得出,使用该策略维护每个 $L_m(t)$的代价为 $O(N_aN_r)$,即每个时刻的总维护代价为 $O(N_aN_r^2)$.由于比起 $\delta$-List$^{\alpha}$少存储了相应的结构,在空间代价上略好于 $\delta$-List$^{\alpha}$.在清洗响应时间上同样为 $O(N)$.

3.3.4 基于区间树策略 $\delta$-List$^{\gamma}$

在求一个 $\delta$ 链中任意两个对象关联度的时候,可以考虑构造一棵便于查找的区间树.图 3 展示了利用区间树求关联度的过程,对于一个有 7 个对象构成的 $\delta$ 链,首先按照顺序将关联度 $\delta$ 索引到一个数组中,之后根据数组的 index 作为结点进行建树,如果结点个数为偶数,则这棵树是一个完全的二叉树.叶结点依次为所有的 index,任意一个中间结点为两个叶结点中 index 对应关联度 $\delta$ 较小的那个 index,并在结点上记录该结点包含的 index

的最大值和最小值,整个树的构建采用二分的递归算法,假设每个逻辑区域对应的平均对象个数为 $N_a$,建这棵区域树的复杂度为 $O(N_a\log N_a)$,但是,由于所有的非叶子结点都需要新开辟的空间,因此需要额外存储代价,此外,该算法还需要额外的建立索引的代价.在进行某两个对象关联度的查找时,需要从跟结点开始,采用二分法,直到找到对应的左边界结点和右边界结点,并将对应的两个关联度进行比较,找出较小的关联度,该数值即为要求的解,这个过程需要 $O(\log N_a)$的复杂度.综上,对于一个链表来说,这个过程需要的时间复杂度为 $O(N_a\log N_a)$,实际处理的代价应为 $kN_a\log N_a, k>1$,其中,$k$ 受两个监控对象在链表中的相互位置的影响.在分裂度较大的情况下,该算法会优于$\delta$-List$^{\alpha}$,在逻辑区域个数 $N_a$ 较大的情况下,该算法会优于$\delta$-List$^{\beta}$.清洗漏读数据的响应时间同样为 $O(N)$.相关的算法可以通过修改$\delta$-List$^{\alpha}$算法得到.在图 3 中,对于两个对象 id=10 和 id=32 的关联度,通过索引和建树的过程,最后找到了用斜体标识的 index=4 和 index=6 的结点,通过比较关联度,得出$\delta(o_{10},o_{32})$=6.

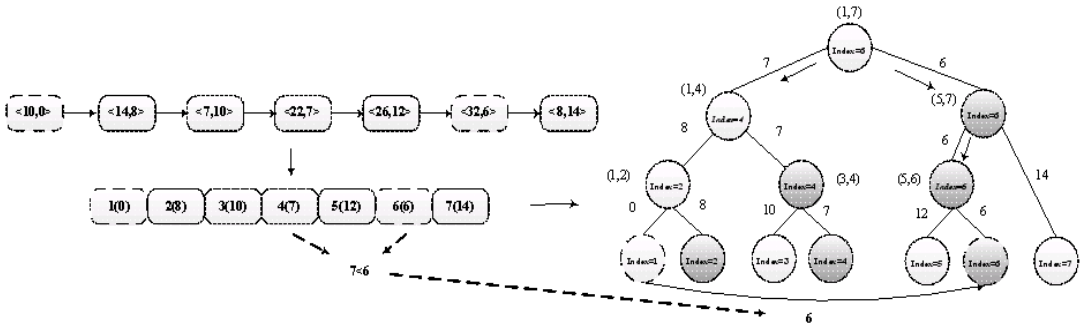


Fig.3 Illustration of construction and search for interval tree

图 3 区域树建立和查找示意图

### 3.4 基于聚簇$\delta$-链模型

根据强聚簇关系,将所有满足强聚簇关系Δ的结点构成一个更大的“聚簇结点”,每个聚簇结点只需存储一个$\delta$值,因此可以将子链进一步压缩,在某些聚簇包含对象个数较多的情况下,能够进一步节省空间.因为可以证明,一个对象的最大代表元与该对象是邻接结点,因此该优化并不会影响分类和重组的时间代价,而且可以保证每个链表存储的$\delta$值小于 $N_a$.特别是在一些不仅仅需要通过代表元进行数据清洗,而且还需要实时进行聚簇监控的应用中,该算法还有更好的实时性.基于聚簇$\delta$-链模型算法,可以通过对上面基于链的算法进行修改得到,限于篇幅,这里不具体对算法进行说明.图 1 的 c 步骤图示了这一过程.

## 4 一些相关问题的说明

### 4.1 关于清洗准确度分析

不难看出,如果某个物体发生漏读,则其在上一时刻的最大代表元此时与该物体还在一个聚簇中,该清洗算法就会得到准确的结果.假设每个强聚簇关系Δ构成的等价类在某一时刻发生分裂的概率为 $P_c$,每个聚簇平均分裂的个数为 $N_c$,则 $P(o_i,o_k\in g'_m(t)\wedge o_i,o_k\in g'_n(t+1))=1-P_c(1-1/N_c)$.一般在概念上有组参与的 RFID 监控应用中,因为频繁的聚簇分裂本身违背了小组的特征,因此,$P_c$ 和 $N_c$ 的值都会非常小,所以本文提出的算法会达到很高的准确率.例如当 $P_c$=0.2,$N_c$=2 时,这在一般的小组参与的场景中是可以保证的,此时 $P(o_i,o_k\in g'_m(t)\wedge o_i, o_k\in g'_n(t+1))$=0.9,有着很高的清洗准确率.需要说明的是,$N_c$ 反映的是聚簇的分裂程度,$N_p$ 反映的是子链的分裂程度,它们的含义并不相同;$N_p$ 较大可能仅仅代表场景中存在着较大的干扰特性,并不代表 $N_c$ 也会较大.同时,漏读率也是影响最后清洗结果的主要因素,特别是当漏读率较大时,同一组内在某个逻辑区域内所有的数据全部漏读,可能造成清洗结果的错误.综上,本文提出的数据清洗模型更适合于 $P_c$ 相对较小的场景,数据量的大小并不会影响数据清洗的效果.比起时序平滑的方法,本文的模型可以适用于更大的数据量和漏读率,有着很好的健壮性.对于 $P_c$ 很大的情况,可以设定聚簇清洗的阈值.当关联度小于该阈值时,结合已有的时序平滑方法.对于组

概念不明显的应用场景,可以提高清洗的准确度.

### 4.2 POR场景讨论

上述讨论都是针对全覆盖区域 EOR 展开的,但在实际应用中,部分覆盖区域 POR 也是非常普遍的,比起EOR,除了各个 $\Upsilon_m(t)$ ,POR 需要多考虑一个特殊的逻辑区域 $\Upsilon'(t)$ ,根据本文关联度$\delta$的定义, $\Upsilon'(t)$ 中的各个物体也独立构成了一个图或者链,上述模型和算法通过修改可适用于 POR 场景.如果整个场景的逻辑区域个数为 $N_r$,需要维护的子链个数为 $N_r$+1,则对维护算法的性能几乎不会造成影响.在维护关联度的过程中,可以用队列来保留那些没有被任何位置探测到的物体集合 $\Upsilon'(t)$ ,但在这种策略下,因为每个时刻 $t$ 都需要对 $\Upsilon'(t)$ 中的对象判断是否是漏读数据,因此需要一些额外的开销,而且采用随机选取策略,准确度会受到一定的影响,我们会在未来工作中重点地对 POR 场景的模型优化进行深入研究.

### 4.3 多读和最大划分策略讨论

与漏读数据相比,多读数据发生的概率要小很多,而且造成的影响和危害也比较小.通过对关联度的高效维护和代表元的确定,可以像填补漏读数据一样来高效地去除多读数据,通过对物体逻辑编号的额外标志数组进行维护,也可以在 $O(N)$的复杂度内完成对多读数据的判断.因此,本文的模型和算法适用于对多读数据的清洗.最大划分策略会得到更高的清洗准确性,但会在发生漏读或者多读数据时带来一定的额外开销,延长了数据清洗的相应时间.但在大多数情况下,由于采用随机选取策略的清洗算法准确度已经很高,这种准确度的提升并不明显,所以一般不采用这种策略.

## 5 实验分析

### 5.1 实验设置

本实验的硬件环境是兼容机 2.4GHz 的 Pentium 4 CPU,主存为 1G,硬盘为 120GB.软件环境是 Windows XP 操作系统,Virtual Studio C++6.0 编程环境,算法采用 C 语言编写.由于受到硬件条件和场地的限制,无法得到真实的数据集.参照相关文献的做法,我们仿真了一个博物馆的参观场景,并对时间代价、空间代价和准确性进行评估.在进行时间代价和空间代价的测试时,采用了随机模拟的方法,此时 $N_p$ 较大,分裂重组很多,可以对算法效率的健壮性有一个很好的评估.此外,因为随机模拟的特点,可以通过调解 $N_a$ 和 $N_r$ 来间接地反映 $N_p$ 变化对各种算法效率的影响.在比较准确性时,为了进一步体现出小组的特点,通过对 $P_c$ 和 $N_c$ 的控制来调节应用场景数据,进而说明了漏读率对清洗效果的影响.

### 5.2 时间代价分析

首先对各种算法在小数据量($N$ 取值较小)情况下的时间性能进行实验,$T$ 表示模拟实验运行的时间.图 4 给出了调整 $N_r$ 的实验结果,由于基于链的算法要远远优于基于图的算法,可将它们区分在不同的子图中进行显示.例如图 4 结果所示,基于图的算法要比基于链的算法平均维护时间多 100 倍,这主要因为$\delta$-图的算法复杂度系数一般为 $N$,而$\delta$-链算法复杂度系数为 $N_r$;但由于数据量太小,各种基于链的算法并不稳定,对比关系也并不明显,在 $N_a$ 较小的情况下,$\delta$-Graph$^L$ 的维护时间会优于$\delta$-Graph$^M$.

在实际的应用场景中(例如博物馆监控),监控对象的个数最多可以达到 10 万的规模,逻辑区域个数也要大于 100.当数据量变得很大时,基于概率清洗模型的时序平滑方法和基于聚簇的$\delta$-图策略根本无法正常运行,本文提出的$\delta$-链优化算法体现出了更大的优势.图 5(a)显示了 $N$=$10^4$ 时,各种算法的维护性能.因为$\delta$-List$^\beta$和$\delta$-List$^\gamma$所有子链维护代价分别为 $NN_r$ 和 $kN\log N_a$,因此,随着 $N_r$ 的增加,即 $N_a$ 的减小,$\delta$-List$^\beta$和$\delta$-List$^\gamma$的维护代价会分别体现出增加和减小的趋势.而$\delta$-List$^\alpha$比较特别,因为$\delta$-List$^\alpha$维护代价受 $N_p$ 的影响很大,在随机模拟情况下,随着 $N_r$ 的增大,$N_p$ 也会相应地增大,此时,复杂度由 $O(N_rN_a)$逐渐向 $O(N_rN_a^2)$过渡,因此,处理代价在开始阶段出现了增加的趋势.当 $N_r$>$N_a$ 时,分裂度 $N_p$ 最大只能为 $N_a$,$N_a$ 与 $N_r$ 成反比,因此,处理代价在达到最大值后随着 $N_a$ 的减小出现了减小的趋势.图 5(b)显示出在调整 $N$ 时,各种算法的维护代价,通过分析,也可以得出相应的结

论.通过该实验可以看出,在大数据量情况下,不同的策略在不同的参数下分别会有最优的性能,因此可以考虑通过分析相应参数来适应性地选择 3 种不同的$\delta$-链策略.

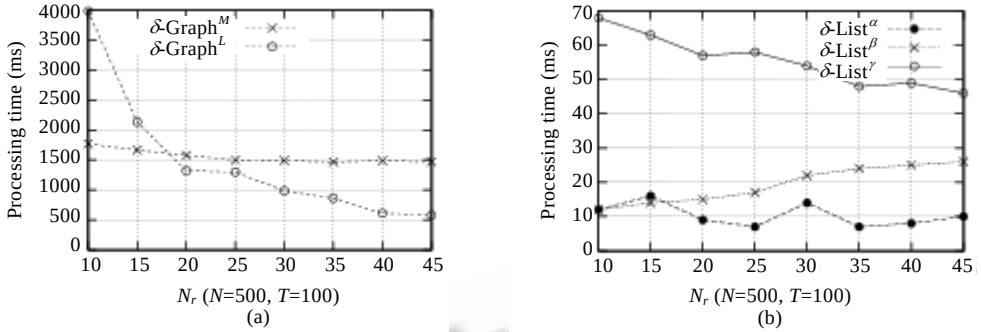


Fig.4 Time performance comparison of different algorithms for small data amount

图 4 小数据量下的算法时间性能比较

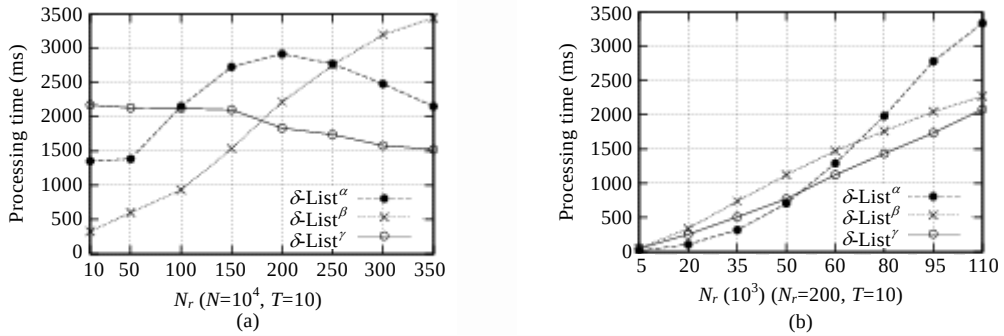


Fig.5 Time performance comparison of different algorithms for large data amount

图 5 大数据量下的算法时间性能比较

### 5.3 空间代价分析

本节对各种算法的存储代价进行了分析,因为要处理 RFID 数据流,所以,所有的数据都是存储在内存当中的.在小数据量的情况下,从图 6(a)的实验结果可以看出,$\delta$-链比起$\delta$-图算法,由于只需要 $O(N)$的复杂度,其存储代价非常小(在图中甚至无法显示),而因为$\delta$-Graph$^L$ 需要多存储对应的 id 号,因此代价更大.但$\delta$-Graph$^M$ 由于采用二维数组,当$N$较大时,程序将无法运行,因此$\delta$-Graph$^L$的空间健壮性稍好一些.此外,图 6(b)表明,$N_r$对$\delta$-图算法的空间代价几乎没有影响.

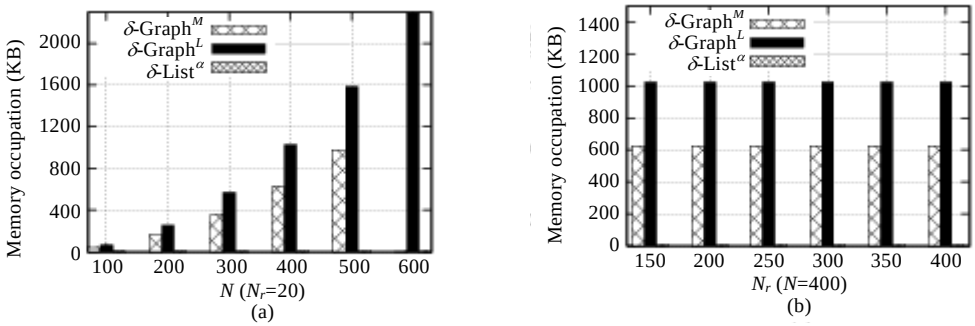


Fig.6 Space performance comparison of different algorithms for small data amount

图 6 小数据量下的算法空间性能比较

在大数据量的情况下,从图 7 的实验结果可以看出,由于采用压缩技术,存储代价依然维持在较小的范围

内,$N_r$ 也会对存储代价产生影响.其中,$\delta$-List$^{\beta}$附加数据结构少于$\delta$-List$^{\alpha}$,存储代价较小,由于$\delta$-List$^{\gamma}$需要维护区域树,造成了较大的存储开销.特别地,由于聚簇的$\delta$-链算法将同一个聚簇内的结点压缩成为一个“聚簇结点”来减少对冗余$\delta$值的存储代价,可以在一定程度上进一步节省空间.因此,在内存受限的情况下,可以考虑采用聚簇的$C$-$\delta$-List$^{\beta}$或$\delta$-List$^{\beta}$策略.

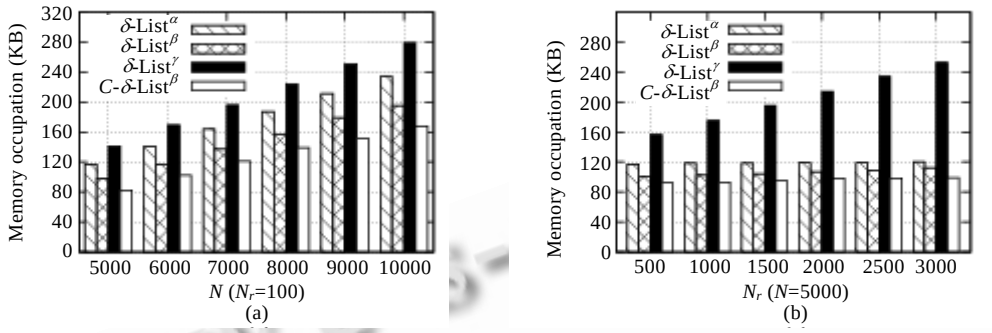


Fig.7 Space performance comparison of different algorithms for large data amount

图 7 大数据量下的算法空间性能比较

### 5.4 准确性分析

本节对数据清洗的效果进行分析,选定 $N$=10000,$N_r$=200.因为所有的基于动态聚簇的方法只是清洗的效率不同,但都不会影响数据清洗的结果.因此,将本文提出的基于动态聚簇的方法统称为 dynamic cluster 方法,并与文献[1]提出的窗口平滑方法进行比较.在大数据量、多逻辑区域参与的情况下,该方法是比较理想的时序平滑方法.从图 8 的实验结果可以看出,本文提出的方法在有组参与的环境中比起时序平滑的方法有着更好的清洗效果.随着漏读率的增加,清洗的准确率有所下降.与时序平滑的方法相比,基于动态聚簇的策略在漏读率很大的情况下,仍然可以保持较高的准确率.

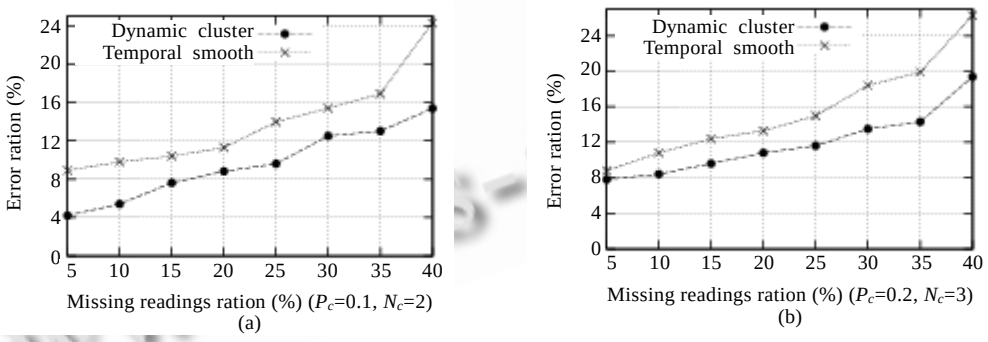


Fig.8 Cleaning accuracy comparison of different strategies

图 8 不同策略的清洗准确性比较

## 6 结 论

RFID 技术有着广泛的应用前景,同时也对数据管理提出了新的挑战.由于 RFID 自身的特点,数据清洗问题是 RFID 数据管理需要考虑的重要问题.结合 RFID 监控应用的特点,本文从一种全新的角度考虑了数据清洗的模型和算法,不同于以往考虑单个监控对象历史读数的清洗技术,通过判断监控对象之间的联系来动态地估计对象所在的小组,并高效地维护小组的变化过程,之后根据组内成员的信息来进行高效的数据清洗.本文对一种图的关联度模型进行分析,之后讨论了压缩的树模型和基于分裂重组思想的链模型,并进行了一系列的查询优化.同时,通过适当的修改,本文的优化算法也可以有效地扩展到其他场景.实验结果表明,由于引入了压缩和分裂重组等重要思想,$\delta$-链算法在时间和空间上都要优于$\delta$-图算法,特别是在大数据量情况下有着很好的健壮性,

在不同的条件下,各种$\delta$-链算法可以达到最优的性能,而 $C$-$\delta$-List$^{\beta}$和$\delta$-List$^{\beta}$策略有着更理想的存储代价.比起时序平滑的方法,本文提出的基于动态聚簇的模型在有组参与的 RFID 监控场景中,有着更好的清洗效果.在未来的工作中,我们将根据不同的场景参数提出适应性的维护策略,并计划进一步对 POR 场景下的模型和框架进行研究.

**References**:


[1] Jeffrey SR, Alonso G, Franklin MJ, Hong W, Widom J. A pipelined framework for on line cleaning of sensor data streams. In: Liu L, Reuter A, *et al*., eds. Proc. of the 22nd Int'l Conf. on Data Engineering. Atlanta: IEEE Computer Society, 2006. 140−142.

[2] Jeffery SR, Garofalakis M, Franklin MJ. Adaptive cleaning for RFID data streams. In: Dayal U, Whang KY, *et al*., eds. Proc. of the 32nd Int'l Conf. on Very Large Data Bases. Seoul: ACM, 2006. 163−174.

[3] Bai Y, Wang FS, Liu PY. Efficiently filtering RFID data streams. In: Proc. of the 1st Int'l VLDB Workshop on Clean Databases. Seoul: Morgan Kaufmann Publishers, 2006. 50−57. http://pike.psu.edu/cleandb06/papers/CameraReady_102.pdf

[4] Gonzalez H, Han J, Shen X. Cost-Conscious cleaning of massive RFID data sets. In: Proc. of the 23rd Int'l Conf. on Data Engineering. Istanbul: IEEE Computer Society, 2007. 1268−1272. http://www.cs.uiuc.edu/~hanj/pdf/icde07_hagonzal.pdf

[5] Khoussainova N, Balazinska M, Suciu D. Towards correcting input data errors probabilistically using integrity constraints. In: Chrysanthis RK, Jensen CS, *et al*., eds. Proc. of the 5th ACM Int'l Workshop on Data Engineering for Wireless and Mobile Access. Chicago: ACM, 2006. 43−50.

[6] Bornhövd C, Lin T, Haller S, Schaper J. Integrating automatic data acquisition with business processes experiences with SAP's auto-ID infrastructure. In: Nascimento MA, Özsu MT, *et al*., eds. Proc. of the 30th Int'l Conf. on Very Large Data Bases. Toronto: Morgan Kaufmann Publishers, 2004. 1182−1188.

[7] Wang FS, Liu PY. Temporal management of RFID data. In: Bohm K, Jensen CS, *et al*., eds. Proc. of the 31st Int'l Conf. on Very Large Data Bases. Trondheim: ACM, 2005. 1128−1139.

[8] Wu E, Diao Y, Rizvi S. High-Performance complex event processing over streams. In: Chaudhuri S, Hristidis V, *et al*., eds. Proc. of the SIGMOD2006. Chicago: ACM Press, 2006. 407−418.

[9] Demers A, Gehrke J, Panda B, Riedewald M, Sharma V, White W. Cayuga: A general purpose event monitoring system. In: Proc. of the CIDR 2007. Asilomar: Morgan Kaufmann Publishers, 2007. 412−422. http://www.cs.cornell.edu/johannes/papers/2007/2007-CIDR-Cayuga.pdf

[10] Vuran MC, Akyildiz IF. Spatial correlation-based collaborative medium access control in wireless sensor networks. IEEE/ACM Trans. on Networking (TON), 2006,14(2):316−329. [doi: 10.1109/TNET.2006.872544]

[11] Abbasi A, Ghadimi E, Khonsari A, Yazdani N. A distributed clustering algorithm for fault-tolerant event region detection in wireless sensor networks. In: Proc. of the ISPA Workshops. Niagara Falls: Springer-Verlag, 2007. 493−502. http://www.springerlink.com/index/j25858t324n7474r.pdf

[12] Aggarwal C, Han J, Wang J, Yu PS. A Framework for clustering evolving data streams. In: Freytag JC, Lockemann PC, *et al*., eds. Proc. of the VLDB2003. Berlin: Morgan Kaufmann Publishers, 2003. 81−92.



**谷峪**(1981－),男,辽宁鞍山人,博士生,CCF 学生会员,主要研究领域为 RFID 数据管理,数据流.

**胡小龙**(1985－),男,硕士生,主要研究领域为 RFID 数据管理.

**于戈**(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库理论与技术.

**王义**(1961－),男,博士,教授,博士生导师,主要研究领域为实时系统.