计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2020.20190207
57(6): 1323-1334, 2020

支持 RFID 供应链路径追溯查询的偏增向量编码策略

廖国琼 杨乐川 张海艳 杨仙佩
(江西财经大学信息管理学院 南昌 330013)
(liaoguoqiong@163.com)

An Offset Addition Vector Coding Strategy for Supporting Path Tracing Query in RFID-Based Supply Chains

Liao Guoqiong, Yang Lechuan, Zhang Haiyan, and Yang Xianpei
(*School of Information Management, Jiangxi University of Finance and Economics, Nanchang* 330013)

Abstract As an important technical support of intelligent Internet of things, radio frequency identification (RFID) technology has been widely used in supply chain tracing and real-time monitoring. In order to improve the tracking and query efficiency of tagged objects in the RFID-based supply chain environment, it is necessary to effectively encode the temporal and spatial data of RFID objects. Considering the characteristics including big volume, the existence of the loop and frequent update of the data in RFID-based supply chains, based on the idea that infinite vectors can be inserted between a pair of vectors, an offset addition vector path coding strategy is proposed. This strategy takes spatiotemporal data nodes as coding objects and assigns a unique pair of vectors to each node by vector addition to achieve uniform coding. At the same time, an optimization strategy is proposed to solve the overflow problem caused by excessive code value, and the correctness of the optimization strategy is proved. The experimental results show that the proposed vector encoding strategy and its optimization strategy can meet the requirements of different types of tracing queries, and have the advantages of fast encoding speed, slow overflow of code value, high update efficiency and can support loop.

Key words radio frequency identification (RFID); path coding; vector coding; tracing query; Internet of things (IoT)

摘 要 作为智慧物联的重要技术支撑,无线射频识别(radio frequency identification, RFID)技术,已广泛用于供应链等物品追溯及实时监控领域.为提高基于 RFID 供应链环境中标签对象路径追溯查询效率,须对 RFID 时空数据进行有效编码.考虑到 RFID 供应链数据具有海量性、存在环路、更新频繁等特点,在 2 个向量之间可以插入无限个向量的思想基础上,提出了一种偏增向量路径编码策略.该策略以时空数据结点为编码对象,利用向量加法给结点分配唯一 1 对向量,实现对每个结点时空信息的统一编码.同时,针对码值过大导致的溢出问题提出了优化方案,并进行了正确性证明.实验结果表明:所提出的偏增向量路径编码策略及其优化策略能满足不同类型追溯查询需求,且具有编码速度快、码值溢出速度慢、更新效率高和支持环路等优点.

收稿日期:2019-03-25;**修回日期**:2019-10-08
基金项目:国家自然科学基金项目(61262009,61772245);江西省自然科学基金重点项目(20151BBG70046);江西省教育厅科技重点项目(GJJ160419)
This work was supported by the National Natural Science Foundation of China (61262009, 61772245), the Key Program of the Natural Science Foundation of Jiangxi Province of China (20151BBG70046), and the Key Project of Science and Technology of Jiangxi Education Department (GJJ160419).

关键词 无线射频识别;路径编码;向量编码;追溯查询;物联网

中图法分类号 TP391

无线射频识别(radio frequency identification, RFID)技术是一种非接触性自动识别技术[1].由于具有无需人工干预、批量识别等优点[2-3],该技术已广泛应用于供应链、图书馆、交通等领域,以实现对物品的全程跟踪及监控.但如何对物品在流通过程中产生的海量信息进行编码以支持路径追溯查询,是基于 RFID 供应链系统面临的主要挑战[4-6].

通常,衡量一个编码策略性能指标主要有查询效率、更新开销及存储开销等[7].然而,供应链环境中路径编码还应考虑 3 个问题:

1) 环路问题.物品在供应链中流通时,可能多次经过同一地点,即在路径中出现环路.例如,在路径 $A\rightarrow B\rightarrow A$ 中,物品 2 次经过地点 A.因此,良好的路径编码策略应能对环路进行描述.

2) 实时更新问题.为满足实时查询需求,编码策略应能及时将来自 RFID 读写器的原始数据转化成编码数据,即能对位置信息进行实时更新,实现随到随编.

3) 溢出问题.海量 RFID 数据会导致编码的码值超出规定数据类型的表示范围,称为溢出现象.一个好的编码策略应能减缓码值的增长速率.

RFID 供应链中物品最初提出的路径编码方法为素数编码[8-9],即对路径中结点分配唯一素数,并利用素数乘积因式分解唯一的特性进行解码.素数编码在查询方面具有较高效率,但其不能解决环路问题,且面对海量 RFID 数据时,素数增长过快,码值容易溢出.为解决单一编码的不足,文献[10]提出了一种素数编码与区间编码相结合的复合编码(composite coding, CC)策略,即对路径的位置信息采用素数编码,时间信息采用区间编码[11-12],并针对溢出问题提供了路径分裂方法,能满足路径查询需求,但此方法不能描述环路,且更新代价较大,不利于实时更新.

针对环路问题,文献[13]提供了一种在素数编码下描述环路的方法,其主要思想为将环路中重复的位置给予相同的素数,在计算同余值时按结点重复次数以幂次进行计算区分.此方法有效地编码了环路,但增加了同余值计算复杂度,且未考虑时间信息.文献[14]提出一种改进的素数编码方法,位置信息编码充分利用值较小的素数,时间信息编码则利用可持久的区间编码(durable region based numbering, DRB)方法编码,能减缓素数溢出,但减缓程度有限,且解码时耗较大,影响查询效率.文献[15]提出了一种路径分裂策略对路径编码树进行分裂,但查询会涉及多张表,导致查询开销增加.

RFID 供应链系统还有一类针对减少存储空间的压缩编码策略,如 pid 编码[16]和基于 T 树的路径压缩策略[17].类似前缀编码,pid 编码存储会产生较大冗余.基于 T 树的策略不涉及具体编码,而是用 T 树来构造编码树提升查询效率,但此方法在构建树的过程中开销较大.

综上所述,已有 RFID 对象路径编码方法都只适用于一定场合,不能完全解决供应链环境路径编码的环路、实时更新及溢出等问题.

向量编码(vector coding)是针对 XML 文档提出的一种层次编码方法[18],该编码基于 2 个向量之间可以无限插入向量的思想对一个对象分配一对向量,具有较高的查询效率和更新效率.但向量编码的码值是基于区间编码产生的,即先对结点进行区间编码,然后再将区间转化成向量进行编码,不能随到随编,实现实时更新.而且,不能解决海量 RFID 数据导致的码值溢出现象.

本文将在原始向量编码基础上,研究一种偏增向量编码(offset addition vector coding, OAVC)策略及优化策略.该策略可直接对 RFID 对象的时空信息进行向量编码,可有效提高编码效率,且在综合考虑查询效率、更新效率及支持环路、溢出等方面具有较好性能.

1 供应链下 RFID 路径表示及编码框架

附有 RFID 标签的物品在供应链上流通时,会被不同位置的阅读器识别,生成形如(Tag_{ID}, Loc, $Time$)形式的原始数据流,其中 Tag_{ID} 是标签对象的唯一标识,Loc 和 $Time$ 分别表示识别地点及时间.图 1 是来自不同阅读器 48 条未加工原始数据样例.

通常,RFID 读写器会以固定周期扫描其探测区域内 RFID 标签对象,因此同一个标签对象可能被同一读写器扫描多次,故原始数据中存在大量冗余信息.实际上,我们只需记录标签在某位置的进入时间和离开时间,即将同一个标签对象在同一个位置的多条原始数据转换成 1 条数据存储,即 Tag_{ID}:

$Loc[ST,ET]$,其中 ST 表示 Tag_{ID} 对象进入位置 Loc 的时间,ET 表示 Tag_{ID} 离开 Loc 的时间.

$(Tag_1,A,1),(Tag_1,A,2),(Tag_2,A,1),(Tag_2,A,2),$
$(Tag_3,A,1),(Tag_3,A,2),(Tag_4,C,1),(Tag_4,C,2),$
$(Tag_4,C,3),(Tag_5,C,1),(Tag_5,C,2),(Tag_5,C,3),$
$(Tag_6,C,1),(Tag_6,C,2),(Tag_6,C,3),(Tag_1,B,4),$
$(Tag_1,B,5),(Tag_1,B,6),(Tag_2,B,4),(Tag_2,B,5),$
$(Tag_2,B,6),(Tag_4,B,4),(Tag_4,B,5),(Tag_5,F,4),$
$(Tag_5,F,5),(Tag_5,F,6),(Tag_6,F,4),(Tag_6,F,5),$
$(Tag_6,F,6),(Tag_7,D,5),(Tag_7,D,6),(Tag_7,D,7),$
$(Tag_3,B,5),(Tag_3,B,6),(Tag_3,A,7),(Tag_3,A,8),$
$(Tag_3,A,9),(Tag_5,D,7),(Tag_5,D,8),(Tag_5,D,9),$
$(Tag_8,B,8),(Tag_8,B,9),(Tag_6,E,11),(Tag_6,E,12),$
$(Tag_6,E,13),(Tag_6,E,14),(Tag_8,C,13),(Tag_8,C,14)$

Fig. 1 RFID raw data record

图 1 RFID 原始数据记录

图 2(a)是图 1 中 48 条原始数据转换后的数据.可以看出,每条数据可描述某物品在某位置的停留时间.例如 $Tag_1:B[4,6]$表示标签对象 Tag_1 在时刻 4 进入位置 B,时刻 6 离开,故 Tag_1 在位置 B 的停留时间为 2.

为便于查询,我们在转换后的数据中再添加 1 个层级属性 l,用于描述对象在移动路径中的位置序号.例如 $Tag_1:B[4,6,2]$表示 B 是 Tag_1 路径中第 2 个经过的位置.

于是,我们将属于同一标签对象的数据进行组合,即可得到形如 $Tag_{ID}:L_1[ST_1,ET_1,l_1]\rightarrow L_2[ST_2,ET_2,l_2]\rightarrow\cdots\rightarrow L_n[ST_n,ET_n,l_n]$的路径信息,如图 2(b)所示.

$Tag_1:A[1,2]$ $Tag_1:B[4,6]$ $Tag_2:A[1,2]$
$Tag_2:B[4,6]$ $Tag_3:A[1,2]$ $Tag_3:B[5,6]$
$Tag_3:A[7,9]$ $Tag_4:C[1,3]$ $Tag_4:B[4,5]$
$Tag_5:C[1,3]$ $Tag_5:F[4,6]$ $Tag_5:D[7,9]$
$Tag_6:C[1,3]$ $Tag_6:F[4,6]$ $Tag_6:E[11,14]$
$Tag_7:D[5,7]$ $Tag_8:B[8,9]$ $Tag_8:C[13,14]$

(a) Spatiotemporal data with tag

$Tag_1:A[1,2,1]\rightarrow B[4,6,2]$
$Tag_2:A[1,2,1]\rightarrow B[4,6,2]$
$Tag_3:A[1,2,1]\rightarrow B[5,6,2]\rightarrow A[7,9,3]$
$Tag_4:C[1,3,1]\rightarrow B[4,5,2]$
$Tag_5:C[1,3,1]\rightarrow F[4,6,2]\rightarrow D[7,9,3]$
$Tag_6:C[1,3,1]\rightarrow F[4,6,2]\rightarrow E[11,14,3]$
$Tag_7:D[5,7,1]$
$Tag_8:B[8,9,1]\rightarrow C[13,14,2]$

(b) Path information with tag

Fig. 2 RFID data after processing

图 2 加工后的 RFID 数据

我们将包含时间和位置信息的每个路径结点 $Loc(ST,ET,l)$称为时空数据结点,并以此为编码对象进行编码,以支持路径追溯查询.本文采用的编码框架如图 3 所示:

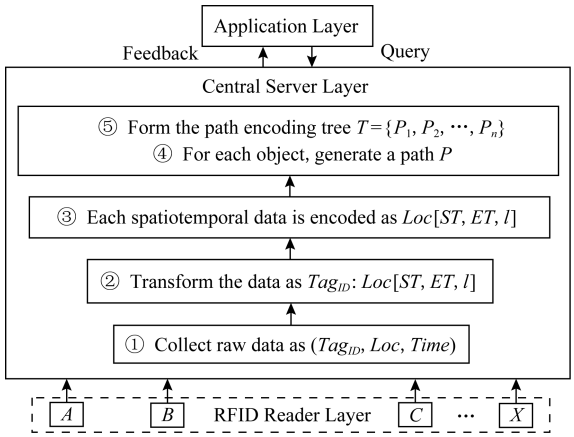

Fig. 3 Framework of RFID path coding

图 3 RFID 路径编码框架

首先,各 RFID 读写器将原始数据上传给中央服务器.然后,中央服务器对数据进行转化处理,并对每条时空信息进行编码.最后,将每个物品对象,按经过的位置顺序生成各自路径信息,且将每条路径中结点的编码根据时空顺序形成路径树,以表的形式存入数据库供应用层查询.

2 偏增向量编码策略

本节将在文献[18]所提出的向量编码上研究一种能支持追溯查询且溢出速率较低的偏增向量编码策略.

2.1 基本向量编码

偏增向量编码仍采用平面直角坐标系中的第一象限中的向量进行编码.该方法跳过基本向量编码中的区间转化的过程,直接根据结点进入顺序按更新规则得到向量编码.

对于向量编码,有如下定义及定理[18]:

定义 1. 一个向量 $\boldsymbol{V}$ 可表示为平面直角坐标系第一象限中的 1 个坐标(x,y),其中 x,y 都为整数,如图 4(a)所示.

定义 2. 对于任意 2 个向量 $\boldsymbol{V}_1(x_1,y_1)$,$\boldsymbol{V}_2(x_2,y_2)$和 1 个标量 r,则有:

$$\boldsymbol{V}_1+\boldsymbol{V}_2=(x_1+x_2,y_1+y_2), \tag{1}$$

$$r\times\boldsymbol{V}_1=(r\times x_1,r\times y_1). \tag{2}$$

定义 3. 设 $G(\boldsymbol{V})$为向量 $\boldsymbol{V}(x,y)$的倾斜度,则可表示为 $G(\boldsymbol{V})=y/x$,即图 4(a)中的 $\tan\theta$.

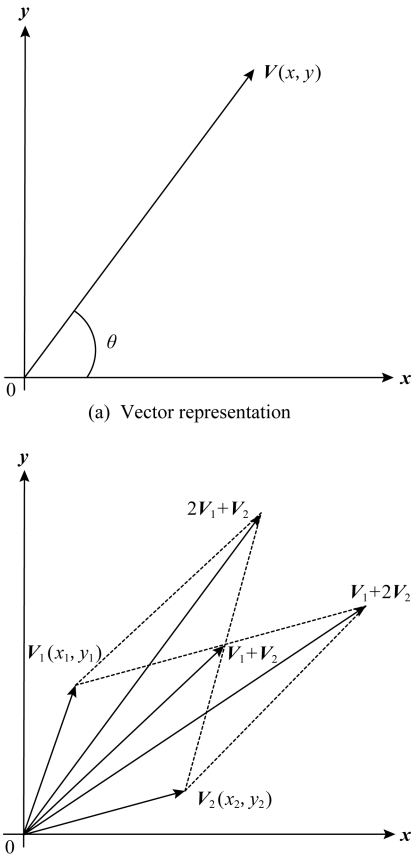

Fig. 4 Vector diagram

图 4 向量示意图

定义 4. 设 $GS(\boldsymbol{V})$为向量 $\boldsymbol{V}(x,y)$的粒度,则可表示为 $GS(\boldsymbol{V})=x+y$.

定理 1. 给定 2 个向量 $\boldsymbol{V}_1(x_1,y_1)$和 $\boldsymbol{V}_2(x_2,y_2)$,若 $y_1x_2>x_1y_2$,则 $G(\boldsymbol{V}_1)>G(\boldsymbol{V}_2)$.

定理 2. 给定 2 个向量 $\boldsymbol{V}_1$ 和 $\boldsymbol{V}_2$,若 $\boldsymbol{V}_3=\boldsymbol{V}_1+\boldsymbol{V}_2$ 且 $G(\boldsymbol{V}_1)>G(\boldsymbol{V}_2)$,则 $G(\boldsymbol{V}_1)>G(\boldsymbol{V}_3)>G(\boldsymbol{V}_2)$.

定理 3. 给定 2 个向量 $\boldsymbol{V}_1$ 和 $\boldsymbol{V}_2$,则一定存在无数个向量,其倾斜度都在 $G(\boldsymbol{V}_1)$和 $G(\boldsymbol{V}_2)$之间.

根据定理 2 和定理 3,如图 4(b),向量 $2\boldsymbol{V}_1+\boldsymbol{V}_2$,$\boldsymbol{V}_1+\boldsymbol{V}_2$,$\boldsymbol{V}_1+2\boldsymbol{V}_2$ 都会在向量 $\boldsymbol{V}_1$ 和 $\boldsymbol{V}_2$ 之间.

因此,可以根据结点位置不同,选择($\boldsymbol{V}_1+\boldsymbol{V}_2$,$\boldsymbol{V}_1+2\boldsymbol{V}_2$)或($2\boldsymbol{V}_1+\boldsymbol{V}_2$,$\boldsymbol{V}_1+\boldsymbol{V}_2$)进行编码.因在 $\boldsymbol{V}_1$ 和 $\boldsymbol{V}_2$ 之间可以反复进行向量加法运算,得到无数个不重复向量,故结点更新变得容易.

2.2 偏增向量编码原理

基于上述原理,给出偏增向量编码(OAVC)策略的基本规则:

规则 1. 对于 1 个 RFID 时空数据结点 n:$Loc(ST,ET,l)$,我们用起始向量 $\boldsymbol{V}^n_{\text{start}}=(x_1,y_1)$,终止向量 $\boldsymbol{V}^n_{\text{end}}=(x_2,y_2)$以及层级 l,共同组成 1 个三元组($\boldsymbol{V}^n_{\text{start}}$,$\boldsymbol{V}^n_{\text{end}}$,$l$)进行编码.其中 x 和 y 分别是二维空间第一象限中的横坐标和纵坐标,由规则 2~4 得到.

规则 2. 路径编码树的根定义为虚结点,其码值固定为((1,0)(0,1),0).

编码结点按其更新位置分为 4 类:无兄弟结点、仅有左兄弟结点、仅有右兄弟结点、既有左兄弟结点又有右兄弟结点.根据其位置不同,确定合适的偏置向量进行编码.

规则 3. 设编码结点 n,其层级为 l,p 是其亲代结点,cps 是最邻近 n 的左兄弟结点,cfs 是最邻近 n 的右兄弟结点,$\boldsymbol{v}_1$ 和 $\boldsymbol{v}_2$ 为 n 的偏置向量,则有 4 种情形:

1) 若 n 无兄弟结点,则 $\boldsymbol{v}_1=\boldsymbol{V}^p_{\text{start}}$,$\boldsymbol{v}_2=\boldsymbol{V}^p_{\text{end}}$;

2) 若 n 仅有左兄弟结点,则 $\boldsymbol{v}_1=\boldsymbol{V}^{cps}_{\text{end}}$,$\boldsymbol{v}_2=\boldsymbol{V}^p_{\text{end}}$;

3) 若 n 仅有右兄弟结点,则 $\boldsymbol{v}_1=\boldsymbol{V}^p_{\text{start}}$,$\boldsymbol{v}_2=\boldsymbol{V}^{cfs}_{\text{start}}$;

4) 若 n 既有左兄弟结点又有右兄弟结点,则 $\boldsymbol{v}_1=\boldsymbol{V}^{cps}_{\text{end}}$,$\boldsymbol{v}_2=\boldsymbol{V}^{cfs}_{\text{start}}$.

为尽可能减小码值,应考虑 $\boldsymbol{v}_1$,$\boldsymbol{v}_2$ 的粒度大小以决定最后的码值,因此有规则 4:

规则 4. 若 $GS(\boldsymbol{v}_1)>GS(\boldsymbol{v}_2)$,则其编码为($(\boldsymbol{v}_1+\boldsymbol{v}_2,\boldsymbol{v}_1+2\boldsymbol{v}_2),l$);反之,则为($(2\boldsymbol{v}_1+\boldsymbol{v}_2,\boldsymbol{v}_1+\boldsymbol{v}_2),l$).

为了判别结点之间子亲代关系,有定理 4.

定理 4. 子亲代判定定理.若有 2 结点 p 与 n,满足关系:$G(\boldsymbol{V}^p_{\text{start}})<G(\boldsymbol{V}^n_{\text{start}})<G(\boldsymbol{V}^n_{\text{end}})<G(\boldsymbol{V}^p_{\text{end}})$,则 p 是 n 的祖先,n 是 p 的后代;进一步,若满足 $p.level-n.level=1$,则 p 是 n 的亲代,n 是 p 的子代.

在编码时,若数据全部按时间先后顺序,每个结点的 ST 值从小到大进入服务器,同层级自左向右编码,则只需进行规则 3 的情形 1)2)编码.然而,各地阅读器由于距离与速度等原因,到达中央服务器的时间会不一样.因此,当到达顺序发生错乱时,可用规则 3 的情形 3)4)调整顺序.这样做的好处在于:一方面能做到任意时刻任意位置进行实时更新;另一方面能保证整棵编码树完全按时间先后顺序排列,在进行基于时间的查询时提高查询效率.

基于上述原理,可构造出如图 5 所示的偏增向量编码树,图 5 中结点 X,Y 为规则 3 的情形 3)4)实时更新样例.当更新结点 X 时,有 $\boldsymbol{v}_1=(1,3)$,$\boldsymbol{v}_2=(3,7)$,$GS(\boldsymbol{v}_1)<GS(\boldsymbol{v}_2)$,因此 $\boldsymbol{V}^X_{\text{start}}=(1\times2+3,3\times2+7)=(5,13)$,$\boldsymbol{V}^X_{\text{end}}=(1+3,3+7)=(4,10)$,$X$ 码值为(5,13)(4,10),2.同理可得,结点 Y 码值为(7,18),(5,13),2.

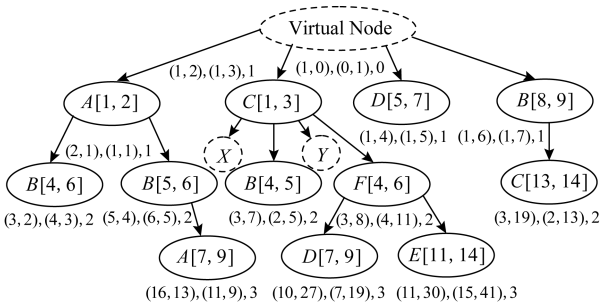

Fig. 5 Coding tree of offset addition vectors

图 5 偏增向量编码树

通过编码树的生成过程可以知道,偏增向量编码的本质是在一定角度内无限地划分角度来表示结点,其思想与区间编码划分区间来表示结点十分类似.但是,区间编码通常是整数区间,需等待数据到达后再进行编码,否则会因整数不可再分而出现大面积更新.而偏增向量编码结点更新时不会影响其他结点,可做到随到随编,满足实时更新及查询需求.

同时,偏增向量编码的对象是时空数据结点,环路问题也能得以解决,因为环路的本质是物品经过同一位置多次.我们可以用编码中的时间信息区分同一位置的每一次经过.例如图 5 中路径 $A[1,2]\rightarrow B[5,6]\rightarrow A[7,9]$,$A[1,2]$与 $A[7,9]$表示 2 个不同结点,因为它们经过 A 的时间不同.

然而,偏增向量编码是通过向量相加的形式得到新结点的编码,码值增长较快.对于海量 RFID 数据,可能会导致码值溢出,故需进行优化.

2.3 向量码值优化

在介绍编码原理时已经提到,为减缓码值溢出,应使偏置向量 $\boldsymbol{v}$ 的粒度 $GS(\boldsymbol{v})$尽可能小.然而,$\boldsymbol{v}$ 的取值是通过结点 cps,cfs,p 的起始和终止向量确定的,又因为大部分数据是按时间先后顺序进入,所以在对结点进行编码时,同一层级中大多数结点编码顺序是自左到右,即欲编码的结点 n 仅有左兄弟而无右兄弟,因此根据规则 3,有 $\boldsymbol{v}_1=\boldsymbol{V}^{cps}_{\text{end}}$,$\boldsymbol{v}_2=\boldsymbol{V}^{p}_{\text{end}}$;欲减小 $\boldsymbol{v}_1$,$\boldsymbol{v}_2$ 的粒度,可从 2 种情形考虑:

1) 在图 6(a)中,n 的码值依赖于其最邻近 n 的左兄弟结点的终止向量 $\boldsymbol{V}^{cps}_{\text{end}}$,而 n 的最邻近右兄弟结点 cfs 的码值依赖于 $\boldsymbol{V}^{n}_{\text{end}}$,可见这种依赖关系是层层递进,且同一层级下最左边结点的码值对其所有亲兄弟的码值影响最大.因此,可以考虑单独分析最左边结点 m,在生成 m 的编码时让粒度较小的向量赋给 $\boldsymbol{V}^{m}_{\text{end}}$.

2) 若规定同一父结点下任意左兄弟向量的斜率都大于右兄弟的斜率时,可得图 6(b).此时,$\boldsymbol{v}_1=\boldsymbol{V}^{p}_{\text{start}}$,$\boldsymbol{v}_2=\boldsymbol{V}^{cps}_{\text{start}}$.由于 cps 不一定存在,因此只考虑 n 父结点 p,当 $GS(\boldsymbol{V}^{p}_{\text{start}})<GS(\boldsymbol{V}^{p}_{\text{end}})$时,显然 $\boldsymbol{v}_1$ 取 $\boldsymbol{V}^{p}_{\text{start}}$时码值粒度更小,所以在编码算法中,应先比较 $\boldsymbol{V}^{p}_{\text{start}}$与 $\boldsymbol{V}^{p}_{\text{end}}$,将粒度较小者作为偏置向量.

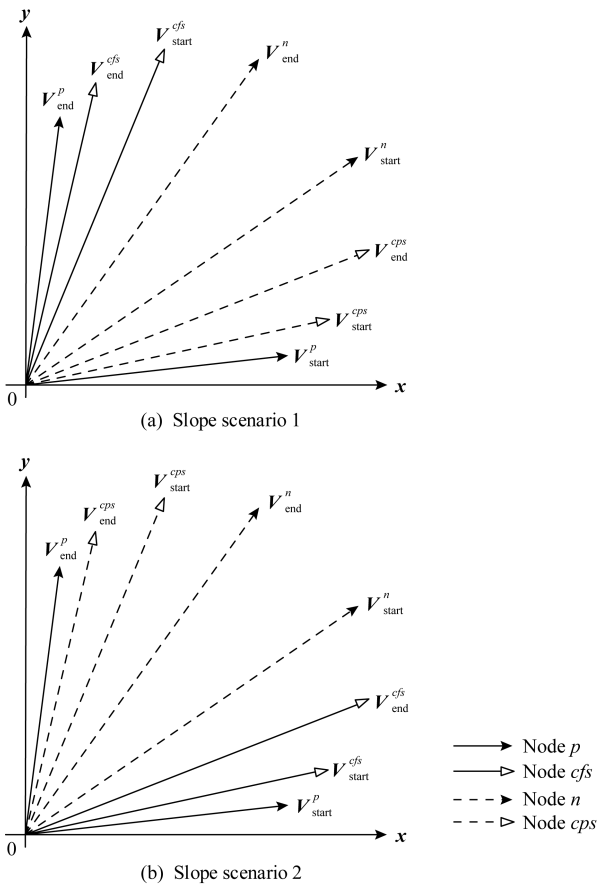

Fig. 6 The coded graph in different slope order

图 6 不同斜率顺序排列下编码示意图

根据上述优化思想,可得优化偏增向量编码(optimized offset addition vector coding, OOAVC)算法如算法 1 所示.

算法 1. 优化偏增向量编码算法.

输入:结点 n 其父结点、最近邻左、右兄弟结点的编码 p,cps,cfs;

输出:结点 n 的编码 $code$.

① if $GS(\boldsymbol{V}^{p}_{\text{start}})\geqslant GS(\boldsymbol{V}^{p}_{\text{end}})$ /* 情形 2 */

② $flag=0$; /* 斜率如图 6(a)所示排列 */

③ if n 有左兄弟结点

④ $\boldsymbol{v}_1=\boldsymbol{V}^{cps}_{\text{end}}$;

⑤ else

⑥ $\boldsymbol{v}_1=\boldsymbol{V}^{p}_{\text{start}}$;

⑦ end if

```
⑧   if n 有右兄弟结点
⑨     v2 = V^cfs_start;
⑩   else
⑪     v2 = V^p_end;
⑫   end if
⑬ else /* GS(V^p_start) < GS(V^p_end) */
⑭   flag = 1; /* 斜率如图 6(b)所示排列 */
⑮   if n 有左兄弟结点
⑯     v1 = V^cps_start;
⑰   else
⑱     v1 = V^p_end;
⑲   end if
⑳   if n 有右兄弟结点
㉑     v2 = V^cfs_end;
㉒   else
㉓     v2 = V^p_start;
㉔   end if
㉕ end if
㉖ if GS(v1) ≥ GS(v2) /* 情形 1 */
㉗   if n 不为最左结点
㉘     if (flag) code = (v1 + 2v2, v1 + v2, l);
㉙     else code = (v1 + v2, v1 + 2v2, l);
㉚     end if
㉛   else
㉜     if (flag) code = (v1 + v2, 2v1 + v2, l);
㉝     else code = (2v1 + v2, v1 + v2, l);
㉞     end if
㉟   end if
㊱ else /* GS(v1) ≤ GS(v2) */
㊲   if n 不为最左结点
㊳     if (flag) code = (v1 + v2, 2v1 + v2, l);
㊴     else code = (2v1 + v2, v1 + v2, l);
㊵     end if
㊶   else
㊷     if (flag) code = (v1 + 2v2, v1 + v2, l);
㊸     else code = (v1 + v2, v1 + 2v2, l);
㊹     end if
㊺   end if
㊻ end if
㊼ return code.
```

根据优化后的算法可以得到新的编码树如图 7 所示,图 7 中结点 X 的码值为(4,11),(5,14),2,结点 Y 的码值为(5,12),(7,17),2.

从结果上对比可以看出,OOAVC 相比于 OAVC,亲兄弟结点中首个结点码值会略微偏大,但其后所有兄弟结点的码值都会偏小.

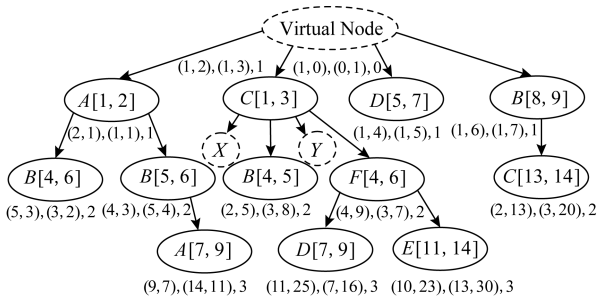

Fig. 7 OOAVC coding tree

图 7 OOAVC 编码树

2.4 优化偏增向量编码亲子代关系证明

要证满足亲子代关系,即证新编码结点的编码满足定理 4,即证新生成的结点 n,其编码值($\boldsymbol{V}^n_{\text{start}}$, $\boldsymbol{V}^n_{\text{end}}$)满足关系:$G(\boldsymbol{V}^p_{\text{start}}) < G(\boldsymbol{V}^n_{\text{start}}) < G(\boldsymbol{V}^n_{\text{end}}) < G(\boldsymbol{V}^p_{\text{end}})$.

证明.

1) 当 $GS(\boldsymbol{V}^p_{\text{start}}) \geqslant GS(\boldsymbol{V}^p_{\text{end}})$时,如图 6(a)所示,$\boldsymbol{v}_1$ 的取值为 $\boldsymbol{V}^{cps}_{\text{end}}$ 或 $\boldsymbol{V}^p_{\text{start}}$,$\boldsymbol{v}_2$ 的取值为 $\boldsymbol{V}^{cfs}_{\text{start}}$ 或 $\boldsymbol{V}^p_{\text{end}}$,有:

$$G(\boldsymbol{V}^p_{\text{start}}) < G(\boldsymbol{V}^{cps}_{\text{start}}) < G(\boldsymbol{V}^{cps}_{\text{end}}) < G(\boldsymbol{V}^{cfs}_{\text{start}}) < G(\boldsymbol{V}^{cfs}_{\text{end}}) < G(\boldsymbol{V}^p_{\text{end}}).$$

所以 $G(\boldsymbol{V}^p_{\text{start}}) \leqslant G(\boldsymbol{v}_1), G(\boldsymbol{v}_2) \leqslant G(\boldsymbol{V}^p_{\text{end}})$.

2) 当 $GS(\boldsymbol{V}^p_{\text{start}}) < GS(\boldsymbol{V}^p_{\text{end}})$时,如图 6(b)所示,$\boldsymbol{v}_1$ 的取值为 $\boldsymbol{V}^{cfs}_{\text{end}}$ 或 $\boldsymbol{V}^p_{\text{start}}$,$\boldsymbol{v}_2$ 的取值为 $\boldsymbol{V}^{cps}_{\text{start}}$ 或 $\boldsymbol{V}^p_{\text{end}}$,有:$G(\boldsymbol{V}^p_{\text{start}}) < G(\boldsymbol{V}^{cfs}_{\text{start}}) < G(\boldsymbol{V}^{cfs}_{\text{end}}) < G(\boldsymbol{V}^{cps}_{\text{start}}) < G(\boldsymbol{V}^{cps}_{\text{end}}) < G(\boldsymbol{V}^p_{\text{end}})$.

所以 $G(\boldsymbol{V}^p_{\text{start}}) \leqslant G(\boldsymbol{v}_1), G(\boldsymbol{v}_2) \leqslant G(\boldsymbol{V}^p_{\text{end}})$.

由 1)2)可得:

$$G(\boldsymbol{V}^p_{\text{start}}) \leqslant G(\boldsymbol{v}_1), G(\boldsymbol{v}_2) \leqslant G(\boldsymbol{V}^p_{\text{end}}).$$

由算法 1 可知 $\boldsymbol{V}^n_{\text{start}} = 2\boldsymbol{v}_1 + \boldsymbol{v}_2, \boldsymbol{V}^n_{\text{end}} = \boldsymbol{v}_1 + \boldsymbol{v}_2$ 或 $\boldsymbol{V}^n_{\text{start}} = \boldsymbol{v}_1 + \boldsymbol{v}_2, \boldsymbol{V}^n_{\text{end}} = \boldsymbol{v}_1 + 2\boldsymbol{v}_2$.

又由定理 2 得,$G(\boldsymbol{v}_1) < G(\boldsymbol{v}_1 + \boldsymbol{v}_2) < G(\boldsymbol{v}_2)$.

因 $2\boldsymbol{v}_1 + \boldsymbol{v}_2 = (\boldsymbol{v}_1 + \boldsymbol{v}_2) + \boldsymbol{v}_1$,得 $G(\boldsymbol{v}_1) < G(2\boldsymbol{v}_1 + \boldsymbol{v}_2) < G(\boldsymbol{v}_1 + \boldsymbol{v}_2)$,所以 $G(\boldsymbol{v}_1) < G(2\boldsymbol{v}_1 + \boldsymbol{v}_2) < G(\boldsymbol{v}_1 + \boldsymbol{v}_2) < G(\boldsymbol{v}_2)$.

因 $\boldsymbol{v}_1 + 2\boldsymbol{v}_2 = (\boldsymbol{v}_1 + \boldsymbol{v}_2) + \boldsymbol{v}_2$,得 $G(\boldsymbol{v}_1 + \boldsymbol{v}_2) < G(\boldsymbol{v}_1 + 2\boldsymbol{v}_2) < G(\boldsymbol{v}_2)$,所以 $G(\boldsymbol{v}_1) < G(\boldsymbol{v}_1 + \boldsymbol{v}_2) < G(\boldsymbol{v}_1 + 2\boldsymbol{v}_2) < G(\boldsymbol{v}_2)$.

又因 $G(\boldsymbol{V}^p_{\text{start}}) \leqslant G(\boldsymbol{v}_1), G(\boldsymbol{v}_2) \leqslant G(\boldsymbol{V}^p_{\text{end}})$,所以 $G(\boldsymbol{V}^p_{\text{start}}) < G(\boldsymbol{V}^n_{\text{start}}) < G(\boldsymbol{V}^n_{\text{end}}) < G(\boldsymbol{V}^p_{\text{end}})$.

综上,优化偏增向量编码仍满足亲子关系.

证毕.

3 路径追溯查询

3.1 存储模式

为了便于查询,我们将路径树转化 3 个数据库中的表进行存储.

1) 标签表(tag table).存储标签号(Tag_{ID})及路径编号($Path_{ID}$),以方便查询物品所在路径,如表 1 所示:

Table 1 Tag Table

表 1 标签表

Tag_{ID}	$Path_{ID}$	Tag_{ID}	$Path_{ID}$
1	1	5	4
2	1	6	5
3	2	7	6
4	3		

2) 路径表(path table).存储路径编号及该路径当前末尾结点的编码((VS_x,VS_y),(VE_x,VE_y)),用于根据子亲代判断定理确定整条路径,如表 2 所示.

3) 时间表(time table).存储编码值以及结点时空信息,其中((VS_x,VS_y),(VE_x,VE_y),$Level$)表示编码,$Loc(ST,ET)$表示时空信息,如表 3 所示.

为提高查询和更新效率,在进行路径查询或更新时,我们将从表 3 提取数据构建编码树到内存中,再结合表 1~2 信息(也存放在内存)进行查询.

Table 2 Path Table

表 2 路径表

$Path_{ID}$	VS_x	VS_y	VE_x	VE_y
1	5	3	3	2
2	9	7	14	11
3	2	5	3	8
4	11	25	7	16
5	10	23	13	30
6	1	4	1	5
7	2	13	3	20

Table 3 Time Table

表 3 时间表

VS_x	VS_y	VE_x	VE_y	$Level$	Loc	ST	ET
2	1	1	1	1	A	1	2
1	2	1	3	1	C	1	3
5	3	3	2	2	B	4	6
2	5	3	8	2	B	4	5
4	9	3	7	2	F	4	6
1	4	1	5	1	D	5	7

3.2 追溯查询实现

供应链中的典型追溯查询如表 4 所示,大致可分为 3 类:追踪查询如 Q_1、面向路径查询 Q_2 和 Q_3、聚合查询 Q_4~Q_6,下面对其做具体说明.

Table 4 Query Classification Table

表 4 查询分类表

Query	Description	Statement
Q_1	Tracing query	$\langle // Tag_{ID} \rangle$
Q_2	Path-oriented query without time	$\langle // Loc \rangle$
Q_3	Path-oriented query with time	$\langle // Loc, ST, ET \rangle$
Q_4	Query for the label that first enters a location	$\langle \min(ST) // Loc \rangle$
Q_5	The average time taken for a query to pass through two locations together	$\langle avg(ET_{Loc_2}-ST_{Loc_1}), // Loc_1 // Loc_2 \rangle$
Q_6	Query for the number of tags that pass through two locations together	$\langle count(), // Loc_1 // Loc_2 \rangle$

1) 追踪查询

查询某标签的历史路径,包含访问过的每个位置及进出时间.首先,在表 1 中找到对应的 $Path_{ID}$,通过 $Path_{ID}$ 确定末尾结点的码值,然后从根结点出发,根据子亲代判定定理查出整条路径,详见算法 2.从算法 2 中我们可以看出,追踪查询效率的高低关键在于子亲代判断速度的快慢.

算法 2. 追踪查询.

输入:标签号 Tag_{ID}、编码树的根结点 $root$;

输出:Tag_{ID} 所经过的路径 P.

① 在标签表中找到 Tag_{ID} 对应的 $Path_{ID}$;

② 在路径表中找到 $Path_{ID}$ 对应的编码 $code$;

③ for ($i=0$;$root.child[i]$的编码!=$code$;
$i++$)

④ if $code$ 与 $root.child[i]$满足子亲代判定

⑤ 将 $root.child[i]$存入 P;

⑥ $root=root.child[i]$;

⑦ end if

⑧ end for
⑨ if $P \neq \varnothing$
⑩ 将 *code* 代表的结点存入 P;
⑪ end if
⑫ 返回 P.

2) 不含时间的面向路径查询

查询访问过某位置的标签集合.此查询需先确定位置信息匹配的结点,再根据结点找到路径和标签号,详见算法 3.

算法 3. 面向路径查询(不含时间).

输入:位置 *loc*、编码树的根结点 *root*;

输出:查询到的标签集合 T.

① 从 *root* 遍历树:
② if 结点曾访问过 *loc*
③ 将结点的编码存进 *codelist* 中;
④ end if
⑤ 遍历路径表,找到 $Path_{ID}$ 对应的 *code*:
⑥ if *code* 在 *codelist* 里
⑦ 将对应的 $Path_{ID}$ 存进 *Plist* 中;
⑧ end if
⑨ 遍历标签表,对 Tag_{ID} 对应的 $Path_{ID}$:
⑩ if $Path_{ID}$ 在 *Plist* 里:
⑪ 将对应的 Tag_{ID} 存进 T 里;
⑫ end if
⑬ 返回 T.

从算法 3 可知,面向路径的查询牵涉到多张表的完全遍历,因此较于追踪查询开销较大.其次,面向路径的查询效率同样与子亲代关系判断速度有关,且因查询的只是位置信息,所以也与编码信息有关.

3) 含时间的面向路径查询

与算法 3 类似,在输入时增加进入时间和离开时间约束,在算法 3 的行②增加时间约束即可.因增加了约束,查询时比较次数会提高,开销增大.同时因查询的不是单一信息,而是时空信息,单独编码位置的方法会不适用.

4) 聚合查询

聚合查询是比前 3 类查询更为复杂的查询,其查询的条件比较多,且通常需要在查询结果的基础上添加聚合函数.

算法 4. 聚合查询.

输入:编码树的根结点 *root*、查询条件 *condi*、聚合函数 *aggfunc*;

输出:最后的查询结果 Q.

① 从 *root* 遍历树:
② if 结点满足查询条件 *condi*
③ 搜索相关表,查询结点对应的记录;
④ 将记录存进 *Qlist* 中;
⑤ end if
⑥ $Q = aggfunc(Qlist)$;
⑦ 返回 Q.

4 编码实验与结果分析

4.1 实验目的

本实验将从 2 个方面进行实验.

1) 将所提出的偏增向量编码及优化策略 OAVC, OOAVC 与 3 种近年来 RFID 路径编码方法对时空数据进行编码,比较查询开销、更新开销、初始编码时间开销、最大编码值等.选用的对比编码策略为:

① 区间编码(region coding, RC).该编码利用先序遍历的方法,给每一个结点分配 2 个值,从根结点出发编码第 1 个值,编码孩子结点后回到孩子结点编码第 2 个值.2 个码值如同区间的左右,若有结点的码值在区间范围内,则为其孩子结点.

② 素数编码(prime coding, PC).该编码给每个结点分配唯一的素数,利用素数积因式分解唯一性进行子亲代判断.

③ 复合编码(composite coding, CC).该编码利用区间编码对时空数据结点进行编码,同时针对位置信息单独再用素数编码建立不含时间的位置路径树.对不同查询,灵活运用 2 棵树信息进行查询.

2) 比较 OAVC 和 OOAVC 的最大编码值,分析优化效果.

4.2 数据集

本实验的数据集通过仿真实验产生.我们构造一棵深度为 M 的满 N 叉树,将时空数据结点作为树的结点,并按一定规律随机赋值,在位置信息上允许路径有重复的位置出现,即位置信息有环路.其中:M 表示树的层数(虚结点的孩子结点为第 1 层),其含义为路径长度的最大值,用来控制树的深度;N 代表每个结点的孩子数,以控制树的宽度.

为分析树的深度、宽度对不同编码的影响,我们分别生成数据集 $D_1 \sim D_6$,如表 5~6 所示.其中,数据集 $D_1 \sim D_3$ 是固定 $N=5$,只增加树的深度(即 3~5)生成;数据集 $D_4 \sim D_6$ 是固定 $M=4$,只增加树的宽度(即 6~8)生成.

Table 5 Dataset Increases by the Depth of the Tree
表 5 数据集深度递增

Dataset	M	Number of Nodes	Number of Records	N
D_1	3	156	11 189	5
D_2	4	781	34 938	5
D_3	5	3 906	166 551	5

Table 6 Dataset Increases by the Width of the Tree
表 6 数据集宽度递增

Dataset	M	Number of Nodes	Number of Records	N
D_4	4	15 55	52 818	6
D_5	4	2 801	105 931	7
D_6	4	4 681	152 091	8

4.3 结果分析

1) 查询开销

我们对表 4 中每种查询随机生成查询条件,在数据集 $D_1 \sim D_6$ 中分别重复实验,计算平均时间,作为查询开销的评价指标.

图 8 为各种编码策略在数据集 $D_1 \sim D_6$ 中各类查询的时耗图,其中横坐标为数据集类别,纵坐标为平均查询时耗.

对于 Q_1 而言,OOAVC 和 OAVC 在判断子亲代关系时,由于一个编码有 4 个值要比较,因此会略慢,但与其他编码相差不大.

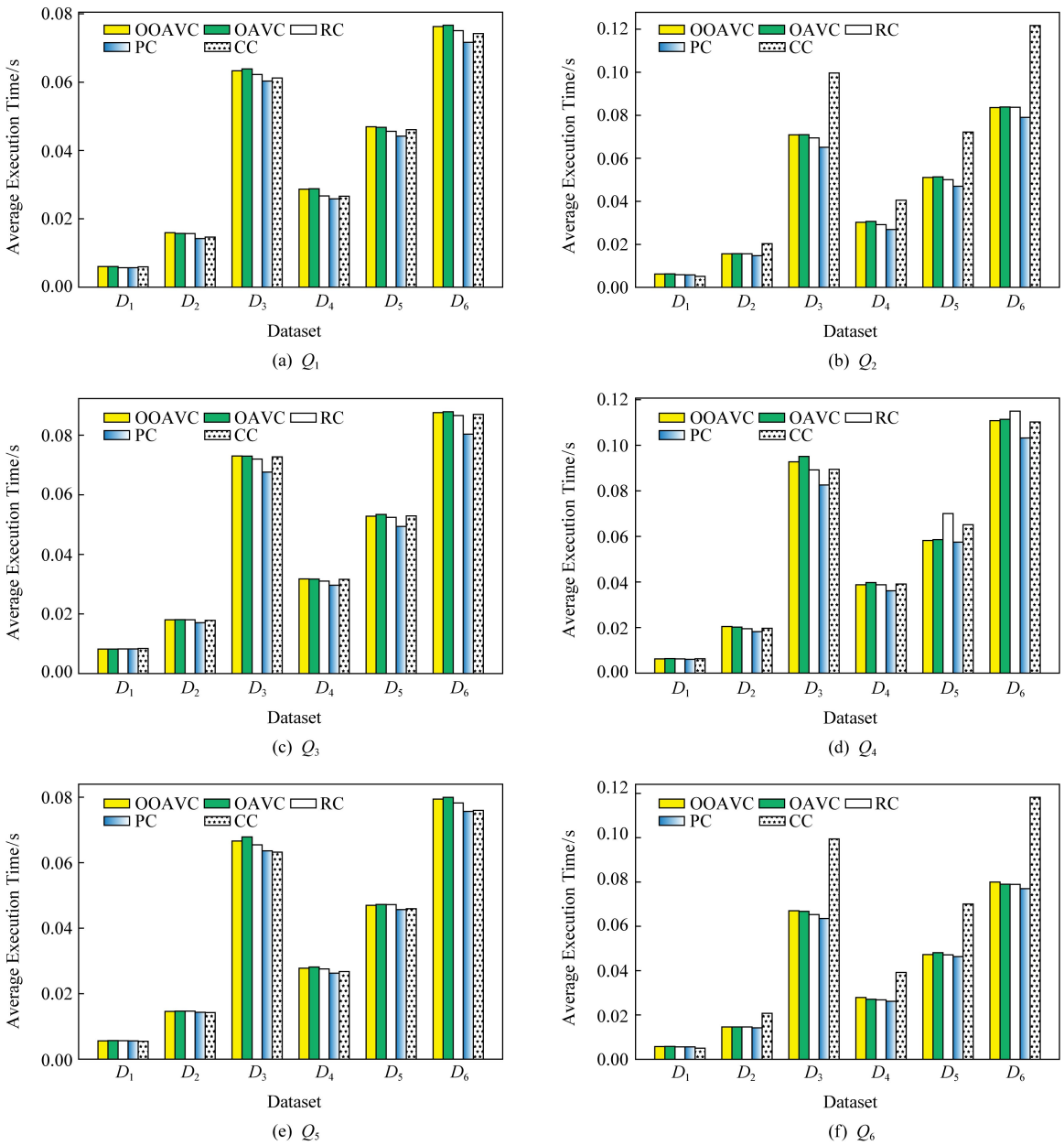

Fig. 8 Query overhead on the $D_1 \sim D_6$

图 8 不同编码在数据集 $D_1 \sim D_6$ 上的查询开销

对于 Q_2,除复合编码外,OOAVC 和 OAVC 与其他编码大致相同.而对于复合编码,尽管其额外编码了位置信息,相当于降低了搜索范围,本应在此类查询上占据优势,但本实验数据集在位置上存在环路,一个位置可能对应多个编码,导致在解码时要遍历整张表,此外编码与位置对应的表存在外存中,因此整体上表现最差,这也体现出环路问题会使只考虑位置信息编码变得更为复杂.

对于 Q_3 而言,查询包含时间和空间信息,此时所有编码的编码对象都为时空数据结点,开销主要影响因素为子亲代判断速度,OOAVC 和 OAVC 尽管子亲代判断过程稍复杂,但查询速度与其他编码相差不大.可以发现,Q_3 整体开销略高于 Q_2,这是因为每次查询要多比较时间信息,从而开销增大.

对于聚合查询 Q_4,Q_5 涉及到时间信息,因此各编码整体上与 Q_3 结果类似,而 Q_6 只涉及空间信息,结果与 Q_2 类似.

综合各类查询上来看,素数编码的效率最高,而 OOAVC 与 OAVC 尽管不占优势,但差距并不大,能支持和满足大多数查询.

2) 更新开销

同样地,我们随机选取树中的某个结点,插入 1 个叶子结点作为其孩子并进行编码,在数据集 $D_1 \sim D_6$ 中分别重复操作,计算平均更新开销作为评价指标,结果如图 9 所示:

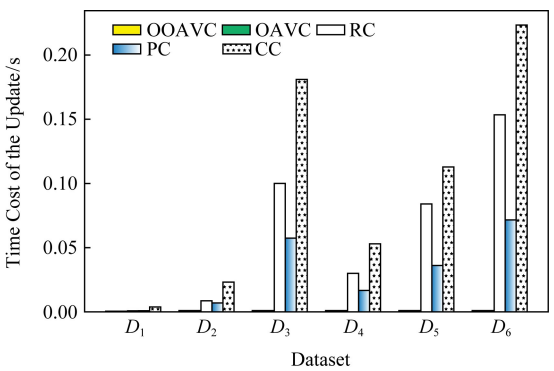

Fig. 9 Comparison of update cost of each encoding on $D_1 \sim D_6$

图 9 各编码在数据集 $D_1 \sim D_6$ 上更新开销比较

由图 9 可看到,相比于区间、素数、复合编码,所提出的 2 种偏增向量编码更新开销几乎为零.区间编码是因为编码码值都连续,没有空位留给新插入的结点,1 个结点的插入会导致大面积结点需要重新编码.素数编码是因为当编码到一定大小时,搜索新的素数也需要花费时间,因此效率也偏低.复合编码在查询时继承了 2 个编码缺陷,更新开销最大.

3) 初始编码时间开销

初始编码时间开销是指给定一棵树结构,对其每个结点进行编码所需要的时间.同样,我们在数据集 $D_1 \sim D_6$ 中统计编码平均时间作为评价指标,得到图 10 所示:

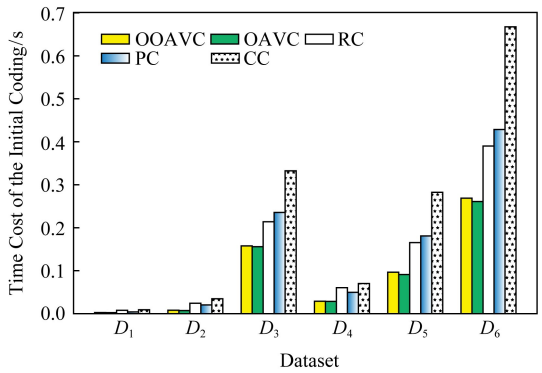

Fig. 10 Comparison of initial cost of each encoding on $D_1 \sim D_6$

图 10 各编码在数据集 $D_1 \sim D_6$ 上初始编码开销比较

可以发现,OOAVC 与 OAVC 的初始编码开销均较大程度优于其余策略.在数据集较小的情况如 D_1,D_2,D_4,区间编码的开销是要大于素数编码的,但当数据集变大时,搜索新素数导致素数编码开销更大.同时,复合编码由于有 2 棵树需要编码,开销最大.

4) 最大编码值

最大编码值即为路径编码的最大值.对于 OOAVC,OAVC,RC 而言,路径编码用路径中叶子结点的编码来表示,最大编码值即为结点编码的最大值;对于 PC,CC 而言,路径编码由根结点到最后一个叶子结点编码的乘积得到.

表 2 中存储了每条路径的路径编码,可直接从表 2 中找到最大的码值.由于不同编码的码值差距过大,我们用编码值的 lb 对数表示,结果如图 11 所示.

由图 11 我们可以看到,素数和复合编码的最大编码值已经远远超过了其他编码的码值.理论上,复合编码的最大码值应小于等于素数编码,这取决于位置信息相同的路径多少,而从图 11 中看出,两者最大编码值相差并不大,说明都存在严重的溢出问题.

而通过观察对比,在更深的树如数据集 D_3 上,最大码值会比更宽的树如数据集 D_5 上更大,这也与素数的乘积增长极快相吻合.事实上,再在数据集 D_3 上继续提高深度或宽度,都已经会导致超出所定义的数据结构,导致溢出错误.

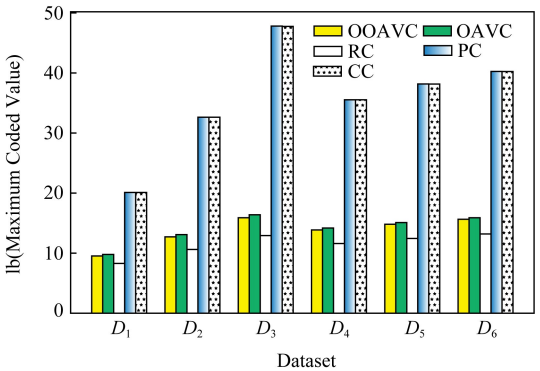

Fig. 11 Maximum coded value of each encoding in dataset $D_1 \sim D_6$

图 11 各编码在数据集 $D_1 \sim D_6$ 上最大编码值

5) 码值优化比较

为了更清晰地观测不同编码的码值,我们将各类编码具体的码值表给出,如表 7 所示.我们可以发现,尽管同素数相比,向量编码的码值较小,但与区间编码相比码值增长同样十分迅速,因此对 OAVC 的码值进行优化是十分有必要的,可以发现,OOAVC 码值得到了一定减少,而当数据集逐渐变大时,优化的效果也更明显.

同时,表 7 中素数编码与复合编码的最大编码值相差有限,说明溢出问题仍是复合编码的难点,而相较于复合编码的最大编码值,无论是 OOAVC 还是 OAVC 都要小得多.

综合实验结果,我们将各编码策略整体表现归纳如表 8 所示.

Table 7 Table of Maximum Encoding Values

表 7 最大编码值表

Dataset	OOAVC	OAVC	RC	PC	CC
D_1	738	881	311	1 127 401	1127401
D_2	6 723	8 721	1 561	6695634539	6695634539
D_3	61 245	86 329	7 811	246539959360519	239251598765003
D_4	14 883	18 601	3 109	49932921233	49894637923
D_5	28 899	35 113	5 601	310031752691	306314420219
D_6	51 075	60 705	9 361	1309515654203	1304635146851

Table 8 Comparison of Comprehensive Performance of Each Code

表 8 各编码综合性能比较

Evaluation	OOAVC	OAVC	RC	PC	CC
Q_1,Q_3,Q_4,Q_5	higher	higher	little difference	little difference	little difference
Q_2,Q_6	little difference	little difference	little difference	little difference	higher
Update	very low	very low	ordinary	higher	higher
Initialization	lower	lower	higher	higher	higher
Overflow	slower	slower	slower	faster	faster
Loop Support	good	good	good	ordinary	ordinary

可以看出,所提出的偏增向量编码能较好地满足各类追溯查询要求,具有编码速度较快、码值溢出较为缓慢、更新效率高且支持环路的特点,因此是一种效率较高且具有强鲁棒性的编码策略.

5 总结及展望

本文根据基于 RFID 供应链环境中标签对象路径追溯查询需求,提出了一种能支持环路、实时更新的偏增向量编码策略.同时,对该编码潜在的码值溢出问题提出了优化方法,并对其正确性进行了证明.在模拟数据集上完成了实验,结果表明:所提出的编码策略能基本满足大多数查询需求,且具有健壮、强鲁棒的特点,能较好地适应供应链复杂环境.

目前,已有编码策略大多仅在集中式环境下使用,下一步拟研究分布式供应链向量编码策略.

参 考 文 献

[1] Musa A, Dabo A A A. A review of RFID in supply chain management [J]. Global Journal of Flexible Systems Management, 2016, 17(2): 189-228

[2] Lee Y, Cho J. RFID-based sensing system for context information management using P2P network architecture [J]. Peer-to-Peer Networking and Applications, 2018, 11(6): 1197-1205

[3] Fahim A, Elbatt T, Mohamed A, et al. Towards extended bit tracking for scalable and robust RFID tag identification systems [J]. IEEE Access, 2018, 6: 27190-27204

[4] Qi Saiyu, Zheng Yuanqing, Chen Xiaofeng, et al. Double-edged sword: Incentivized verifiable product path query for RFID-Enabled supply chain [C] //Proc of the 37th IEEE Int Conf on Distributed Computing Systems. Piscataway, NJ: IEEE, 2017: 414-424

[5] Parada R, Meliàseguí J, Pous R. Anomaly detection using RFID-based information management in an IoT context [J]. Journal of Organizational and End User Computing, 2018, 30(3): 1-23

[6] Tian F. An agri-food supply chain traceability system for China based on RFID & blockchain technology [C] //Proc of the 13th Int Conf on Service Systems and Service Management. Piscataway, NJ: IEEE, 2016: 1-6

[7] Liao Guoqiong, Wan Qizhi, Jiang Jian, et al. A geometry vector encoding strategy for tracing containment relationship of RFID objects [J]. Journal of Chinese Computer Systems, 2014, 35(6): 1298-1303 (in Chinese)
(廖国琼, 万齐智, 蒋剑, 等. 支持 RFID 对象包含关系追溯的几何向量编码策略[J]. 小型微型计算机系统, 2014, 35(6): 1298-1303)

[8] Lee C H, Chung C W. Efficient storage scheme and query processing for supply chain management using RFID [C] // Proc of the 27th ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2008: 291-302

[9] Chen Xiong, Wang Xiaomei. The research of the supply chain path coding based on prime encoding technology [J]. Journal of Shanghai Normal University: Natural Science Edition, 2012, 41(5): 483-489 (in Chinese)
(陈雄, 王笑梅. 基于素数编码技术对供应链中路径编码的研究[J]. 上海师范大学学报: 自然科学版, 2012, 41(5): 483-489)

[10] Lee C H, Chung C W. RFID Data processing in supply chain management using a path encoding scheme [J]. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(5): 742-758

[11] Zhao Wen, Liu Xueyang, Ma Sen, et al. Formal specification of hierarchical region RFID code resolution service [C] //Proc of the 3rd Int Workshop on Intelligent Systems and Applications. Piscataway, NJ: IEEE, 2011: 1-10

[12] Li Lizhen. Research on RFID data warehousing model based on composite coding [D]. Guangzhou: South China University of Technology, 2010 (in Chinese)
(李力振. 基于复合编码的 RFID 数据仓储模型研究[D]. 广州: 华南理工大学, 2010)

[13] Lu Yao, Chen Lijun. An improved RFID data encoding scheme for cycling path problem [C] //Proc of the 4th IEEE Int Conf on Software Engineering and Service Science. Piscataway, NJ: IEEE, 2013: 611-615

[14] Fu Nan, Zheng Liying, Zhu Yuhang. Research on RFID path data coding compression storage technology [J]. Journal of Lanzhou Jiaotong University, 2013, 32(1): 82-87 (in Chinese)
(伏楠, 郑丽英, 朱宇航. RFID 路径数据编码压缩存储技术研究[J]. 兰州交通大学学报, 2013, 32(1): 82-87)

[15] Fan Hua, Wu Quanyuan, Lin Yisong, et al. A split-path schema-based RFID data storage model in supply chain management [J]. Sensors, 2013, 13(5): 5757-5776

[16] Chen Zhuxi, Sun Yan, Hu Kongfa, et al. Research on RFID data compression technology based on path coding [J]. Journal of Yangzhou University: Natural Science Edition, 2008, 11(2): 53-56 (in Chinese)
(陈竹西, 孙艳, 胡孔法, 等. 基于路径编码的 RFID 数据压缩技术研究[J]. 扬州大学学报: 自然科学版, 2008, 11(2): 53-56)

[17] Sheng Feng, Wang Yongli. RFID data compression algorithm based on path node coding [J]. Journal of Computer Research and Development, 2011, 48(Z2): 646-651 (in Chinese)
(盛丰, 王永利. 基于路径结点编码的 RFID 数据压缩算法[J]. 计算机研究与发展, 2011, 48(Z2): 646-651)

[18] Xu Liang, Bao Zhifeng, Ling Tok Wang. A dynamic labeling scheme using vectors [C] //Proc of the 18th Int Conf on Database and Expert Systems Applications. Berlin: Springer, 2007: 130-140

Liao Guoqiong, born in 1969. PhD. Professor and PhD supervisor at the School of Information Management, Jiangxi University of Finance and Economics. Senior member of CCF. His main research interests include databases, data mining and social networks.

Yang Lechuan, born in 1996. Master candidate at the School of Information Management, Jiangxi University of Finance and Economics. His main research interests include RFID data management and social networks.

Zhang Haiyan, born in 1998. Master candidate at the School of Information Management, Jiangxi University of Finance and Economics. Her main research interests include RFID data management.

Yang Xianpei, born in 1997. Bachelor at the School of Information Management, Jiangxi University of Finance and Economics. Her main research interests include RFID data management.