高技术通讯 2014 年 第24 卷 第6 期:597 ~601

doi:10.3772/j.issn.1002-0470.2014.06.007

TP-mine:基于分割簇的 RFID 轨迹数据增量聚类算法①

胡孔法②* 谢佳东* 赵 利**

(*南京中医药大学信息技术学院 南京 210023)

(**扬州大学信息工程学院 扬州 225009)

摘 要 针对具有增量特性的射频识别(RFID)轨迹数据的挖掘进行了研究,提出了轨迹聚类算法 TP-mine。该算法将每个新的轨迹简化成一个有向线性片段以便于找到轨迹子部分的聚类,使用分割簇来存储紧密的相似轨迹线性片段,比原始的轨迹占的空间要小。TP-mine 算法在整体聚类时对分割聚类所生成的分割簇进行操作,而不是对所有时间段的全部轨迹进行操作,可以高效地得出轨迹的聚类结果,从而实现对 RFID 技术所产生的轨迹数据进行有效挖掘,来发现移动对象潜在的移动趋势。

关键词 射频识别(RFID),分割簇,轨迹数据,增量聚类

0 引 言

射频识别(radio frequency identification,RFID)技术已应用于食品安全、药品跟踪、现代物流与供应链等领域,产生了海量的产品移动路径轨迹数据,如何分析和挖掘这些轨迹数据成为当前 RFID 技术推广应用的关键之一[1]。在移动对象轨迹数据挖掘方面已有相关研究,如 Kalnis 等人提出了移动簇方法[2],Benkert 等人提出了 flock 方法[3],Jeung 等人提出了 convoy[4]方法。Li 等人进一步释放了 convoy 方法的约束条件,并提出 swarm 模式去发现零星的对象组[5]。这些方法都需要移动对象组在至少 k 个连续的时间戳是在一起的,但这样在现实中可能会不实际。因为现实的移动对象可能会暂时离开,但是大部分时间是在一起运动,换句话说,强迫时间的连续性可能导致有趣移动对象簇的损失。在轨迹簇发现方面,Chen 等人提出了对于现实序列的编辑距离的概念[6],Tsai 等人提出了从轨迹簇挖掘组移动模式[7],Lee 等人指出建立在整个轨迹的距离衡量可能在子轨迹中丢失有趣的共同路径[8]。他们强调几何或空间的对象轨迹的紧密性。在现实生活中,经常移动对象的集合不可能一直一起移动,它们移动在一起只是在一些时间戳内。

随着 RFID 技术的广泛应用,挖掘 RFID 轨迹数据也变得非常重要。对 RFID 轨迹进行挖掘可以发现移动对象潜在的移动趋势,同时也可以发现成群的移动对象。轨迹聚类在数据挖掘中扮演一个重要的角色,由于 RFID 数据具有序列性特点,轨迹聚类通常以渐增格式接收。但现存的轨迹聚类算法多为静态数据集开发的,不适合于具有增量特性的 RFID 轨迹数据。这些增量轨迹数据挖掘在对用户进行重要物资的运输调度、药品的生产运输跟踪等方面都将有重要作用[9]。本文针对具有增量特性的 RFID 轨迹数据提出了相应的 RFID 轨迹聚类算法 TP-mine,将每个新的轨迹简化成一个有向线性片段以便于找到轨迹子部分的聚类,使用分割簇来存储紧密的相似轨迹线性片段,比原始的轨迹占的空间要小。一旦加入新的轨迹数据,分割簇渐增地更新以反映出变化。再使用整体聚类对分割聚类所生成的分割簇而不是所有时间段的全部轨迹进行操作。因为分割簇的数量比原始轨迹要少,在分割聚类所生成的分割簇数据集上进行整体聚类可以高效地得出轨迹的聚类结果。

① 国家自然科学基金(61003180),江苏省自然科学基金(DK20140958),江苏省产学研联合创新资金(BY2013063-08),江苏省"333"科研项目(BRA2012156)和江苏省"六大人才高峰"(2009180)资助项目。

② 男,1970 年生,博士,教授;研究方向:物联网与大数据管理,数据仓库和商务智能,中医药信息集成与大数据分析;联系人,E-mail:kfhu05@126.com

(收稿日期:2013-09-18)

1 问题定义

我们将输入的 RFID 轨迹数据表示为时间相关的轨迹数据集序列 $\{S_{t_1}, S_{t_2}, \cdots, S_{t_n}\}$，其中 S_{t_i} 是在时间 t_i 的轨迹集合。每个 $S_{t_i} = \{tr_1, tr_2, \cdots, tr_{ntr}\}$，每个 tr_j 是一个轨迹。一个单独的轨迹 tr_j 用折线表示，折线是连接的线段的序列，即物品在不同位置的连线，可以表示为 $tr_j = p_1p_2\cdots p_{len_j}$，每个点 p_i 是一个时间点。tr_j 可以进一步简化为一个新的折线，其具有更少的点，而偏离度要小于原来的阈值。

定义 1(聚类) 将物理或抽象对象的集合分成相似的对象类的过程称为聚类(clustering)。

定义 2(簇) 簇是数据对象的集合，同一个簇中的对象彼此相似，而一个簇与其它簇中的对象是相异的。

定义 3(分割簇) 一个轨迹分割簇由有方向的线段 l_1、l_2、…、l_n 集合组成，被定义成以下元组的形式：$(n, ls_{\text{center}}, ls_\theta, ls_{\text{length}}, ss_{\text{center}}, ss_\theta, ss_{\text{length}})$，$n$ 是分割簇中线段数量，ls_{center}，ls_θ，ls_{length} 分别是线段中心点、角度和长度的线性和。ss_{center}，ss_θ，ss_{length} 分别是线段中心点、角度和长度的平方和。

定义 4(代表性线段) 一个分割簇的代表性线段通过开始点 s 和结束点 e 来表示：

$$s = (center_x - \frac{\cos\theta}{2}len,\ center_y - \frac{\sin\theta}{2}len)$$

$$e = (center_x + \frac{\cos\theta}{2}len,\ center_y + \frac{\sin\theta}{2}len) \quad (1)$$

其中 $center_x = LS_{\text{center}_x}/N$，$center_y = ls_{\text{center}_y}/n$，$len = ls_{\text{length}}/n$，$\theta = ls_\theta/N$。

s 和 e 可以通过分割簇特征来计算。

2 轨迹分割

进行轨迹分割的主要目的是发现一些特征点，轨迹在这些点上改变很快。轨迹 $TR_i = p_1p_2\cdots p_{len_i}$，特征点集合表示为 $\{p_{c_1}, p_{c_2}, \cdots, p_{c_m}\}$ $(c_1 < c_2 < c_3 < \cdots < c_m)$。轨迹 TR_i 在每个特征点被分割，每个分割段都用线段来表示，图 1 是一个轨迹和轨迹分割的例子。

接下来的问题是如何选取这些特征点。最好的特征点既要保持原来轨迹和分割后的轨迹差异性尽可能地小，又要使轨迹分段的数量越小越好，这两点是相互矛盾的，同时满足是不可能的。如果一个轨迹中所有的点都作为特征点，它的精确度是最高的，但是分段的数量也是最多的；相反，如果只取开始点和结束点作为特征点，分段的数量最少，此时精确度却最低。因此，我们需要一个折中的方法来保证两者都是最优的。

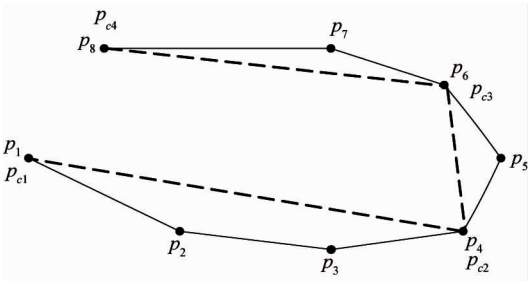

图 1 一个轨迹和它的分割

先定义一个线段和一个分割簇之间的距离。既然一个分割簇有它的代表性线段，距离实际上可以在 2 个线段之间定义，主要分为 3 部分：中心点距离 (d_{center})，角度距离 (d_θ) 和平行距离 $(d_\parallel)$。类似的距离测量在文献[8]被使用。与文献[8]不同的是，我们使用成分 d_{center}，而不是 $d_\perp$。选择 d_{center} 是因为它在一个更平衡的衡量比 d_θ 和 $d_\parallel$ 上。

令 s_i 和 e_i 是 l_i 的开始点和结束点，相似于 l_i 的 s_j 和 e_j。不失一般性，较长的线段分配给 l_i，较短的线段分配给 l_j。图 2 给了距离函数直观的解释。

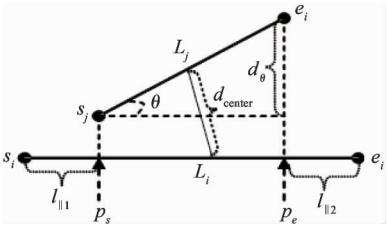

图 2 分割段之间距离

距离函数被定义为 3 个组成部分的和：

$$dist(L_i, L_j) = d_{\text{center}}(L_i, L_j) + d_\theta(L_i, L_j) + d_\parallel(L_i, L_j) \quad (2)$$

中心距离为 $d_{\text{center}}(L_i, L_j) = \| center_i - center_j \|$，其中 $\| center_i - center_j \|$ 是中心点 L_i 和 L_j 之间的欧几里德距离。

角度距离为

$$d_\theta(L_i, L_j) = \begin{cases} \| L_j \| \times \sin(\theta), & \theta(0° \leqslant \theta \leqslant 180°) \\ \| L_j \|, & 90° \leqslant \theta \leqslant 180° \end{cases} \quad (3)$$

其中 $\| L_j \|$ 表示 L_j 的长度,$\theta(0° \leqslant \theta \leqslant 180°)$ 表示 L_i 和 L_j 之间的最小夹角。

平行距离为 $d_{\|} = (L_i, L_j) = \min(l_{\|1}, l_{\|2})$,其中 $l_{\|1}$ 是 p_s 到 s_i 的欧几里德距离,$l_{\|2}$ 是 p_e 到 e_i 的欧几里德距离,p_s 和 p_e 是 s_j 和 e_j 在 L_i 上的投影点。

我们采用信息理论中最小描述长度(MDL)方法来发现最优的折中。MDL 主要有两部分组成:$L(H)$ 和 $L(D \mid H)$。$L(H)$ 是分段的长度之和,$L(D \mid H)$ 是轨迹和分段之间差异度之和。最佳情况下是使 $L(H)$ 和 $L(D \mid H)$ 的和最小。

$L(H)$ 和 $L(D \mid H)$ 的形式化定义如下:

$$L(H) = \sum_{j=1}^{m-1} \log_2(len(p_{c_j} p_{c_{j+1}})) \tag{4}$$

$$L(D \mid H) = \sum_{j=1}^{m-1} \sum_{k=c_j}^{c_{j+1}-1} \{\log_2(d_{\perp}(p_{c_j} p_{c_{j+1}}, p_k p_{k+1})) + \log_2(d_{\theta}(p_{c_j} p_{c_{j+1}}, p_k p_{k+1}))\} \tag{5}$$

接下来我们用一个实例来说明 $L(H)$ 和 $L(D \mid H)$,如图 3 所示。

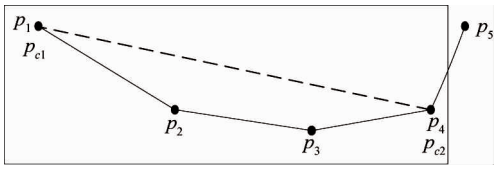

图 3　部分轨迹分割图

$$L(H) = \log_2(len(p_1 p_4)) \tag{6}$$

$$L(D \mid H) = \log_2(d_{\perp}(p_1 p_4, p_1 p_2) + d_{\perp}(p_1 p_4, p_2 p_3) + d_{\perp}(p_1 p_4, p_3 p_4)) + \log_2(d_{\theta}(p_1 p_4, p_1 p_2) + d_{\theta}(p_1 p_4, p_2 p_3) + d_{\theta}(p_1 p_4, p_3 p_4)) \tag{7}$$

图 3 显示了 $L(H)$ 和 $L(D \mid H)$ 的形式化定义。假设一个轨迹 $tr_j = p_1 p_2 \cdots p_{len_j}$,特征点的集合 $PC = \{p_{c_1}, p_{c_2}, p_{c_3}, \cdots, p_{c_{p_i}}\}$。$length(p_{c_j} p_{c_{j+1}})$ 表示线段 $p_{c_j} p_{c_{j+1}}$ 的长度,$L(H)$ 就表示所有轨迹分割线段长度之和。另外,$L(D \mid H)$ 表示一个轨迹和它的轨迹分割集合之间的差异性之和。为了衡量差异性,我们使用垂直距离和角度距离的和。我们之所以没有考虑平行距离是因为一个轨迹包含着它的轨迹分割。可以发现我们使用 $L(H)$ 来衡量的简洁性和用 $L(D \mid H)$ 衡量的精确度。由三角形的三边不等原理,$L(H)$ 随着轨迹分割数量的增加而增加,$L(D \mid H)$ 随着轨迹分割集合的增加而增加。

在定义完 $L(H)$ 和 $L(D \mid H)$ 之后,我们给出了特征点的选取算法,如算法 1 所示。

算法 1:特征点选取算法(CP-mine)

输入:一个轨迹 $TR_i = p_1 p_2 \cdots p_{len_i}$;

输出:特征点集合 CP_i;

1) 先将 p_1 加入集合 CP_i;

2) 设置开始索引 startindex = 1,length = 1;

3) 当 index + length $\leqslant$ len_i;

4) 当前索引 currIndex = index + length;

5) $\text{cost}_p = MDL_p(p_{startIndex}, p_{currIndex})$;

6) $\text{cost}_{nop} = MDL_{nop}(p_{startIndex}, p_{currIndex})$;

7) 如果 (cost_p > cost_{nop}),

8) 增加 $p_{currIndex-1}$ 到集合 CP_i; //在当前点分割

9) s

10) $startIndex = currIndex - 1$, $length = 1$;

11)　否则 $length = length + 1$;

12)　将 p_{len_i} 加入集合 CP_i。

算法 1 显示了特征点选取算法,我们对轨迹中的每个点计算 MDL_p 和 MDL_{nop}(算法 5,6 行)。如果 MDL_p 大于 MDL_{nop},立即将之前的点 $p_{currIndex-1}$ 插入到特征点的集合 CP_i 中。然后,我们重复以上的过程,否则增加候选项轨迹分割的长度。

简化后的轨迹表示为 $tr_j^{\text{simplified}} = l_1 l_2 \cdots l_n$,其中 l_i 和 l_{i+1} 直接通过线段连接(轨迹分割)。

给定如此的输入数据,目标是产生簇的集合 $O = \{C_1, C_2, \cdots, C_{n_c}\}$。一个簇是一个定向轨迹线段的集合 $C_i = \{l_1, l_2, \cdots, l_{\text{ln}}\}$,其中 l_i 是一个直线段,来自于在某个时间戳 t_i 的简化轨迹 $tr_j^{\text{simplified}}$。因为我们在线段上多聚类而不是整个轨迹,被我们发现的簇实际上是子轨迹簇,是许多移动对象经常访问的路径。

3　轨迹聚类算法

3.1　轨迹分割聚类算法

当有新轨迹到达时将会影响聚类的结果,此时引入轨迹分割簇来维持一个好的粒度聚类。分割簇比最终的簇是更严格的,每个簇仅仅包含和概括了部分轨迹的信息。轨迹分割聚类能实现更有效的对最终簇的计算。

算法 2 显示了形成和维持分割簇的流程,处理步骤如下。在一批新轨迹到达时,我们计算每个轨迹的每个线段 l_j 的最近分割簇 pc_k。如果在 l_j 和 pc_k 之间的距离小于一个距离的阈值 D,l_j 将被插入 pc_k。否则,将为 pc_{new} 创建一个新的分割簇 pc_{new}。如果新的分割簇的创建导致分割簇的总数量过载,一些分割簇将被合并。

算法 2:轨迹分割聚类算法(TP-mine)

输入:新的 RFID 轨迹的集合 $S_{t_i} = \{tr_1, tr_2, \cdots, tr_{ntr}\}$,已经存在的分割簇 $PC = \{pc_1, pc_2, \cdots, pc_n\}$ 和距离的阈值 D;

输出:更新后的 PC;

1) 对于 S_{t_i} 中的每个 tr_i 以及 tr_i 中的每个线段 l_j;

2) 从 PC 中找到离 l_j 最近的 pc_k;

3) 如果 $distance(l_j, pc_k) \leqslant D$;

4) 将 l_j 加入到 pc_k 并更新 pc_k, 否则;

5) 为 l_j 创建一个新的分割簇 pc_{new};

6) 当分割簇的大小超过内存约束时,就要合并一些分割簇。

3.2 建立和更新分割簇

当一个新的线段 l_i 被接收,我们首先找到可以接受 l_i 的最近的分割簇 pc_k(算法 2 的第 2 行所示)。如果在 l_i 和 pc_k 之间距离小于距离阈值 D, l_i 就被加入 pc_k,pc_k 相对的更新记录。反之,要创建一个新的分割簇(如算法 2 的 3 到 5 行)。

在发现最近的分割簇 pc_k 之后,如果到 L_i 的距离是小于 D, l_i 被插入到 pc_k, 并且在 pc_k 中的线性和和平方和同时被更新。因为它们仅仅是和,附加的属性应用和更新是有效的。如果最近的分割簇和 l_i 之间的距离大于 D,一个新的分割簇将为 l_i 创建。在新分割簇的初始测量要素如中心点、θ 和长度从线段 l_i 获取。

3.3 合并分割簇

在现实应用中,存储空间一直是个约束。如果分割簇总共使用的空间超过给定的空间阈值,一些分割簇不得不被合并来满足空间的约束。与此同时,如果分割簇数量继续保持增加,将会影响算法效率,因为最大的时间消耗部分是发现最近的分割簇。最重要的是,没有必要保存所有的分割簇,因为一些分割簇在几轮更新后可能变得更近。因此,当必须要加速效率和节省内存时,算法需要合并靠近的分割簇。明显地,包含类似线段的分割簇对是作为合并的最好候选项,因为合并结果有更少的信息丢失。

给定 M 个分割簇,在任意两个分割簇之间的距离被计算。将它们按照相似性从高到低排序。最相似的数对可作为合并的最好候选项。既然合并它们会导致最少的信息损失,那就将它们合并直到分割簇数量满足给定的空间阈值。

3.4 轨迹整体聚类

最后一步产生全部的轨迹簇。

当新的一批数据到来时分割聚类被采用,分割聚类被采用当它被用户调用时。既然通过分割簇之间的距离定义,很容易在时空点采用任意的聚类方法。我们需要简单地用分割簇之间的距离来替换时空点的距离。在我们的框架中,我们使用类似于 TRACLUS[8] 的基于密度的聚类方法,在整体聚类步骤中的聚类技术和在 TRACLUS 的聚类算法相同。唯一的不同是整体聚类是在分割簇集上进行而不是在 TRACLUS 中轨迹划分的集合上。分割簇被聚类通过基于密度的算法,发现最大的和密度关联的成分,形成一个整体簇。

4 实验与结果分析

目前,对于轨迹增量聚类的算法还很少,为了进一步说明本文所提出的轨迹聚类算法的有效性,我们将其与 TRACLUS 算法[8] 进行比较。实验环境是基于 Windows XP SP3、内存为 2GB, CPU 为 AMD 2.0GHz,利用 Visual C ++ 语言编写,所采用的实验数据为基于某 RFID 物流管理系统中的人工合成数据。

实验 1 是通过不同轨迹点的数量来进行有效性的比较,实验结果如图 4 所示。

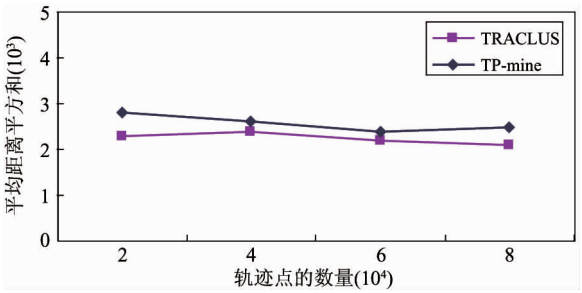

图 4 通过轨迹点数量进行有效性的比较

由图 4 所示的实验结果可以看出,与 TRACLUS 算法相比,TP-mine 算法的平均距离平方和要稍微高一点。但是我们在实验过程中发现,TP-mine 算法的处理时间比 TRACLUS 算法要少得多。因为,在分割簇上进行聚类要远比在整个轨迹上进行聚类快得多。

实验 2 是通过不同轨迹点数量比较两种算法的运行时间,实验结果如图 5 所示。

由图 5 可见,TP-mine 算法的效率要比 TRACLUS 算法高得多。因为 TRACLUS 算法进行轨迹分割的数量要远比分割簇的数量大得多。可以看出,当数据集越大时,TP-mine 算法的效率越明显。

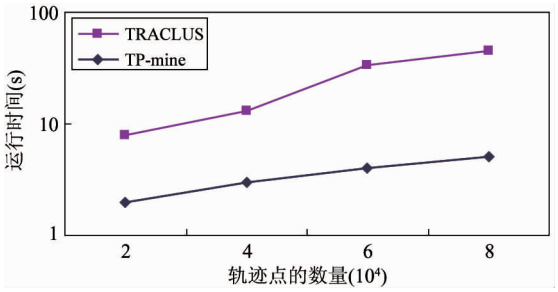

图 5　通过轨迹点数量进行效率的比较

5　结　论

本文提出的一种增量的 RFID 轨迹聚类算法 TP-mine,它将每个新的轨迹简化成一个有向线性片段以便于找到轨迹子部分的聚类,使用分割簇来存储紧密的相似轨迹线性片段,比原始的轨迹占的空间要小。一旦加入新的轨迹数据,分割簇渐增地更新以反映出变化。再使用整体聚类对分割聚类所生成的分割簇而不是所有时间段的全部轨迹进行操作。因为分割聚类所生成的分割簇数量比原始轨迹要少,在分割聚类所生成的分割簇上进行整体聚类可以很高效地得出轨迹的聚类结果。

参考文献

[1] Li Z, Han J, Ding B, et al. Mining periodic behaviors of object movements for animal and biological sustainability studies. *Data Mining and Knowledge Discovery*, 2011, 15(6): 1-32

[2] Kalnis P, Mamoulis N, Bakiras S. On discovering moving clusters in spatio-temporal data. In: Proceedings of International Symposium on Spatial and Temporal Databases, Angra Dos Reis, Brazil, 2005. 364-381

[3] Benkert M, Gudmundsson J, Hubner F, et al. Reporting flock patterns. In: Proceedings of the 16th ACM Sigspatial International Conference on Advances in Geographic Information Systems, Irvine, USA,2008. 497-528

[4] Jeung H, Yiu M, Zhou X, et al. Discovery of convoys in trajectory databases. In: Proceedings of the 34th International Conference on Very Large Data Bases, Auckland, New Zealand, 2008. 856-971

[5] Li Z, Ji M, Lee J, et al. MoveMine: Mining moving object databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Indianapolis, USA, 2010. 1203-1206

[6] Chen L, Ozsu M, Oria V. Robust and fast similarity search for moving object trajectories. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, Maryland, USA, 2005. 491-502

[7] Tsai H, Yang D, Chen M. Mining group movement patterns for tracking moving objects efficiently. *IEEE Transactions on Knowledge and Data Engineering*, 2011, 23(2): 266-281

[8] Lee J, Han J, Whang K. Trajectory clustering: A partition and group framework. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, Beijing, China, 2007. 593-604

[9] Wei L, Zheng Y, Peng W. Constructing popular routes from uncertain trajectories. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing, China, 2012, 195-203

TP-mine: A partition cluster based incremental clustering algorithm for RFID trajectory data mining

Hu Kongfa*, Xie Jiadong*, Zhao Li**

(*School of Information Technology, Nanjing University of Chinese Medicine, Nanjing 210023)

(**College of Information Engineering, Yangzhou University, Yangzhou 225009)

Abstract

Aiming at the mining of the RFID (radio frequency identification) trajectory data with the incremental properties, the study put forward a trajectory clustering algorithm, TP-mine. The TP-time simplifies each new trajectory into a directed linear segment in order to find the clusters of the trajectory's subparts, to store the similar trajectory line segments in partition-clusters which take much smaller space than raw trajectories. The integrity-clustering is performed on the set of partition-clusters rather than on all trajectories over the whole time span. Thus the integrity-clusters are generated efficiently to show the clustering results of trajectories. So the RFID trajectory data can be mined and the moving objects' potentially moving trends can be found.

Key words: radio frequency identification (RFID), partition-clusters, trajectory data, incremental clustering