Hans Journal of Wireless Communications 无线通信, 2013, 3, 123-128
http://dx.doi.org/10.12677/hjwc.2013.36019 Published Online December 2013 (http://www.hanspub.org/journal/hjwc.html)

Prediction Method Based on Target Trajectory Pattern Matching

Qun Lai[1], Jinye Yu[2], Shaoyan Jiang[2], Li Li[3]

[1]Guangdong Power Grid Yunfu Power Supply Bureau, Yunfu
[2]Guangdong Power Grid Zhongshan Power Supply Bureau, Zhongshan
[3]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing
Email: lldaxiang@163.com

Received: Oct. 8th, 2013; revised: Oct. 14th, 2013; accepted: Oct. 18th, 2013

Abstract: A specific target trajectory prediction algorithm based on mobile pattern matching is proposed, which is called PS-Tree algorithm. In this algorithm, historical data of targets' movements generated in monitored region are used for pattern mining, coordinate series of targets are converted to region trajectory series, and frequent moving modes are figured out by building up PS-tree. The algorithm matches current trajectories with the ones in pattern library to forecast the movements of the target when a target moves into monitored region. The simulation results prove that PS-Tree algorithm has low time and space consumption but high prediction accuracy.

Keywords: Data Mining; Target Trajectory Prediction; Mobile Pattern Matching; Prefix-Shared Tree

基于模式匹配的目标轨迹预测方法

赖　群[1]，余锦业[2]，姜绍艳[2]，李　莉[3]

[1]广东电网公司云浮供电局，云浮
[2]广东电网公司中山供电局，中山
[3]北京邮电大学网络与交换技术国家重点实验室，北京
Email: lldaxiang@163.com

收稿日期：2013 年 10 月 8 日；修回日期：2013 年 10 月 14 日；录用日期：2013 年 10 月 18 日

摘　要：本文在目标轨迹预测中采用了数据挖掘的方法，提出了一个具体的基于移动模式匹配的目标轨迹预测算法。该方法通过不断挖掘历史移动轨迹来构造前缀共享树的方法挖掘出频繁移动模式，之后通过模式匹配预测出目标的移动轨迹。仿真结果表明该算法的时间消耗和空间消耗较小，同时具有很高的预测准确性。

关键词：数据挖掘；目标轨迹预测；移动模式匹配；前缀共享树

1. 引言

在目标追踪无线传感器网络中，越来越多的数据证明目标的移动是有一定规律可寻的，而序列模式挖掘自 1995 年提出以来，随着研究的不断深入，数据挖掘技术已经日趋成熟，并且数据挖掘用于移动模式发现也有先例，所以不难从历史数据中挖掘出这些移动规律，得出一定的移动规律后可以将其应用到对目标的将来位置的预测中来。本章正是沿着这一思路展开了探索和研究，采用数据挖掘的方法，提出了一个具体的基于移动模式匹配的目标轨迹预测算法。算法对目标历史移动轨迹进行挖掘，形成一定的轨迹模式库。在移动目标进入监测区域时，根据当前路线和模

式库中的路线进行匹配，预测目标之后的移动路线。

2. 相关工作

移动轨迹挖掘是近年数据挖掘领域的研究热点之一，它的挖掘结果可以为许多应用提供有价值的信息，因此成为许多论文的研究对象。由于不同轨迹挖掘算法中考虑的研究对象稍有区别，因此可以把这些研究方法分为三类进行简单的介绍。

1) 以轨迹整体为对象的轨迹挖掘。

此类方法对于移动轨迹的相似性是基于移动轨迹整体考虑的，即考虑移动轨迹上所有点的相似性。两条轨迹相似，要求起点、终点以及移动轨迹中的所有点的位置都相近。论文[1]中给出了一种建立轨迹模型的方法，并且给出了两条包含了复杂数据类型的轨迹距离的计算方法。[2]中给出了两种使用经典的聚类算法，K-means 和层次聚类方法来对轨迹进行聚类的方法。

2) 以轨迹局部相似性的轨迹挖掘。

此类方法是基于整体的相似性的思想，这种思想并不适用于大多数移动的应用场景中，因为该方法无法发现某个时间片段上大量有相似性移动轨迹的用户群体。而且，对于离散化的移动数据特点来说，这种考虑轨迹上每个点相似度的思想所挖掘出的结果准确性也不能得到保证。因此，需要提出考虑轨迹局部相似性的算法来满足移动的应用。

针对这一问题，[3]中提到一种对移动轨迹进行划分然后再聚集的方法。它将一条移动轨迹划分成不同的部分，然后将这些零散的移动轨迹碎片进行聚集，找到相似的移动轨迹的部分。

3) 挖掘空间－时间轨迹序列。

前面提到的两类轨迹模式挖掘算法所考虑的对象是空间维上的移动序列。然而在许多应用中，时间也是分析物体移动模式的一个重要的维度，例如研究城市交通规划、动物行为规律等。因此，近年来有一类研究将时间维引入到轨迹模式挖掘中。[4]一文中提出了有时间标注的序列(TAS, Temporally Annotated Sequences)这一概念。有时间标注的序列指在一个移动轨迹序列中，任意两点(每点标识一个研究所感兴趣的地理位置或区域)之间有一个时间标注，即限制从一个地点到达另一个地点所花费的时间。

3. 相关定义

为了方便对目标轨迹预测方法的研究，我们针对频繁模式的数据挖掘做出如下定义：

定义 1(轨迹 *T*)：定义 *T* 为移动目标的运动轨迹序列，$T=\{(x_0,y_0),(x_1,y_1),(x_2,y_2),(x_3,y_3),\cdots,(x_{i-1},y_{i-1}),\cdots\}$，其中$(x_i,y_i)$表示某一时刻目标的位置坐标。轨迹的集合为*Tset*。轨迹示例图如图 1 所示：

定义 2(区域网格化)：将被监测区域 *G* 正六边形网格化(正六边形比正方形网格更接近传感器节点的感知范围，表示更为精确)。*G* 中的每个单元格定义为$R(x_i,y_i)$。

定义 3(区域轨迹 *T'*)：区域轨迹 *T'*是轨迹序列 *T* 所经过的区域坐标集合，我们将其定义为

$$T'=\{R_0(x_0,y_0),R_1(x_1,y_1),R_2(x_2,y_2),R_3(x_3,y_3),\cdots,R_{i-1}(x_{i-1},y_{i-1}),\cdots\}$$

图 2 中的黑色实线表示一条轨迹 *T*，蓝色区域为该轨迹 *T* 所对应的区域轨迹 *T'*。区域轨迹的集合为*T'set*。

基于上述定义，可以根据给定的目标移动历史轨迹挖掘出所有的频繁路径，将这些频繁路径作为移动目标的移动模式；之后，再依据这些移动模式并采用模式匹配方法对移动用户的移动轨迹进行预测。

4. PS-Tree预测算法

(1) 轨迹模式挖掘

为了根据移动目标的当前路径并结合移动轨迹模式来预测目标将来时刻的移动轨迹，我们设计了 PS-Tree 算法，首先根据区域轨迹构造前缀共享树，挖掘出频繁路径作为移动模式，然后根据移动目标的当前轨迹预测出目标即将出现的区域。

在构造前缀共享树之前，我们首先对树中的不同的节点类型进行说明：

1) 根节点：

匹配过程的起始节点，由于没有实际内容存储在该节点的中不能，所以作为一个叶节点的父节点。

2) 中间节点：

中间节点代表轨迹中的一个记录点，存储这个点的对应位置信息，中间节点的结构比起根节点要复杂，包含两个主要的部分，一个部分存储记录点的位置信息，另外一部分存储以该点为路径最终点的频次

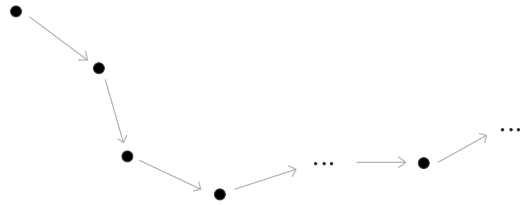

Figure 1. Example of trajectory
图 1. 轨迹示例图

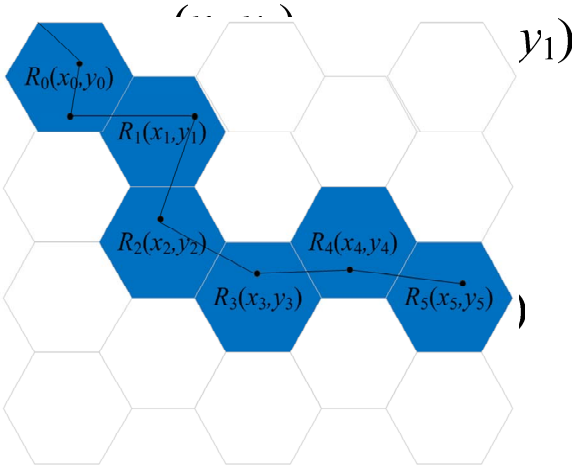

Figure 2. Example of regional trajectory
图 2. 区域轨迹示例图

信息。

3) 叶节点：

叶节点对应了一条路径的结束，同中间节点一样，在叶节点中也存储两部分，记录点的位置信息以及以该点位路径最终点的频次信息。

构造前缀共享树的过程如下：

1) 首先根据历史轨迹数据，将每一条轨迹定义为 T 的格式。

2) 对于每一条轨迹 T，首先将其转化为其对应的区域轨迹 T'。

3) 设置前缀共享树的根节点，从区域轨迹的第一个单元开始构造前缀共享树。

4) 对于区域轨迹中的单元，若树的当前层中有同名单元节点，则进入该节点的下一层(即该节点的子节点)；若树的当前层中没有同名单元节点，则创建同名节点，进入该节点的下一层。

5) 当处理完区域轨迹的最后一个单元时，若该路径之前不存在，则为该路径创建一个完成计数功能的叶节点，并初始化为 1；若该路径已经存在，则其叶节点的计数加一。

算法的执行过程如表 1 所示：

Table 1. Generation algorithm of prefix-sharing tree
表 1. 前缀共享树生成算法

```
Input: Tset
Output: prefix-sharing tree
for each Ti in Tset{
   for each Si in Ti{
convert Si into Ri according to its position
      set Ti' = Ti
   }
}
i = 0
while (i <Tset.number){
node_cur = root
    for (j=1, j<=Ti'.nodenumber, j++) {
tag = 0
       for each n in node_cur.childnodes
      if(Ti.node[j].name == n.name ){
      set tag = 1;
            set node_cur = n;
            break;
      }
         if(tag == 0){
      add node_new to node_cur.childnodes
      set node_cur=node_new
         }
         if(j==Ti'.nodenumber) {
      node_cur.count++;
         }
      }
}
```

算法首先采用历史轨迹集合 *Tset* 作为输入，对于 *Tset* 中的每一个节点 T_i 需要确定其对应的区域轨迹 T_i'，以准备加入前缀共享树。加入前缀共享树的过程分为两个步骤，定位过程与更新频次过程。对于每个点 $node_{new}$，分为轨迹中间点和轨迹终止节点。对于轨迹中间节点，如果轨迹中当前点与该层节点 $node_{cur}$ 某节点同名，则进入该节点的子节点层，并设为当前层 $node_{cur}$；如果没有同名节点，则代表树中没有共有前缀，需要在当前层建立新的节点，同时进入这个新

节点的子节点层。对于轨迹末节点，在插入同时，需要在刚刚建立的节点处，修改频次计数器 $node_{cur}.count$，将记录增加 1。前缀共享树就是按此规则建立。

举个例子，给定由历史轨迹转化为的区域轨迹集(在此将 $R_0(x_0,y_0)$简写为 R_0，并以此类推)：

$R_0 \rightarrow R_3 \rightarrow R_4$

$R_0 \rightarrow R_1 \rightarrow R_2$

$R_0 \rightarrow R_3 \rightarrow R_4$

$R_0 \rightarrow R_1 \rightarrow R_3 \rightarrow R_5$

$R_0 \rightarrow R_1 \rightarrow R_2$

$R_0 \rightarrow R_1 \rightarrow R_2 \rightarrow R_5 \rightarrow R_6 \rightarrow R_9$

$R_0 \rightarrow R_1 \rightarrow R_2$

$R_1 \rightarrow R_4 \rightarrow R_6$

$R_0 \rightarrow R_4 \rightarrow R_7$

根据算法，由该区域轨迹集构造的前缀共享树如图 3 所示。

(2) 基于模式匹配的轨迹预测算法

在上一步骤构造前缀共享树后，根据目标当前的移动轨迹，预测出目标即将出现的区域，过程如下：

1) 基于输入定位输入最终结点在树中的位置

2) 遍历该节点的子孙节点，根据标记单元计数找出高于阈值的高频点集

3) 返回高频点集到输入最终节点的所有路径

算法的执行过程如表 2 所示。

该算法的输入为一条轨迹 T，与一个频繁阈值 $threshold$，返回的是该轨迹的预测区域集合 $Pset$，该集合包含目标最可能的接下来要走的区域轨迹路径的坐标。对于每个轨迹中的节点，通过广度优先搜索，找到该节点在前缀共享树中的位置，如果没有，则返回该轨迹 T 直线的延长线上的坐标集合作为预测轨迹

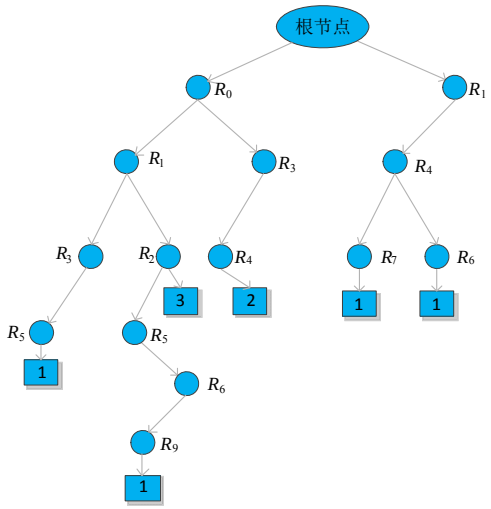

Figure 3. Example of prefix-sharing tree
图 3. 前缀共享树示例图

Table 2. Prediction algorithm of target emergence region
表 2. 目标出现区域预测算法

```
Input: T, threshold, Prefix-sharing Tree
Output: Pset
set Tnode = T.headnode;
Set TreeRoot = PrefixTree.root
Set node_cur= T_search(Tnode, TreeRoot);
If node_cur != null
   D_search(node_cur, threshold);
End if
For each node in frequentNodes
    If node != node_cur
        While node != node_cur
            add node in Pset
        End while
    End if
End for
Output Pset
T_search(node p, node q)
{
    Foreach node in q.childnodes
        if p.name == node.name
            if p ! = T.tail
                T_search(p.next, node);
            Else
                Return node;
            End if
        End if
}
D_search(node n, double threshold){
   if n.count>= threshold
      add n in nodefrequent;
   for all node_i in n.childnodes
      D_search(node_i)
}
```

Pset。找到一个节点后，则在该节点的子树中采用广度优先搜索确定轨迹中的下一个节点的位置，直到找到输入轨迹 T 的最后一个节点在树中的位置 $node_{cur}$ 上述工作在 *T_search* 函数中完成。接下来则扫描 $node_{cur}$ 对应的子树中包含的全部初始化过计数器 *node.count* 的节点，如果这些节点中某个节点 *n* 的计数频次 *node.count* 大于输入的阈值 *threshold*，则返回从 $node_{cur}$ 到这个节点 *n* 的全部路径，加上从 *root* 到 $node_{cur}$ 的路径，组成预测轨迹 *Pset*，返回给用户，在 *D_search* 中完成。需要注意的是，节点输入的阈值有两种形式，一种为次数型，比较的是指定节点的计数器数值与阈值的大小；另一种为比例型，比较的是指定节点计数器/树中所有计数器的值总和与输入阈值的大小。两种方法均可作为频繁轨迹的鉴定方式。

5. 仿真结果分析

为了测试算法的性能，我们进行了对比实验，将从模式挖掘效率以及预测准确性两方面对算法进行验证。所需的软件环境为：Windows 7，Eclipse 以及 matlab。仿真实验所采用的历史轨迹数据集为程序模拟生成，历史数据集为进入监测区域的轨迹集，其构建方式如下：从随机边缘网格内选取一点作为轨迹起始点。对于第 *i* 个点，构建点 $I+1$ 的方式为，随机生成 $\left(X_{i+1}-X_i\right)^2+\left(Y_{i+1}-Y_i\right)^2 \le r^2$，其中 *r* 为半径参数，可以改变。当轨迹中某点 *k* 构建完落在监测区域范围外或 *k* 的个数大于 50，当前轨迹构建终止，循环构建下一条轨迹。

1) 模式挖掘效率

本文选择了两个算法与 PS-Tree 算法进行对比。一个是文献[5]中基于 Apriori 的算法，Apriori 算法是第一个也是最有影响力的频繁模式挖掘算法，是一种自底向上的方法，产生所有的候选集然后测试并删去非法数据。另一个是文献[6]中基于 FP-Tree 算法，它可以作为频繁模式挖掘算法的里程碑。该算法使用了一种紧缩的数据结构来存储查找频繁数据集所需要的全部信息。将提供频繁数据集的数据库压缩到一棵 FP-tree 来保留数据集关联信息，然后将压缩后的数据库分成一组条件数据库(一种特殊类型的投影数据库)，每个条件数据库关联一个频繁数据集。数据流平均处理时间和空间消耗是用来衡量模式挖掘效率的两个重要因素，它们可以评估算法是否适用于该数据流的环境。

首先，我们使用不同的数据集规模来衡量数据流平均处理时间是否达到了预期目标，这里我们选择了四中不同规格的数据集模式，分别为 500、1000、2000 和 4000。数据流平均处理时间表示为 T_{avg}，其表达式如下：

$$T_{\text{avg}}=\frac{T_{\text{datasize\_x}}}{N_{\text{out}}} \tag{1}$$

其中，$T_{\text{datasize\_x}}$ 代表数据集大小为 X (数据集规格)处理数据所需要的时间，N_{out} 代表不同方法挖掘出的模式种类个数。

图 4 给出了三种方法的数据流平均处理时间比较。从曲线中可以发现，随着数据流的到达，基于 PS-Tree 方法比基于 FP-Tree 方法运行的要更快一些。因为 FP-Tree 的方法源自于 FP-Growth 的想法，因此，它的步骤与 FP-Growth 的流程类似，它将扫描两次数据集，一次用于构建 FP-Tree 的，还有一次是为了挖掘出 FP-Tree 的模式，所起耗时要长一些。而 Apriori 方法的数据流平均处理时间远远高于这两种方法，原因在于它会多次扫描数据集以确定哪些数据是频繁的。

在试验中，我们将比较基于 PS-Tree 的方法和基于 FP-Tree 的方法在空间上的消耗。由于基于 Apriori 方法将反复扫描并记录所有的数据集，所以不太适合用来比较空间消耗。

从图 5 可以看出，基于 PS-Tree 方法在空间消耗方面要优于基于 FP-Tree 的方法，因为基于 FP-Tree 的方法在挖掘过程中会保存所有符合条件的子树，大量的中间结果导致了很大的空间消耗。而基于 PS-Tree 的方法不会存储中间结果，因此 PS-Tree 曲线的增长趋势较为温和，空间消耗更小。

2) 预测准确性

轨迹预测方面的相关研究有很多，本文选择了文献[7]中 MPP 算法以及文献[8]中的 FPT 算法。MPP 算法中的模式挖掘算法是基于 Apriori 算法进行设计的，在执行过程中不可避免地要若干次遍历移动历史数据集，然后根据挖掘出的模式进行预测。而 FPT 预测算法是一种基于路网的不确定性轨迹频繁模式挖掘算法了，为了实现快速轨迹预测，设计了基于层次位图的轨迹模式索引树。仿真实验从历史数据中随机抽取 50~200 条轨迹进行预测实验，给出轨迹的前 20 个坐标，预测出轨迹区域。预测准确率如图 6 所示。

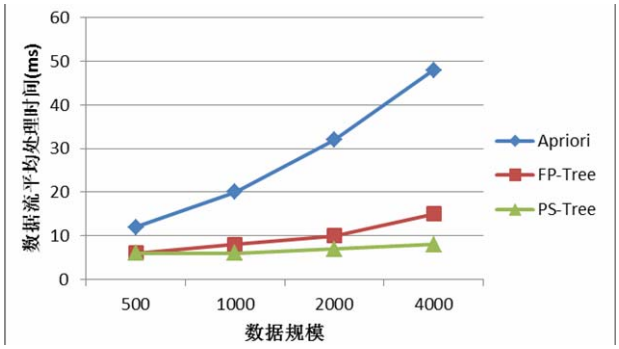

Figure 4. Comparison of average processing time for data stream
图 4. 数据流平均处理时间比较

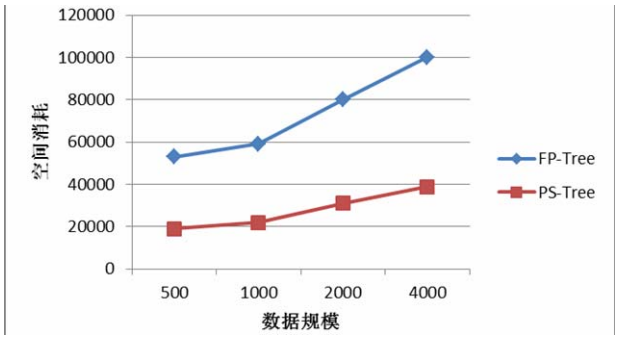

Figure 5. Comparison of space consumption
图 5. 空间消耗比较

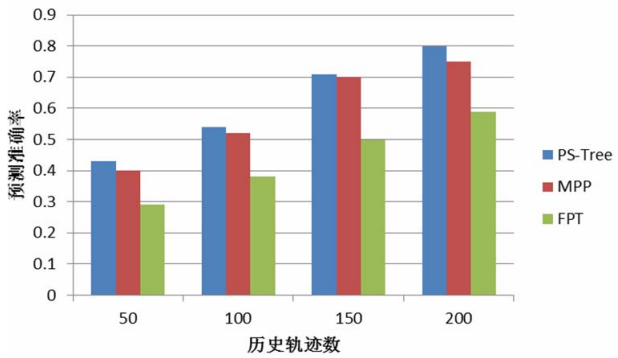

Figure 6. Forecasting accuracy
图 6. 预测准确率

从图中可以看出，在基于数据挖掘的预测算法中，给出了历史轨迹数越多，预测的准确率越高。PS-Tree 考虑到了由于 MPP 反复扫描数据集，预测精度较高，但是预测时间很长，耗费的空间很大。FPT 方法设计了轨迹模式索引树，虽然可以快速的定位查询，但是相对而言预测准确率则有一定的局限性。

6. 结论

本文提出了一种效率较高的目标轨迹预测方法(PS-Tree 算法)：该方法通过不断挖掘历史移动轨迹，通过构造前缀共享树的方法挖掘出频繁移动模式，之后通过模式匹配预测出目标的移动轨迹。本方法应用灵活，算法简单高效，具有很强的实用价值。

参考文献 (References)

[1] Ketterlin, A. (1997) Clustering sequences of complex objects. *Proceedings of KDD Conference*, ACM, New York, 215-218.

[2] Nanni, M. (2002) Clustering methods for spatio-temporal data. PhD Thesis, University of Pisa, Pisa.

[3] Nanni, M. and Pedreschi, D. (2006) Time-focused density-based clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, **27**, 1-20.

[4] Lee, J.-G., Han, J.W. and Whang, K.-Y. (2007) Trajectory clustering: A partition-and-group framework. *Proceedings of the* 2007 *ACM SIGMOD International Conference on Management of Data*, Beijing, 12-14 June 2007, 593-604.

[5] Giannotti, F., Nanni, M. and Pedreschi, D. (2006) Efficient mining of sequences with temporal annotations. *Proceedings of SIAM Conference on Data Mining*, SIAM (Society for Industrial and Applied Mathematics), Philadelphia, 346-357.

[6] Giannotti, F., Nanni, M., Pinelli, F. and Pedreschi, D. (2007) Trajectory pattern mining. *KDD*'07, ACM, New York, 330-339.

[7] Agrawal, R. and Srikant, R. (1994) Fast algorithms for mining-association rules. *Proceedings of the* 20*th VLDB Conference*, Santiago, 12-15 September 1994, 487-499.

[8] Han, J.W., Pei, J. and Yin, Y.W. (2000) Mining frequent patterns without candidate generation. *Proceedings of the* 2000 *ACM SIGMOD International Conference on Management of Data*, ACM, New York, 1-12.