基于FP-Tree模型的频繁轨迹模式挖掘方法

牛新征[1]，牛嘉郡[1]，苏大壮[2]，佘 堃[2]

(1. 电子科技大学计算机科学与工程学院 成都 611731；
2. 电子科技大学信息与软件学院 成都 611731)

【摘要】通过对经典频繁模式数据结构FP-tree的扩展与改进，提出了一种适用于处理轨迹数据的灵活高效的FP-tree轨迹挖掘方法(NFTM)。首先运用二维筛选和GPS格式过滤的方法对轨迹进行预处理，然后将有效数据经一次扫描后，生成按照真实轨迹顺序排列且具备时空属性的改进型FP-tree，使用动态数组存储模式挖掘过程中得到的候选集，根据用户的输入针对性输出相应时间和频率范围的频繁轨迹。最后通过与GSP算法、Prefixspan算法的对比测试表明，该算法具有更短执行时间和更优性能。

关 键 词 FP-tree； 频繁轨迹模式； 模式挖掘； 时空属性
中图分类号 TP393 **文献标志码** A doi:10.3969/j.issn.1001-0548.2016.01.014

FP-Tree-Based Approach for Frequent Trajectory Pattern Mining

NIU Xin-zheng[1], NIU Jia-jun[1], SU Da-zhuang[2], and SHE Kun[2]

(1. School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 611731;
2. School of Information and Software Engineering, University of Electronic Science and Technology of China Chengdu 611731)

Abstract Frequent trajectory pattern mining algorithms research focuses on how to make frequent pattern mining algorithms suitable for the mining of temporal and spatial trajectory database and reduce the times of scanning database. This paper proposes a novel trajectory mining algorithms based on novel FP-tree trajectory mining(NFTM)for a more flexible and efficient data processing which is achieved by the improvement and extension of the classic algorithm of frequent pattern structure FP-tree. The first step is the preprocessing of the original locus by using two-dimensional screening and the GPS format sifting. Then the valid data are scanned once only to generate the improved type of FP-tree, which is permutated based on the order of the authentic locus and which is also of the space-time attribute. The candidate collection derived from the process of the pattern mining in dynamic digit group storage is creatively applied. The frequent locus in the corresponding period and frequency range is output in line with users' input. Finally, through tests in comparison with two prevalent algorithms — the GSP algorithm and the Prefixspan algorithm, the conclusion is drawn that the new algorithm has the advantages of shorter executing time and greater ability.

Key words FP-tree; frequent trajectory pattern; pattern mining; spatial-temporal attribute

在过去的十年里，随着移动设备的发展和普及，使全球范围内各种大小的移动对象都得到较准确的定位和有效的跟踪。用户不但可以通过定位技术方便地获取个人位置信息，而且信号接收设备可以从定位终端上采集到大量移动对象的轨迹数据。轨迹记录着用户在客观世界的活动，一定程度上体现了人们的意图、喜好和行为模式。如用户的活动规律、所经过的地点顺序和最常走的几条路线等。越来越多的科研人员开始投入到GPS轨迹的研究，希望可以从粗放时空数据中挖掘出有价值的信息。

频繁轨迹模式挖掘是轨迹研究中最流行的一种，可以建立在频繁模式挖掘的基础之上，所以本文研究了对频繁模式挖掘经典算法FP-growth的改进创新，从而能灵活高效地从大量轨迹数据流中提取所需的频繁轨迹模式。其研究成果可以为个性化推荐、交通、旅游等应用领域提供方便的服务。

1 研究背景

频繁模式挖掘是数据挖掘研究中的一个重要课题。文献[1-2]提出的算法被视为频繁模式挖掘领域

收稿日期：2014－11－19；修回日期：2015－01－12
基金资助：国家自然科学基金(61300192)
作者简介：牛新征(1978－)，男，博士，副教授，主要从事网络计算、移动数据库、中间件技术、数据挖掘等方面的研究.

最基本的两大算法。文献[3]采用逐层迭代策略产生频繁项集，是一种自底向上的方法。为了提高其效率，研究人员提出了FP-growth算法，利用了一个特殊的存储结构FP-tree，能有效地提高挖掘效率。基于以上两大算法，研究人员提出了很多其他处理频繁模式挖掘的方法，如GSP[3]算法在Apriori算法的基础上引入了时间约束、滑动时间窗以及分类层次技术。文献[4]提出了PrefixSpan算法，采用基于投影的分治法来减小数据。文献[5]提出SPADE算法采用基于序列格的搜索技术和连续操作，将原始问题分解为子问题。

以上算法都旨在挖掘事物数据库，并未同时考虑到时间和空间属性。但是时空属性对于轨迹挖掘来说至关重要，文献[6]首先将原始轨迹转换成有时间约束的时空地点序列，提出了prefix-projection的方法来提取频繁时空模式。文献[7-8]定义了一个标有时态的轨迹序列模式Tpattern。文献[9]将分布式编程思想应用到频繁项集挖掘方法的MG-FSM[10]算法。

但以上诸多算法都需要两次以上的数据库扫描，降低了算法执行效率，因此，文献[11]提出FARM算法，生成了一个不仅考虑了数据的时间属性，还能在一个紧凑的数据结构中存储大量数据的新型框架。此外，基于MASTER-M[12]算法的数据估计框架还可以为MSN估算传感器丢失的数据。

由以上研究可以看出，对于频繁轨迹模式挖掘而言，目前最关键的两个研究点在于如何挖掘带有时空属性的轨迹和减少扫描数据库的次数[13-14]。本文针对这两个问题对FP-growth算法进行了改进：

1) 提出一种基于经纬度二维筛选和GPS格式过滤的轨迹预处理方法。

2) 改进FP-tree使其适用于轨迹处理，使用动态数组存储挖掘出的候选轨迹集。

3) 能根据用户的需要输出不同时间段和不同频率段的频繁轨迹。

2　问题描述

本文所研究的轨迹是带有时间属性的文本轨迹流，每条轨迹由多个地点串联而成，每两个地点之间还带有时间差。

定义 1　一个轨迹点v_i是由一个三元组(x_i,y_i,t_i)构成的，x_i ,y_i是经度和纬度，t_i是时间戳。

定义 2　一条时空轨迹是由序列 $S = v_0, v_1,\cdots, v_n = (x_0, y_0, t_0),(x_1, y_1, t_1), \cdots, (x_n, y_n, t_n)$构成的，$t_i(i=0\cdots n)$是时间戳，$\forall 0 \leqslant i < n, t_i < t_{i+1}$，$(x_i,y_i)$是轨迹中某一点的经纬度。

本文的轨迹流模式是在文献[7]上的扩展改进，原轨迹流模式定义如下：

$$T = v_0 \xrightarrow{t_1} v_1 \xrightarrow{t_2} \cdots \xrightarrow{t_3} v_n$$

式中，v_n为地点；t_n为时间戳。但不能认为从某地到下一个地点所需的时间可以严格控制为某个具体的时间点，而应该用一个大致的时间段来描述。所以本文采用的轨迹流模式如下：

定义 3　本文所使用的轨迹模式为：

$$T = v_0 \xrightarrow{t\alpha_1} v_1 \xrightarrow{t\alpha_2} \cdots \xrightarrow{t\alpha_n} v_n$$

式中，$t\alpha_n$为一个时间区间($t_{\min_n} \sim t_{\max_n}$)，表示从$v_{n-1}$～$v_n$所花费的最短时间和最长时间分别是$t_{\min_n}$和$t_{\max_n}$。

定义 4　给定一个轨迹数据库D，共有n条轨迹，轨迹支持度support是轨迹序列S的数量占总数n的百分比。

定义 5　最小支持度阈值min_sup是挖掘过程中过滤轨迹点和过滤轨迹的衡量标准，最后加入到频繁轨迹集中的轨迹S的支持度support(S)> min_sup。

3　NFTM模型建立

3.1　模型综述

本文的算法由3个模块组成: 预处理模块、轨迹树构建模块和频繁轨迹模式挖掘模块。在预处理模块中，根据地点数据的经纬度进行异常和冗余数据的检测并删除。然后将数据输入到第2个频繁轨迹树生成模块中，通过一次数据库扫描，动态地搭建一棵频繁轨迹树，并生成相邻结点之间的时间差。进入第三个挖掘模块后，可以根据用户所要求的时间和频率段输出频繁路径，以及走完该路线所花费的最长时间和最短时间。该模型的整体框架如图1所示。

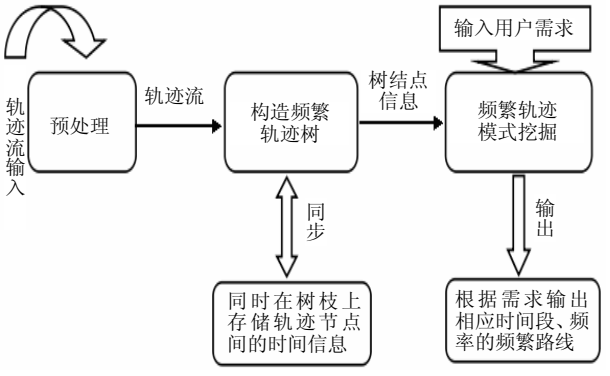

图1　NFTM的整体框架

3.2 预处理模块

本文提出一种基于经纬度和GPS数据格式[15]的轨迹数据预处理方法。

轨迹模式挖掘所用到的是GPS轨迹数据集，设每个数据点由一个包含经度、纬度和时间的三元组item构成，而原始数据集非常庞大且格式不统一，所以首先检查每个数据点的数据格式是否符合要求，将不规则点从数据集中删去。设数组ft[ ]存储经格式匹配函数过滤后的数据。将所有数据按经度进行分类，设数组sl[*m*][ ]用来存储具有相同经度*m*的数据项，再将同一经度的item集按照不同的纬度进行划分，并统计每类的item个数，用number表示。将小于阈值(通过对大量数据的测试得到，设为*T*)的数据组删除。由于数据的采集间隔时间较短，造成了大量的冗余数据，严重浪费了处理时间。所以将相邻连续的同一地点合并成一个，到达时间为第一个地点的到达时间。设final[ ]存放预处理后剩余的数据项。伪代码如下：

```
输入:  file directory (一个文件目录)
    For each file from 0 to k
        for each item from 0 to n
            if format(item_n)==true
                add item_n  to ft[ ];
        Put the triples which have the same
  longitude into sl[m][ ];
        for sl[ ][ ] from sl[0][ ] to sl[m][ ];
            for each item form sl[m][ ];
                if number of item_n < T
            remove (item_n) from sl[m][ ];
              i=0;
             While sl[m][ ]!=null
                  If sl[m][i]  ！ = sl[m][i+1]
                      add sl[m][i] to final[ ];
    the final[ ] coverage the original file
```

3.3 频繁轨迹树生成模块

在频繁轨迹树生成模块中，通过建立一种紧凑的数据结构存储树节点的属性和相邻结点之间的时间差值，只需扫描一次数据库可动态的构建轨迹树。构建轨迹树部分的规则如下:

1) 如果多个路径都经过某点，那么这个点应被合并成一项存储在项头表中。把两条轨迹从头结点开始的前几个相同节点合并，从不同的节点开始分支。

2) 对频繁模式的挖掘加入时间限制，分别挖掘出不同时间段的频繁路径或地点。时间段的限制根据用户的输入来定。

3) 考虑到实际应用场景，所以在构建轨迹树时将各地点按真实轨迹顺序排列。

4) 新增了一个有效的数据结构，用来存储整条频繁路径的时间长度。

使用一个例子来描述如何应用以上规则构建一棵频繁轨迹树，首先定义了时间限定在(0～20)的6条简化轨迹如下：

1) (v_1,1)—(v_2,3)—(v_3,6)—(v_4,7)

2) (v_1,1)—(v_2,9)—(v_4,13)

3) (v_4,3)—(v_2,9)—(v_5,11)—(v_6,14)

4) (v_4,2)—(v_2,3)—(v_6,4)—(v_7,8)

5) (v_4,7)—(v_1,9)—(v_7,14)—(v_6,17)

v_i代表地点，数字*n*代表时间。将以上6条轨迹输入本模块后，开始进行频繁轨迹树的构建，具体步骤如下：

1) 首先输入时间限制，将预处理后的数据根据时间属性再进行一次过滤，只保留在规定时间段内的数据。建立树的根结点，设置为NULL。

2) 第一条轨迹输入后，生成如图2所示的项头表，用来存储每个数据项的值、对应的支持度以及指向数据项在树中所处位置的指针。此外，在头结点下生成一条轨迹，构建频繁轨迹树的第一条分支。在树枝上即两个相邻的结点之间存储时间间隔。

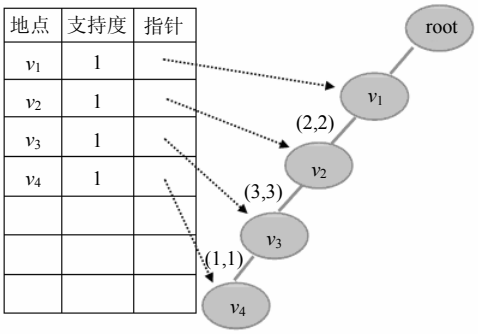

图2 输入(v_1,v_2,v_3,v_4)之后的存储结构

3) 输入第二条轨迹，支持度改变。v_1和v_2的出现顺序与第一条轨迹相同，所以合并路线。由于第二条轨迹中v_1和v_2的时间差为8，所以时间差的最大值由2改为8。从v_4起在树中生成了另一条轨迹分支。v_4结点第二次出现，所以第一条轨迹中的v_4还需要具备一个指向第二条轨迹中v_4的指针，称为指向下一同名结点的指针。输入第三条轨迹之后，如图3所示，由于新增了两个项头表中没有的地点v_5和v_6，所以将其加入项头表中，并添加指向它们的指针。

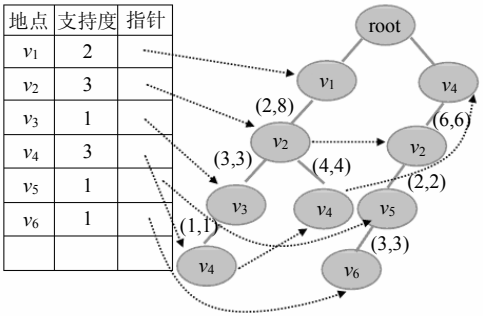

图3　输入(v_1,v_2,v_4) 和(v_4,v_2,v_5,v_6)之后的存储结构

4) 输入第四条轨迹，v_4、v_2和第三条轨迹重合，它们之间时间差的最小值由6变为1。将第一次出现的v_7地点与支持度连同指向它的指针加入项头表中。输入第五条轨迹之后，如图4所示，一颗完整轨迹树构建完成。

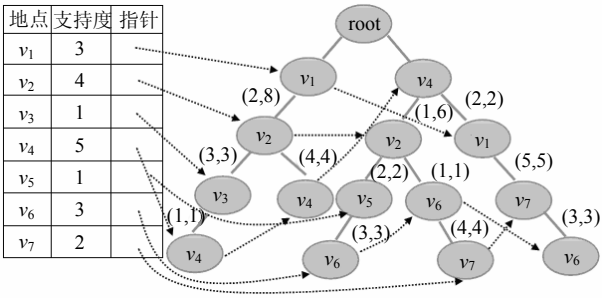

图4　输入(v_4,v_2,v_6,v_7)和(v_4,v_1,v_7,v_6)之后的存储结构

3.4　挖掘模块

在挖掘模块中，定义动态数组temp[ ]用来存放轨迹挖掘过程中的候选轨迹集，动态数组frequency[ ]用来存放轨迹的频率。初始化频繁轨迹集frequentSet[ ]和符合频率条件的轨迹集finalSet[ ]使其为空。伪代码如下：

输入：min_supp(最小支持度阈值)、start and end(需要挖掘的频率范围)

```
for all locations l_i in the header table
  if support(l_i)< min_supp
     traverse the node-link containing l_i;
remove all nodes have l_i from tree;
 for remaining items of header table
  for all items named l_i
   generate sub-tree with all paths ending by
l_i; the same combined into one route, the route
generating set S, placed in the temp[ ];
Put the support_route into the frequency[ ];
    for all routes in temp[ ]
       if S.support >= min_supp;
           add S to frequentSet[ ];
    for all routes in frequentSet[ ]
     if S.support >=start &&
    S.support< = end
         add S to finalSet[ ];
    output finalSet
```

从以上算法可以看出，本文相比于FP-growth算法的不同之处是：FP-growth算法在挖掘频繁模式时，需遍历FP树来递归的生成条件FP树，条件FP树的每一个结点至少需要3个指针，护起来要花费较多的动态空间。本文使用数组存储条件树，将遍历树得到的子路线存储在数组中，能有效地节约空间，提高时空效率。很多频繁轨迹模式挖掘的算法最终只输出最高频率的路径，这样极易造成对特定群体而言只输出一个地点，如对学生而言，最终可能只输出“学校”。所以经改进，可以根据用户的需求输出一定频率范围内的路径。

4　实验及其结果分析

4.1　数据集与环境

本文算法的测试部分所使用的是来自Geolife项目的真实数据集[GSP code 2013]，使用的测试集为200 KB，约12 000个地点项。测试集中的每一条轨迹路线都是由一系列地点项组成，每个项都包含经纬度和时间戳。将该算法与两个经典频繁模式挖掘算法GSP和Prefixspan进行了对比。所有测试实验都在具有2.30 GHz速度的处理器以及4.00 GB主存的电脑上完成。代码使用JAVA语言编写。

4.2　测试结果和分析

本文从以下3个方面进行了对比。

1) 第一个对比如图5所示。由图5可以看出，3个算法的运行时间都随着最小支持度的增长而增长。特别是GSP算法，在支持度小于0.55时，增长趋势平滑，但之后迅速增长，时间趋近于无穷。由此可得，该算法的运行时间比其他两个算法更短，且随着支持度的变化增长趋势稳定。

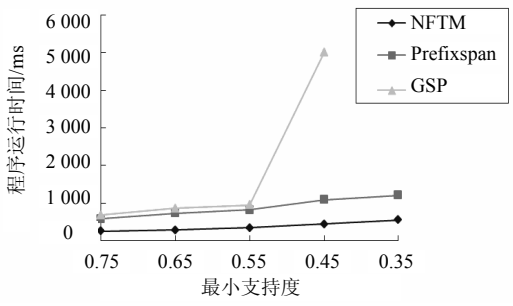

图5　不同最小支持度下的运行时间

2) 第二个对比将最小支持度统一为0.55，如图6所示。从图中可看出，Prefixspan和该算法都是随着地点数量的不断增加，程序运行时间变长。不同的是前者的增长速度较快，而后者变化趋势平稳，说明该算法对于处理不同地点数量的轨迹数据具有较优的性能。GSP算法变化趋势并不平缓，这是因为轨迹数据增加并不表明得到的频繁轨迹增加。如一个含1 000数据点的轨迹集可能有100个频繁1项集，但是一个含500个数据点的轨迹集可能有200个频繁1项集。GSP算法需多次扫描数据库，扫描到的频繁1项集越多，程序运行时间越长。而该算法只需要扫描一次数据库，所以比GSP算法性能更稳定。

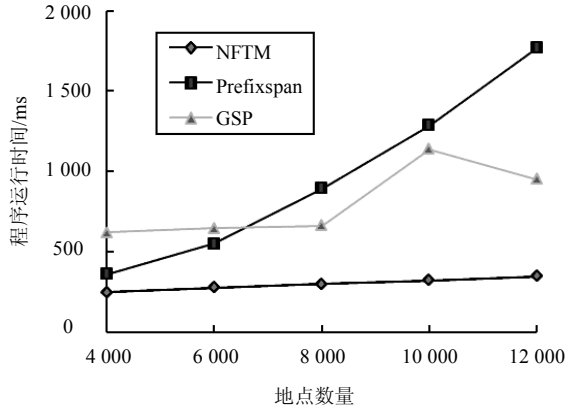

图6 不同地点数量下程序运行时间的变化

3) 第三个对比如图7所示。最小支持度固定在0.55，使用了4 000条轨迹进行测试。可以看出随着轨迹平均长度增加，程序运行时间越来越短，且变化平稳。该算法的运行时间比另外两个更短。

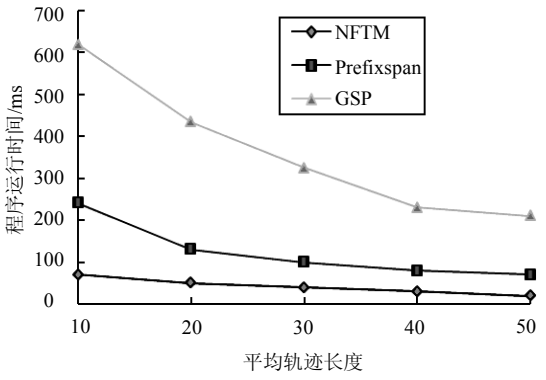

图7 不同平均轨迹长度下程序运行时间

4) 第四个对比如图8所示，规定参与测试的轨迹平均长度为20，最小支持度阈值设为0.55。所使用的轨迹数量由500～2 500。程序运行时间随着轨迹数量的增长而增长。该算法比其余两个算法整体运行时间短，且增长趋势更为平缓，可见性能更优。

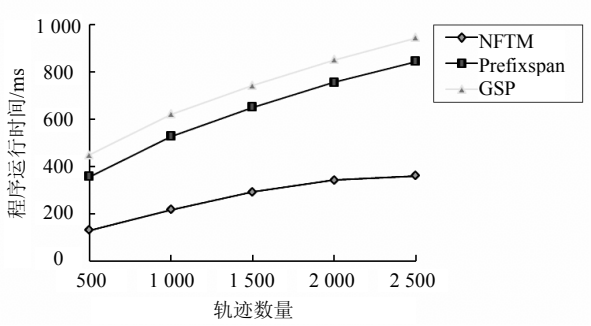

图8 不同轨迹数量下的程序运行时间

5 结 束 语

本文以FP-tree为模型，提出了一种更加优化且适用于处理轨迹数据的频繁轨迹模式挖掘算法NFTM。通过有针对性的预处理模块精简了轨迹数据。在构建轨迹树部分，只进行一次数据库扫描，并且生成按照真实顺序排列的同时具备时空属性的轨迹树。最后充分考虑各种现实因素，在挖掘部分使用数组作为存储结构，并加入了自主限定时间、频率的思想，使挖掘出的频繁轨迹更加人性化，符合实际应用需求。

参 考 文 献

[1] AGRAWAL R, SRIKANT R. Fast algorithms for mining association rules[C]//Very large data bases(VLDB). San Francisco, CA, USA: Morgan Kaufmann Publishers, 1994: 487-499.

[2] HAN Jia-wei, PEI Jian, YIN Yi-wen, et al. Mining frequent patterns without candidate generation: a frequent pattern tree approach[J]. Data Mining and Knowledge Discovery, 2004, 8(1): 53-87.

[3] SRIKANT R, AGRAWAL R. Advances in database technology—EDBT'96[M]. Berlin, Heidelberg: Springer, 1996: 1554-1558.

[4] PEI Jian, HAN Jia-wei, MORTAZAVI-ASL B, et al. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth[C]//ICDE'01 Proceedings of the 17th International Conference on Data Engineering. Washington D C, USA: IEEE Computer Society, 2001: 215-224.

[5] ZAKI M J. SPADE: an efficient algorithm for mining frequent sequences[J]. Machine Learning, 2001, 11(5): 31-60.

[6] KANG Juyoung, YONG Hwan-Seung Y. Mining trajectory patterns by incorporating temporal properties[C]// Proceedings of the 1st International Conference on Emerging Database. Busan, Korea: Hwan-Seung, 2009: 63-68.

[7] GIANNOTTI F, NANNI M, PINELLI F, et al. Trajectory pattern mining[C]//Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. California, USA : Scan Jose, 2007: 330-339.

(下转第134页)

Is there any (mobile wireless) small world out there?[J]. Ad Hoc Networks, 2012(10): 1520-1531.
[9] WEI K, ZENG D, GUO S, et al. Social-aware relay node selection in delay tolerant networks[C]//22nd International Conference on ICCCN: Computer Communications and Networks. Nassau, Bahamas: IEEE, 2013: 1-7.
[10] KERANEN A, OTT J, KARKKAINEN T. The one simulator for DTN protocol evaluation[C]//Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques. Arizona, USA: IEEE, 2009.
[11] EAGLE N, PENTLAND A. Reality mining: Sensing complex social systems[J]. Personal and Ubiquitous Computing, 2006, 10(4): 255-268.
[12] MERONI P, GAITTO S, PAGANI E, et al, Data setunimi/pmtr[DB/OL]. [2014-09-30]. http://crawdad.cs.dartmouth.edu /unimi/pmtr, Dec. 2008.
[13] SCOTT J, GASS R, CROWCROFT J, HUI P, et al. DataSetCambridge/haggle/imote/infocom2006[DB/OL]. [2009-05-29].http://crawdad.cs.d-artmouth.edu/cambridge/haggle/imote/infocom2006, May 2009.
[14] DERANGO F, AMELIO S, FAZIO P. Enhancements of epidemic routing in delay tolerant networks from an energy perspective[C]//9th International Wireless Communications and Mobile Computing Conference (IWCMC). Valencia, Italy: IEEE, 2013: 731-735.

编 辑 叶 芳

(上接第90页)

[8] PEI Jian, HAN Jia-wei, MAO run-ying. CLOSET: an efficient algorithm for mining frequent closed itemsets[C]// Proceedings of the 5th ACM-SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. Dallas, USA: ACM, 2000: 11-20.
[9] MILIARAKI I, BERBERICH K, GEMULLA R, et al. Mind the gap large-scale frequent sequence mining[C]// Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. New York, USA: ACM 2013: 797-808.
[10] DEAN J, GHEMAWAT S. Mapreduce: Implied data processing on large clusters[C]//6th Symposium on Operating Systems Design and Implementation. [S.l.]: USENIX Association, 2004: 137-149.
[11] GRUENWALD L, CHOK H, ABOUKHAMIS M. Using data mining to estimate missing sensor data[C]//IEEE International Conference on Data Mining Workshops. Omaha, NE, USA: IEEE, 2007: 207- 212.
[12] GRUENWALD L, YANG Han-qing , SADIK M S, et al. Using data mining to handle missing data in multi-hop sensor network applications[C]//9th ACM International Workshop on Data Engineering for Wireless and Mobile Access. New York, USA: ACM, 2010: 9-16.
[13] Microsoft Research Asia. GSP code [EB/OL]. [2011-10-30]. http://www.philippe-fournierviger.com/spmf/index.php?link= documentation.php.
[14] GENG Xiao-liang, ARIMURA H, UNO T. Pattern mining from trajectory GPS data[C]//2012 IIAI International Conference on Advanced Applied Informatics. Fukuoka, USA: IEEE, 2012: 60-65.
[15] LIANG Wan-ga, HUA Kunyuan, TAO Ku, et al. Mining frequent trajectory pattern based on vague space partition[J]. Knowledge-Based Systems, 2013, 50: 100-111.

编 辑 黄 莘

(上接第117页)

[6] PRABAVATHY B, PRIYA K, BABU C. A load balancing algorithm for private cloud storage[C]//Fourth International Conference on Computing, Communications and Networking Technologies. Tiruchengode: IEEE, 2013:1-6.
[7] PARVEZ N, WILLIAMSON C, MAHANTI A, et al. Analysis of bittorrent-like protocols for on-demand stored media streaming[C]//Proceedings of ACM Sigmetrics. Annapolis: ACM, 2008: 301-312.
[8] XIE T, QIN X. Security-aware resource allocation for real-time parallel jobs on homogeneous and heterogeneous clusters[J]. IEEE Trans on Parallel and Distributed Systems, 2008, 19(5): 692-697.
[9] CHRISTIAN C, KEIDAR I, SHRAER A. Trusting the cloud[J]. Newsletter ACM Sigact News, 2009, 40(2): 81-86.
[10] ZHANG Hong, LI Bo, JIANG Hong-bo, et al. A framework for truthful online auctions in cloud computing with heterogeneous user demands[C]//2013 Proceedings IEEE Conference on Infocom. Turin: IEEE, 2013: 190-201.
[11] JUN Feng, MARTY H. Eliminating replica selection-using multiple replicas to accelerate data transfer on grids[C]// Proceedings of the Tenth International Conference on Parallel and Distributed Systems. Washinton: IEEE, 2004.
[12] PATEL K, ANNAVARAM M, PEDRAM M. Generalized network flow-based resource allocation for hosting centers[J]. IEEE Transactions on Computers, 2013, 62(9): 1772-1785.

编 辑 叶 芳