一种基于受限网络的移动对象索引结构

易显天 徐 展 张 可 郭承军
(电子科技大学电子科学技术研究院 成都 611731)

摘 要 为了提高受限网络中移动对象索引效率和满足近邻查询需求,基于FNR-Tree索引结构和Geohash编码算法,提出一种能够满足近邻查询的移动对象索引结构RNR(restricted network R-Tree)。通过添加哈希表、链表等辅助索引结构来提升索引结构操作效率,融合Geohash编码和相关算法来使得索引结构能高效满足近邻查询的需求。通过将指定区域按一定规则划分,可使得索引结构具备在不规则范围查询的能力。使用旧金山市地理数据和移动对象数据对索引结构性能进行了测试,结果表明RNR具有较高索引结构操作效率,并且能够高效地提供窗口查询和近邻查询的功能。

关键词 受限网络,索引,近邻查询,移动对象,Geohash
中图法分类号 TP392 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.5.042

Index Structure for Moving Objects Based on Restricted Network

YI Xian-tian XU Zhan ZHANG Ke GUO Cheng-jun
(Research Institute Electronic Science and Technology of UESTC, University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract Aiming at improving the efficiency of index structure and providing neighbor query function, we proposed an index structure called RNR(restricted network R-Tree) which is based on FNR index structure and Geohash coding algorithm. This index structure can meet the demand of nearest neighbor query with comparative high efficiency. We improved the efficiency of RNR index structure by adding auxiliary index structure like hash table and list. We integrated index structure, Geohash coding algorithm and related algorithm to meet the need of neighbor query. We divided the designated area according to the certain rule to make the index have the ability to do irregular range query. We tested the RNR index structure efficiency by using San Francisco geographic data and moving object movement information data. The experimental results show that RNR index structure has high indexing efficiency and can provide efficient window query and neighbor query function.

Keywords Restricted network, Index, Nearest neighbor query, Moving object, Geohash

1 引言

近些年,移动对象数据库(Moving Objects Databases, MOD)作为移动计算技术的重要分支以及位置应用的底层支撑技术之一,得到了广泛的研究和重视[1]。移动对象数据库可以管理大量的移动对象信息,而不断发展的定位技术使得记录移动对象位置成为可能[2]。

索引作为移动对象数据库的核心技术之一,是提升移动对象数据库性能的重要研究内容。移动对象数据库索引的研究大致可分为两类:第一类是针对移动对象历史数据的索引[3],例如RT-Tree[1]等;第二类是针对移动对象当前和未来位置的索引,有PMR-quadtree[2]等。在实际应用中,人们越来越意识到在受限网络中运动的移动对象信息的存储管理的重要性[4]。目前在受限网络中经常使用的3种索引结构[5-7]为:FNR-Tree、MON-Tree和IMORS-Tre,三者各有优缺点。但是共同的缺点是索引效率不高,并且不支持最近邻查询。近邻查询在交通调度控制、位置服务等领域均有广泛运用。

本文基于FNR-Tree的思想,在添加辅助索引结构的基础上,融合Geohash算法,提出一种有更高索引效率及能够提供最近邻查询和不规则范围窗口查询的索引结构RNR(restricted network R-Tree)。

2 FNR和Geohash编码

2.1 FNR

FNR树(Fixed Network R-tree)的基本思想是采用静动态对象分别处理。索引结构分上下两层,上层是一棵静态的2D R-Tree,用来索引路网中的道路。其中每一个叶子节点包含一条路网中的路段信息。下层有多棵1D R-Tree,每棵对应一条路网中的路段,按照时间的顺序构建,用来索引对应路段上的移动对象信息。上层2D R-Tree的叶子节点中包含了

到稿日期:2014-06-29 返修日期:2014-12-23

易显天(1989—),男,硕士生,主要研究方向为空间数据库,E-mail:yixiantian7@foxmail.com;徐 展(1970—),男,硕士,高级工程师,主要研究方向为物联网相关技术、嵌入式系统设计;张 可(1979—),男,博士,副研究员,CCF高级会员,主要研究方向为物联网、计算机网络与信息安全;郭承军(1985—),男,博士,主要研究方向为卫星定位导航。

指向一棵下层 1D R-Tree 的指针。FNR 动静态对象分别处理的思想避免了在移动对象信息更新时对整个索引进行改动。但其仍然存在索引操作效率不高和不能有效提供轨迹查询、近邻查询等缺点。

2.2 Geohash 编码

Geohash 是一种经纬度编码方案，是在 geohash.org 编写 Web 服务时首次被提出来的。这种编码方式将二维空间细分成小格，每一个小格都有唯一的码来表示。之后基于 Geohash 的编码方式广泛应用于附近兴趣点(Point of Interest，POI)的搜索。

Geohash 的思想是在经纬度两个方向不断交替地对地球表面进行二分，一直循环划分到任意精度。具体为：当沿着经度方向二分时，左侧区域编码为 0，右侧区域的编码为 1；当沿纬度方向进行二分时，下侧区域编码为 0，上侧区域编码为 1。

经纬度数次二分之后，把指定位置对应的二进制从范围由大到小按序排序，可得到对应位置经纬度的编码，例如：纬度产生编码：1011100011，经度产生编码：1101001011。将经度代表的编码放在偶数位，将纬度代表的编码放在奇数位，重新组合经纬度的编码可得到新的编码：1110011101001000 1111。将二进制编码转换为 32 进制编码，再用 0～9，b～z(除去 a、I、l、o)对上面得到的 32 进制的编码进行编码即得到 Geohash 对应编码。

2.3 Geohash 特点

由 Geohash 编码方案可知其编码具有以下特点：1)唯一性，根据 Geohash 编码过程中区域划分规则，在划分地球表面过程中的每个单元格都有唯一的编码与其对应。2)每一个 Geohash 编码并不是表示一个二维点坐标，而是表示一个二维矩形区域。区域的大小由编码长度控制。故编码的前缀可以表示更大范围。3)Geohash 的每个编码同时表示经度和纬度两个坐标。

2.4 基于 Geohash 的近邻查询

Geohash 编码中同样的编码前缀表示处于同一个范围中，基于这样的特点，常将 Geohash 用于附近点查询。由于 Geohash 编码是基于空间填充曲线，而此空间填充曲线具有突变性，因此会出现编码是相邻，但距离却相差很远，例如：由图 1 可知对于 0111 与 1000，编码是相邻的，但距离相差很大。因此除了使用编码匹配进行筛选外，还辅助以实际的距离计算来得到最近邻点。

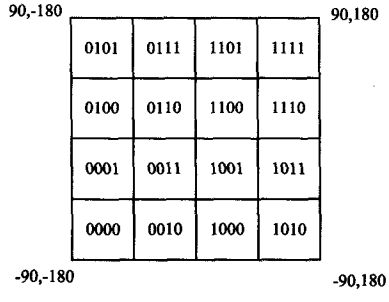


图 1 Geohash 编码

3 RNR 索引

3.1 索引结构

索引使用基于路段的模型，每条道路用路段和路段之间的两个端点的二维坐标来表示，整个道路网络是 $N=(R,D)$，R 是路段的集合，D 是端点的集合[8]。

RNR 索引结构分上下两层，上层索引结构用来保存路网信息，由 2D R-Tree 和哈希表 Road_Hash 组成。下层的索引结构保存在路网上的移动对象信息，由 1D R-Tree、哈希表 Mobj_Hash 和链表 mList 组成，上层 2D R-Tree 中的每个叶子节点中保存了一个指针，这个指针指向一棵下层的 1D R-Tree[9]。索引结构大致如图 2 所示。

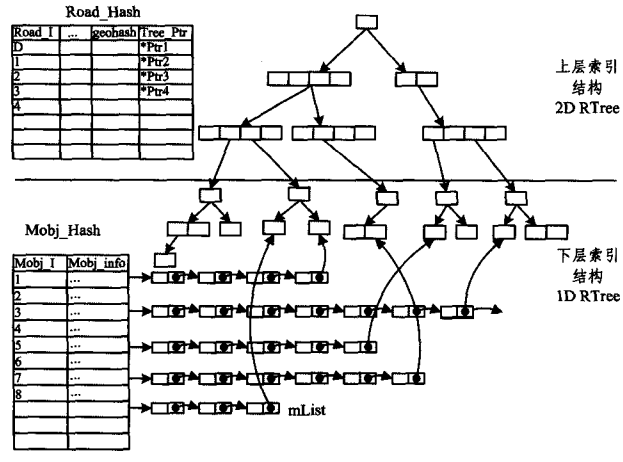


图 2 RNR 索引结构

3.2 上层节点及索引结构

2D R-Tree 用来存储路网信息，非叶子节点结构为(Ptr，MBR)，Ptr 是指向孩子节点的指针，MBR(Minimum Bounding Rectangular)为最小外包矩形，是包含其对应子节点中的 MBR 的最小矩形。叶子节点结构为(Tree_Ptr，MBR)，Tree_Ptr 是指向下层 1D R-Tree 树的指针，MBR 是包含一条路段的最小外包矩形。上层哈希表 Road_Hash 中每个数据项的结构为(Road_ID，Road_geohash，Tree_Ptr)，Road_ID 为路段的标识号，Road_geohash 为路段对应的 MBR 中心点对应的 Geohash 编码，Tree_Ptr 为指向对应路段 1D R-Tree 的指针。上层哈希表 Road_Hash 中每一个数据项对应一个 2D R-Tree 中的叶子节点。

3.3 下层节点及索引结构

1D R-Tree 用来存储路段中移动对象信息，非叶子节点数据项结构为(Ptr，time_extent)，Ptr 是指向子节点的指针，time_extent 表示时间范围，即移动对象停留在这个路段的时间段，用两个时间点表示。叶子节点为(leaf_info，time_extent)，time_extent 为时间范围，leaf_info 为一个结构体，包含移动对象信息，结构为(Road_ID，Mobj_ID，x，y，velocity)，Road_ID 表示移动对象所在路段的标识，Mobj_ID 表示移动对象标识号，x、y 表示移动对象所在二维位置，velocity 表示移动对象速度。

下层哈希表 Mobj_Hash 数据项结构为(Mobj_ID，Mobj_info)，Mobj_ID 为移动对象标识号，Mobj_info 为移动对象信息，其结构为(Road_ID，x，y，velocity，Mobj_geohash，updataTime_now，List_ptr)，Mobj_geohash 为移动对象位置对应 Geohash 编码，updataTime_now 为移动对象信息更新时间，List_ptr 为指向链表 mList 的指针。链表中每一个节点对应的数据项为(Road_ID，x，y，velocity，updataTime，node_Ptr)，node_Ptr 为指向 1D R-Tree 树中对应节点指针，其余数据项与前面提及的同名数据同义[10]。

4 RNR 索引操作

4.1 插入

上层索引结构插入算法如下：1. 读取一个路段信息，新建对应的 1D R-Tree。2. 把路段的 MBR 信息插入到 2D R-Tree 中，并把此路段对应的叶子节点中的指针指向刚新建的1D R-Tree 。3. 同时把此路段信息插入上层的哈希表 Road_hash 中，并把数据项中的指针指向刚新建的1D R-Tree。

下层索引结构插入如下：1. 读取一个移动对象实时信息。2. 把移动对象的信息插入 1D R-Tree 中。3. 同时把移动对象实时信息插入到下层哈希表和链表中，并把链表尾节点中数据的指针指向 1D R-Tree 中对应的节点。伪代码如下：

```
New RTree//新建 2D R-Tree
while(temp=ReadRoadInfo())读路段数据一行//
{ NEW RTree1//新建 1D R-Tree
  RTree. Insert(rect,&RTree1)//把路段 MBR、节点值等信息插入
  2D R-Tree 并把节点指针指向新生成的 1D R-Tree
  Road_Hash. Insert(temp,&RTree1)//把路段信息和 1D R-Tree 的
  指针插入上层哈希表中}
while(temp=ReadRoadInfo())
{ RTree1. Insert(temp);
  Mobj_Hash. Insert(temp);
mList. Insert(temp)}
```

4.2 更新

上层索引结构是对道路信息的存储，道路数据不常改变，可以看作是静态的索引结构，下层索引结构是对移动对象信息的索引。移动对象信息不断变化，下层索引结构需要随之更新。

一般地，移动对象在 3 种情况下发生数据更新：

1. 移动对象所在路段 ID 改变时触发更新。2. 预测的位置误差范围大于给定值阈值触发更新。3. 超过预定更新时间段触发更新。

下层索引更新算法如下：

当移动对象信息更新时，根据移动对象标识 Mobj_ID 在下层哈希表 Mobj_Hash 中找到对应项，在对应链表中遍历得到最后一个链表节点的信息，与更新的信息进行对比，如果移动对象所在路段 ID 没变，则只需修改移动对象在 1D R-Tree 和链表中的信息即可；如果移动对象所在路段 ID 发生变化，则需要先通过链表中最末端节点的指针找到对应 1D R-Tree 中的节点，删除此节点，然后再根据现有的移动对象所在路段 ID，在上层 Road_Hash 中找到对象项，索引到对应下层的对应 1DR 树中，重新插入更新后的移动对象信息，并同时更新在下层哈希表 Mobj_Hash 中的信息，且在链表 mList 中插入该移动对象信息，并将插入链表节点中的指针指向刚才更新的1D R-Tree 中的节点。伪代码如下：

```
if(Mobj_Hash. Find(Mobj_ID))//下层哈希表中有 Mobj_ID 记录
{if(mlist. LastNode. Road_ID=Mobj_ID. Road_ID)// 路段 ID 与
Mobj_ID 路段 ID 相同
{Update(milst. LastNode)}//更新链表信息
else{DeleteNode(mlist. LastNode. node_Ptr)根据 mlist. LastNode 中
包含的指针，找到 RTree1 中对应结点，并删除；
RTree1. Insert(Mobj_info);// 在 1D Rree 中插入信息 Update(Mobj_
Hash);
Update(mList);//更新哈希表和链表}
else{RTree1. Insert(Mobj_info);}
```

4.3 查询操作

范围搜索一般的形式为“范围 A 的路段在 10:00～12:00 这个时间段内有什么车在行驶”，可以根据 Geohash 匹配在上层哈希表中快速查询到对应项，根据其中的指针找到对应的 1D R-Tree，进行 1D R-Tree 的搜索即可。

伪代码：iter=Road_Hash. search(Road_geohash)//找到 Road_Hash 中的对应项

```
if(iter!=Road_Hash. end())//如果找到
{iter. TreePointer. search()}//进行 RTree 搜索，输出结果
```

轨迹查询为查询一个移动对象在过去某段时间的行驶轨迹。只要根据移动对象标识，在下层哈希表中查询，根据数据项中指针找到对应移动对象链表，然后将链表中符合时间条件的信息依次遍历读出即完成查询。

```
伪代码：iter=Mobj_Hash. find(Mobj_ID)
if(iter!=Mobj_Hash. end())
{Search(iter. mlist)//遍历链表，并输出结果}
```

最近邻查询为某一时间针对某一目标距离最近的一个或几个对象的查询。例如：“10 点离移动对象 A 最近的 10 辆车”。用 Geohash 算法进行前缀匹配后，在一个较小范围进行距离排序即可得到查询结果。伪代码见下文。

5 性能评估

索引结构采用 C++语言实现，实验平台配置为：Windows XP 系统、英特尔酷睿双核 3.2GHz 处理器、2.96GB 内存。将 RNR 与主流的路网索引 FNR[5] 和 MON 在更新效率、范围查询等实验中进行性能对比。测试采用 Brinkhoff 设计的路网移动对象生成器[11] 生成的动态移动对象数据。数据包含了一定数量的移动对象在一段时间内位置更新的相关信息，数据内容包括：移动对象 ID、此次信息的更新时间、二维位置 X 坐标、二维位置 Y 坐标、移动对象速度。实验中通过移动对象数据生成器产生 200－1600 个移动对象。

更新、范围查询、轨迹查询 3 组测试采用的是 Oldenbourg 地理数据，用来测试索引结构操作性能。以 IO 操作的多少作为性能评判的标准。把节点的一次读取作为一次磁盘的 IO 操作，IO 操作数愈少则性能愈优。

区域窗口查询和最近邻查询中需要使用 Geohash 算法，采用 San Francisco 地理数据，如图 3 所示，可以真实测试 Geohash 算法效率。因为路网信息相对较少，占用空间不多，而且数据几乎不变，可以将上层哈希表存储在内存中，IO 操作的消耗可忽略不计，在最近邻查询测试中以获取近邻点的平均时间为性能评判标准，时间愈少性能愈优。在窗口查询的准确性对比中，通过实验验证并分析误差来源。

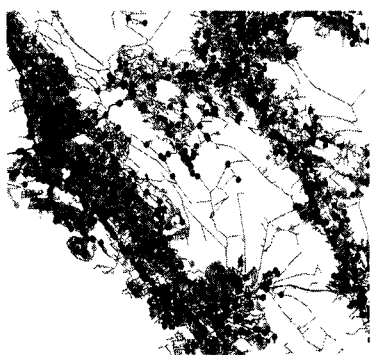
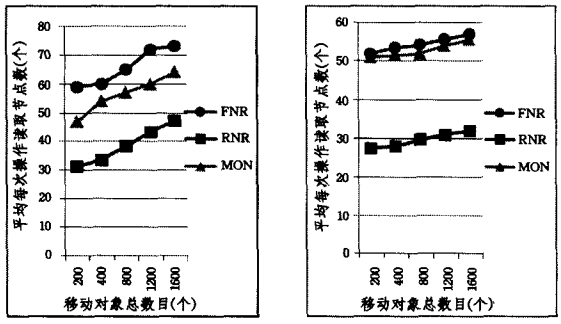


图 3 San Francisco 地理数据生成的地图

5.1 更新

图 4(a)显示的是 3 个索引结构的更新性能。FNR 更新的过程中总是需要先搜索上层 2D R-Tree 才能索引到下层 1D R-Tree，并采用自顶向下的更新方法，需要从根节点开始逐层访问，寻找到合适的位置。MON 的结构中也存在类似的问题。RNR 更新操作依靠哈希表可以快速索引到下层 1D R-Tree，通过底层哈希链表采用自底向上的更新方式，使大部分更新发生在叶子节点层和下层哈希链表中，可以很快完成局部更新。



(a)更新操作　　(b)范围查询

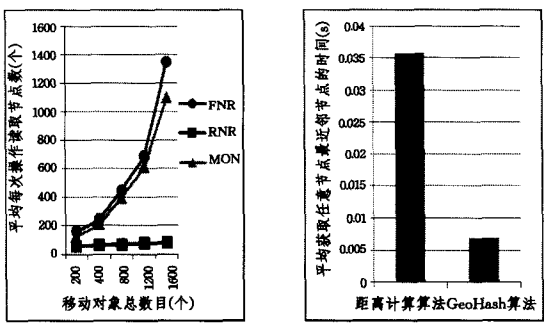
图 4

5.2 范围查询

图 4(b)显示了 3 种索引结构的范围查询的性能。FNR 和 MON 性能相差不大，原因是这两种索引结构存储道路的结构是相似的，都采用 R-Tree。而 RNR 把部分路段信息存储到了哈希表中，在进行范围查询时不必经过较为繁琐的 R-Tree 的搜索，而直接通过哈希表则可快速索引到下层 1D R-Tree 中，进行目标路段的搜索，有效地减少了 IO 操作。

5.3 轨迹查询

图 5(a)显示了轨迹查询的性能。FNR 和 MON 轨迹查询性能极低，查询过程需要遍历 R-Tree，几乎不能提供有效的轨迹查询，而 RNR 索引中由于增加了下层哈希链表结构，通过搜索哈希链表，按条件输出遍历链表即可完成轨迹搜索。由于链表遍历的效率极高，因此在轨迹查询中 RNR 的性能有显著提升。



(a)轨迹查询　　(b)时间性能

图 5

5.4 近邻查询

最近邻查询可以使用控制前缀匹配的长度来决定查询的范围，以一个较小的区域为中心，加上搜索相邻的 8 个同样大小的区域实现近邻查询，这样可以避免搜索的遗漏。先以 7 位为前缀匹配精度，如果范围内有目标则计算离每个目标的距离，获取符合查询要求的目标数目。如果范围内没有目标，或者目标少于查询要求数据，则缩短前缀匹配长度，扩大匹配范围。算法部分伪代码如下：

```
void KNN_search(int Mobj_ID,map<int,Mobj_Info> Mobj_Hash,int demandMun)
{ int accuracy=7;//前缀匹配精度
  int count=0;//要求获取最近邻目标数目
  map<int, Mobj_Info>::iterator iter = Mobj_Hash.find(Mobj_ID);//查询下层哈希表，获得移动对象 ID 对应的数据项
  while(1){
  for(map<int,Mobj_Info>::iterator i=Mobj_Hash.begin();i!=Mobj_Hash.end();i++)
{ if(0==geoHashCompare(iter->second.Ogeohash,i->second.Ogeohash,accuracy))//是否前缀匹配
  { count++;//获取目标数计数增加
  CalculateDistance()//计算离目标距离，与结果表中数据对比，判断后，决定是否存储 } }
  if(count<demandMun)//如果获取结果不够要求数目
  { extent--=1;//则缩短精度匹配范围，进行下一轮前缀匹配}
  else{break;//如果满足要求数目，则退出}}}
```

5.5 Geohash 性能测试

用 Geohash 前缀匹配得到最近邻点的方法对比 LBS 服务中另外一种常见方法：直接计算距离，再经过排序得到最近点。计算公式使用的是 Google 提出的地球表面距离计算公式：

$$s=2*\text{asin}(\text{sqrt}(\text{pow}(\sin((Lat1-Lat2)/2),2)+\cos(Lat1)*\cos(Lat2)*\text{pow}(\sin((Lng1-Lng2)/2),2)))*R$$

其中，(Lat1,Lng1)，(Lat2,Lng2)为 A,B 两点经纬度，R 为地球半径。图 5(b)中显示了距离计算和 Geohash 前缀匹配两种方法之间的效率。

相比之下 Geohash 算法性能明显好于距离计算算法。

5.6 窗口查询

以 R-Tree 为基础的范围查询都是矩形，可以使用一种大而化小的方式将查询区域分割，以便于使用以 Geohash 编码为基础的范围查询，其大致原理是将一块不规则的平面区域，限定最大的划分次数，将平面分成相邻的矩形格，以能够覆盖不规则区域的最少方格来表示这个区域大小。分割方式如图 6(a)所示，然后将方格通过 Geohash 编码，仍然通过前缀比较的方式确定范围内的目标。这样使得 RNR 结构具备了不规则窗口查询的功能。

图 6(b)显示了 RNR 中的区域查询和 MON 的等面积窗口查询得到的目标点的个数比较情况，随着查询面积的增加，两种方法查询到的目标数目的差距不断增加。分析可知：用 Geohash 前缀匹配的方法来代替通过 R-Tree 的窗口查询会有一定的误差，这个误差将会随着查询区域面积的增大和查询区域所在的经度的增加而不断扩大，原因是 Geohash 编码的前提是一个平面的区域，而地球是一个两极稍扁、赤道略鼓的球体，随着纬度的增加，1 经度带来实际距离的变化会逐渐减小。那么在赤道附近用 Geohash 前缀匹配等效的一个正方形区域，在高纬度的地方就会变成一个长方形区域。但是我们可以采用在不同经度值下的固定误差值来修正 Geohash 的结果值以达到避免这种误差的目的。

(下转第 220 页)

ognition for disease [J]. IEEE Proceedings of the First International Conference on Machine Learning and Cybernetics, 2002, 1(4): 50-54

[2] 史开泉，崔玉泉. S-粗集与它的结构 [J]. 山东大学学报：理学版，2002，37(6)：471-474

[3] Shi Kai-quan, Cui Yu-quan. F-decomposition and $\bar{F}$-reduction of S-rough sets [J]. An International Journal of Advances in Systems Science and Applications, 2004, 4(4): 487-477

[4] Shi Kai-quan, Chang Ting-cheng. One direction S-rough sets [J]. International Journal of Fuzzy Mathematics, 2005, 13(2): 319-334

[5] Shi Kai-quan. Two directions S-rough sets [J]. International Journal of Fuzzy Mathematics, 2005, 13(2): 335-347

[6] 史开泉，崔玉泉. S-粗集与它的分解-还原 [J]. 系统工程与电子技术，2005，27(4)：644-651

[7] Hu Hai-qing, Wang Hong-yu, Shi Kai-quan. Two directions S-rough recognition of knowledge and recognition model [J]. An International Journal of Advances in Systems Science and Applications, 2005, 5(3): 368-374

[8] 史开泉. S-粗集与新材料发现-识别 [J]. 系统工程与电子技术，2006，28(3)：383-388

[9] Shi Kai-quan. Function S-rough sets and function transfer [J]. An International Journal of Advances in Systems Science and Applications, 2005, 5(1): 1-8

[10] 史开泉. 函数 S-粗集 [J]. 山东大学学报：理学版，2005，40(1)：1-10

[11] 张萍，史开泉，卢昌荆. 函数 S-粗集与粗规律-分离 [J]. 系统工程与电子技术，2005，27(11)：1899-1902

[12] Zhang Ling, Shi Kai-quan. Security transmission and recognition of F-knowledge [J]. Journal of Systems Engineering and Electronics, 2009, 20(4): 877-882

[13] Qiu Jin-ming, Shi Kai-quan. F-rough law and the discovery of rough law [J]. Journal of Systems Engineering and Electronics, 2009, 20(1): 81-89

[14] 史开泉，姚炳学. 函数 S-粗集与规律辨识 [J]. 中国科学 E：信息科学，2008，38(4)：553-564

[15] 史开泉，赵建立. 函数 S-粗集与隐藏规律安全-认证 [J]. 中国科学 E：信息科学，2008，38(8)：1234-1243

[16] Shi Kai-quan, Yao Bing-xue. Function S-rough sets and law identification [J]. Science in China F: Information Sciences, 2008, 51(5): 499-510

[17] Shi Kai-quan, Zhao Jian-li. Function S-rough sets and security-authentication of hiding law [J]. Science in China F: Information Sciences, 2008, 51(7): 924-935

[18] 史开泉. 函数 S-粗集，函数粗集与信息系统规律拆分-合成 [J]. 计算机科学，2010，37(10)：1-10

[19] 史开泉. S-粗集与数据挖掘单位圆特征 [J]. 计算机科学，2010，37(5)：1-8

[20] Yoon Y, Lee J, Park S, et al. Direct integration of microarrays for selecting informative gene and phenotype classification [J]. Information Science, 2008, 178(12): 88-105

[21] Vogiatzis D, Tsapatsoulis N. Active learning for microarray data [J]. International Journal of Approximate Reasoning, 2008, 47(1): 85-96

[22] Pawlak Z. Rough set approach to knowledge-based decision support [J]. European Journal of Operational Research, 1997, 87(99): 48-57

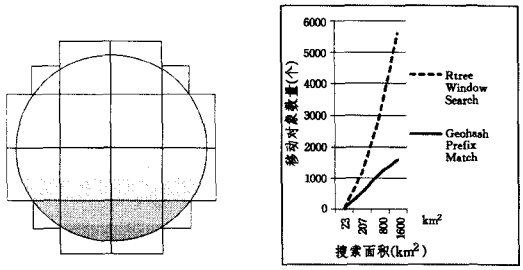
[23] Pawlak Z. Rough sets and fuzzy sets [J]. Fuzzy Sets and Systems, 1985, 79(17): 99-102

[24] Pawlak Z. Rough sets [J]. International Journal of Computer and Information Sciences, 1982, 66(11): 341-356

[25] 史开泉，崔玉泉. S-粗集与粗决策 [M]. 北京：科学出版社，2006，155-165

[26] 史开泉，姚炳学. 函数 S—粗集与系统规律挖掘[M]. 北京：科学出版社，2007：147—198

（上接第 214 页）



(a)分割方法　　(b)范围查询误差

图 6

结束语　本文在 FNR 索引结构的基础上提出了 RNR 索引结构，通过增加辅助索引结构，使得索引结构在更新操作、范围查询、轨迹查询等操作上的效率有大大提升，并且融合 Geohash 算法，使得索引结构能够提供高效的最近邻查询和窗口查询等功能。性能评估实验中使用能够提供真实经纬度数据的 San Francisco 地图数据，验证了近邻查询效果，也对比验证了 Geohash 算法的效率。但是研究中对道路之间的关系考虑不多，下一步将着重研究道路之间的关系，以进一步完善索引结构。

参 考 文 献

[1] 廖巍，熊伟，景宁. 移动对象索引技术研究进展[J]. 计算机科学，2006，33(8)：166-169

[2] 肖晖，李清泉. 移动对象数据库索引研究综述[J]. 计算机应用，2010，30(4)：1064-1067，1071

[3] Yu X, Chen Y. A moving object database model based on road network [J]. Journal of Software, 2003, 1498(9): 1600-1607

[4] Chen J, Meng X. Update-efficient indexing of moving objects in road networks[J]. GeoInformatica, 2009, 13(4): 397-424

[5] 曾倩，金敏. 基于道路分布的移动对象动态组合索引方法[J]. 计算机应用，2008，28(1)：3251- 3253

[6] 李峰，罗磊. 基于道路网络的时空索引方法 IMon-tree[J]. 计算机应用，2012，32(8)：2205-2208

[7] Kyoung-S Kim. Fast indexing and updating method for moving objects on road networks[C]// Proc. 4th Int'l Conf. Web Information Systems Engineering. Los Alamitos, CA: IEEE Computer Society Press, 2003: 34-42

[8] 丁治明. 一种适合于频繁位置更新的网络受限移动对象轨迹索引[J]. 计算机学报，2012，35(7)：1448-1461

[9] 宋广军，郝忠孝，王丽杰. 一种基于受限网络的移动对象索引[J]. 计算机科学，2009，36(12)：138-141

[10] Zhu Y, Zheng V, Yang Q. Computing with Spatial Trajectories [M]. New York: Springer New York, 2011

[11] Brinkhoff T. Generating network-based moving objects [C]// Proc of the 12th Int'l Conf. on Scientific and Statistical Database Management(SSDBM00). 2000: 253-255