基于CURE聚类算法的静态R树构建方法

李 松 崔环宇 张丽平 经海东
(哈尔滨理工大学计算机科学与技术学院 哈尔滨 150080)

摘 要 R树索引结构在空间对象查询和复杂空间关系查询方面具有重要作用。传统空间索引结构R树是动态生成的,树的结构是根据连续插入算法实现的,通过分裂子节点直至生成R树的根节点。动态生成算法会导致R树节点最小外包矩形之间的大量重叠,影响空间查询效率,且空间利用率不高。为了弥补动态生成R树的不足,提出了基于CURE算法的静态R树生成方法,给出CU_RHbuilt建树算法,该算法不仅能有效地处理海量数据,识别任何形状的簇,减少矩形重叠度,而且采用划分技术可较大程度地减小计算代价,空间利用率较高。进一步提出了基于CURE算法的R树节点分裂方法。理论研究与实验表明,所提方法具有较高的查询效率。

关键词 传统R树,静态R树,CURE算法,海量数据
中图法分类号 TP311.13 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.10.039

Static R-tree Building Method Based on Cure Clustering Algorithm

LI Song CUI Huan-yu ZHANG Li-ping JING Hai-dong
(School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China)

Abstract The R tree index structure plays a great role in spatial objects query and complex spatial relations query. The traditional spatial index structure of R tree is generated dynamically. The structure of its tree is realized according to the continuous insertion algorithm. It uses the way of splitting child node to generate the root node of R tree. Dynamic generation algorithm will cause low minimum utilization rate of the node of R tree. In order to make up for the inadequacy of dynamically generated R tree, a static R tree algorithm based on CURE algorithm was proposed, and the CU_RHbuilt tree building method was put forward. This algorithm can not only effectively deal with massive data, recognize clusters of any shape, reduce the overlap degree of rectangles, but also greatly reduce the computational cost as partitioning technology is adopted. The spatial utilization is rather high. The R tree node splitting method based on CURE algorithm was further proposed. Theoretical research and experiment show that the query efficiency of the proposed method is rather high.

Keywords Traditional R-tree, Static R-tree, CURE algorithm, Massive data

1 引言

随着GIS的快速发展,对复杂空间数据(确定和不确定数据信息)的有效处理变得越来越重要,空间数据和空间关系的索引与查询技术的应用也越来越广泛[1,2]。B树等适用于关系型数据库的索引结构已经不能满足人们对处理海量数据信息和复杂空间关系的需求。对索引结构的选择和对现存的索引结构的改进成为当前空间数据库研究领域的热点和难点。

R树[3]作为当前应用比较广泛的一种数据结构,在空间数据库中得到了广泛的应用,如基于R树的最近邻查询[4]、路网中的K近邻查询[5]、连续近邻链查询[6]、云环境下的空间索引[7]、基于R树的空间拓扑关系查询等。鉴于经典R树结构在空间数据查询和空间关系查询方面具有一定局限性,国内外一些学者对R树的改进方法进行了深入的研究。Brakatsoulas等人通过对R树结构的研究,提出了CR(Clustering-R)树[8],利用K-means聚类方法来建立R树,把传统的两路分裂改进为由聚类技术支持的多路分裂。聚类算法有很多种,各有优缺点。例如K-means具有对孤立点比较敏感、依赖于初始值的选择等不足。把K-means应用到R树中所衍生出来的CR树仍然会"继承"聚类算法的缺点,影响R树的体系结构。而且单纯地采用递归算法,会使得建树的时间消耗变得相对较大,不利于处理海量数据。黄继先等[9]根据空间对象的均匀分布与不均匀分布,实现R树节点分裂时分别采用不同的聚类准则以提高其聚类效果和查询效率。由于聚类的结果不能预知,对象可能分布不均匀,为了使R树结构更加平衡,需要调整聚类结果,这将改变原有的对象关系,增加了查询代价。刘润涛、郝忠孝通过建立数据矩形间的

到稿日期:2014-08-25 返修日期:2014-11-22 本文受黑龙江省教育厅科学研究项目(12541128)资助。

李 松(1977-),男,博士,副教授,主要研究方向为空间数据库、数据挖掘,E-mail:lisongbeifen@163.com;崔环宇(1988-),男,硕士生,主要研究方向为空间数据库;张丽平(1976-),女,硕士,副教授,主要研究方向为数据结构和算法设计、空间数据库;经海东(1990-),男,硕士生,主要研究方向为空间数据查询。

序关系对提出了 RQOP 树[10]，使索引树的高度变低，减少节点之间的重叠，提高了索引效率。但是在利用四叉树的同时，一个对象可能会分配给多个叶子节点，容易造成最小外包矩形(MBR)的重合，对索引效率产生影响。汪璟玢提出基于 k-medoids 聚类算法来构建 R 树[11]，弥补了 K-means 对孤立点敏感的不足，以此来提高查询效率和节点空间利用率。但 k-medoids 算法存在初始值 k 的选择问题，影响了聚类和建树效果。

刘润涛等[12]结合改进的 K-均值算法，提出了一种基于 K-means 构建 R 树的新方法，其利用聚类特性，减少矩形的重叠度，目的是使该索引结构更加紧密并提高索引效率。但是 K-means 算法本身具有对孤立点比较敏感、依赖 K 值的选择、采用平方误差函数会造成局部最优而非全局最优等不足。那么利用 K-means 构建 R 树时，这些缺陷仍然会体现在数据结构中。为了弥补已有方法的不足，本文提出了一种基于 CURE 算法的 R 树构建方法，即利用聚类技术，使节点的 MBR 几乎不重叠，以提高索引效率。所提方法弥补了 K-means 在孤立点上的不足，避免了初始值 k 的选择问题，从而提高了聚类效果和节点的空间利用率；而且利用 CURE 算法的取样和划分技术，能够使 R 树更加有效地处理海量的复杂空间数据信息。

2 R 树特点

R 树在空间数据库中应用非常广泛，能有效地解决空间多维属性数据的索引，提高对海量数据和复杂空间关系(如拓扑关系和方向关系)查询的效率。R 树是 B 树在高维空间中的扩展，同样也是一种高度平衡树。国内外学者对 R 树进行了改进，提出了 R 树的一些变种：R^+-树、R^*-树、Hilbert-R 树等数据结构。在 R 树家族中，R^*-树[13,14]是最有效的变种之一，它采用强制重新插入的方法来分裂 R 树节点以提高 R 树的性能，但是其代价较高。对于一棵 M 阶 R 树，其结点结构可以描述如下。

叶子节点：

$(COUNT, LEVEL, \langle OI_1, MBR_1\rangle,$

$\langle OI_2, MBR_2\rangle, \cdots, \langle OI_M, MBR_m\rangle)$

中间节点：

$(COUNT, LEVEL, \langle CP_1, MBR_1\rangle,$

$\langle CP_2, MBR_2\rangle, \cdots, \langle CP_M, MBR_m\rangle)$

其中，$\langle OI_i, MBR_i\rangle$称为数据项，$\langle OI_i\rangle$为空间目标标识，$MBR_i$是该数据项在 k 维空间中的最小外包矩形。$\langle CP_i, MBR_i\rangle$为索引项，$CP_i$ 为指向子树根节点的指针。$COUNT$ 表示节点内最大容量 M 和最小容量 m，$m=M/2$。$LEVEL\geqslant 0$ 表示该节点在树中的层数(0 表示叶节点)。

静态 R 树的生成首先采用对空间数据对象进行预处理的方法，将满足一定条件的空间对象按照某种规则聚集到一个叶子节点中。在生成叶子层之后，把每个叶子节点的最小外包矩形(MBR)作为新的空间对象，按照一定的规则生成父节点。向上递归，直至生成根节点。这类似于层次聚类的方法。本文采用 CURE 聚类算法对空间对象的 MBR 进行预处理，计算每个矩形之间的相似度，进而把矩形进行聚类分簇，首先生成叶子节点，最终生成根节点。动态 R 树可以保证空间利用率在 50%以上，利用 CURE 算法构建的 R 树可以极大地提高空间利用率和索引效率，使得节点的空间利用率接近 100%。

3 基于 CURE 的 R 树索引结构构建

传统 R 树本身具有节点的 MBR 过度重叠的不足，这会造成多路查询或者是多路插入，从而对 R 树的索引产生影响，在处理海量复杂数据信息的查询和复杂空间关系(如不确定拓扑关系)查询等问题时具有一定的局限性。本文采用 CURE 算法构建 R 树，其类似于静态 R 树的生成，而且具有静态 R 树的优点。构建过程中的要点如下：

(1)对于 R 树的 MBR，不能简单地通过计算其中心点来比较距离大小，而应该采用两个矩形中最近点的距离，并且相互重叠矩形之间的距离是 0；

(2)R 树的数据对象是通过 MBR 近似表示的，在计算中心点和代表点过程中，采用 MBR 的中心点进行计算，具体公式为：

$$M_i x=N^{-1}\sum_{i=1}^{2N}x_i, M_i y=N^{-1}\sum_{i=1}^{2N}y_i$$

(3)选取距离中心点最远的边缘点作为代表点，能够有效地捕捉簇的形状，并且利用收缩因子使处理孤立点的能力更强，聚类效果更好。

利用 CURE 算法构建 R 树的基本思想是：用 MBR 近似地表示数据点。首先确定每一个数据矩形，把它们作为只有一个数据项的个体簇。通过计算矩形之间的相似度，把逻辑上最相近的矩形整合到一个簇中。计算新簇的中心点，进而求出代表点。利用簇中的代表点计算与其他矩形之间的相似度，不断聚成新的簇，直至达到节点所能容纳的数据量。当叶子节点生成完毕之后，递归地构造中间节点，直到生成 R 树的根节点。为了描述基于 CURE 算法的 R 树构建，本节首先给出定义 1 和定义 2。

定义 1 根据层次聚类特性，把每个数据对象的 MBR 作为单独簇，采用计算最小距离的方法进行合并，则每个簇的表示为：$C_1\leftarrow(r_{1x}, r_{1y})$，$C_2\leftarrow(r_{2x}, r_{2y})$，$\cdots$，$C_i\leftarrow(r_{ix}, r_{iy})$，各个 MBR 之间的距离关系为 $dist(r_i, r_j)=(r_{ix}-r_{jx})^2+(r_{iy}-r_{jy})^2$。其中，$r_{ix}$，$r_{iy}$表示第 i 项中维度为 x 和 y 的数据对象。

定义 2 假设 r_i，r_j 分别为两个数据对象的 MBR，中心点分别为(M_{ix}, M_{iy})，(M_{jx}, M_{jy})，那么 MBR 的中心点为：

$$w.mean=\frac{|r_i|(M_{ix}, M_{iy})+|r_j|(M_{jx}, M_{jy})}{|r_i|+|r_j|}$$

由中心点可以给出代表点的计算公式：$w.rep=r_i+\alpha(w.mean-r_i)$，其中 α 为收缩因子，取值在 0.2～0.7 之间时聚类效果较好。

基于以上讨论，给出在处理较少数据量时基于 CURE 算法的 R 树构建算法 CU_RSbuilt()，如算法 1 所示。

算法 1 CU_RSbuilt(r_i)

输入：n 个数据矩形 $r_0, r_1, r_2, r_3, \cdots, r_{n-1}$；

输出：R 树头节点指针 head。

```
begin
1. 确定 r0, r1, r2, r3, ..., rn-1;/* 每个矩形作为单独簇 */
2. for each rectangle ri in DateSets do{
3.     d ← ComputerMindistance(ri, rj);/* 计算每两个矩形框之间
       的最短距离 */
```

```
4.      if (Mindist(r_i, r_j)==d){
5.      w ←merge(r_i, r_j);
6.      w. mean←Mean(M_i. M_j)/*利用矩形中心点计算新簇中心点*/
7.      w. rep ←Rep(r_i, w. mean);}/*计算新簇代表点*/
8.      else return 3;
9.      if(Node. number==RtreeMaxStrorge){/*如果节点内数据项恰好等于R树节点的最大容量*/
10.                 return 3;}
11. while(Node. number<RtreeMaxStrorge){
12.     if(ClusterRep. number≤MaxRep. number){
13.   ComputerNewClusterDistance(w. rep, r_i);/*通过计算新簇的代表点与其他矩形距离最近点来判断最近矩形*/
14.                 return 4;}
15.     else{
16.     ComputerMaxDistance(w. mean, r_i);
17.     d←MaxDistance(r_i, r_{i-1});/*计算与上一个代表点距离最远的点,将其作为新的代表点*/
18. until ClusterRep. number=MaxRep. number;
19. w. rep ←Rep(r_i, w. mean);/*利用收缩因子重新计算代表点*/
20.   return 3;}
21.                 }
22. CallTree_Creattion(child(i)→rect, n, child(i))/*递归构造中间节点。n为每个节点内的矩形数,child为每个叶子节点*/
23. return head;}
end
```

在CU_RSbuilt算法中,通过计算矩形之间的距离(ComputerMindistance(r_i,r_j))来判断矩形之间的相似度,将距离最近的矩形合并到一个簇中(merge(r_i,r_j))。计算新簇的中心点以及代表点,通过控制节点内所容纳的最大数据量(RtreeMaxStrorge)和代表点的最大容量(MaxRep. number)来提升节点的空间利用率。通过聚类生成叶子层以后,再次递归构造中间节点(CallTree_Creattion()),直至生成根节点,从而完成整个R树的构建。

当R树中的数据量过大时,CU_RSbuilt算法效率较低。为了处理该问题,进一步提出了新的改进方法:对数据对象,先取样,后划分;经过局部聚类,最后再总体聚类,完成对整个R树的构建。由此,给出定义3。

定义3 在给定的d维空间中有n个数据项,如果满足$D^k(p^1)>D^k(p)$的点p^1不超过$n-1$个,即$|\{q\in D|D^k(q)>D^k(p)\}|\leqslant n-1$,那么就称$p$为$D_n^k$异常,其中$D^k(p)$表示点$p$和它的第$k$个最近邻。

CURE算法本身处理孤立点的方法是检验簇中数据项的稀疏度,通过判断簇内数据增长的速度来处理孤立点,这种方法过于经验化,对数据集的选取也比较敏感。本文采用基于距离的方法能够更加精确地处理孤立点。基于以上讨论,进一步给出在海量数据的情况下,采用CURE构建R树的算法CU_RHbuilt。

算法2 CU_RHbuilt(r_i)

```
输入:n个数据矩形 r_0, r_1, r_2, r_3, ···, r_{n-1};
输出:输出R树头节点指针 head。
begin
1. 确定 r_0, r_1, r_2, r_3, ···, r_{n-1};/*从原始数据集中取出n个矩形样本*/
2. p←Sample_rectangle_divide(n);/*对n个数据矩形样本进行p个划分*/
3. for each rectangle in Sample n do{
4.   k←UseCure(n/p);/*用CURE算法对每一个划分的簇进行聚类*/
5.     if(k<n/pq) return 4;
6.     until n/pq ←UseCure(n/p);
7.     RectClsuter←Use_Cure(Sample)/*用Cure算法对样本集进行聚类之后得到新的矩形簇*/
8.     RectClsuter1←DetectOutlier(RectClsuter);} /*通过删除孤立点得到新的矩形簇*/
9. for each rectangle in RectClsuter1 do{/*进行二次聚类*/
10.     RectClsuter2←Use_Cure(RectClsuter1);
11.     NewRectClsuter←DetectOutlier(RectClsuter2);/* 再次删除孤立点*/
12.     Min←dist(EStorage. rect, NewRectClsuter. rep);/*计算外存中的矩形代表点与簇中矩形代表点之间的最小距离*/
13.     merge(EStorage. rect, NewRectClsuter. Centroid;/*合并外存中矩形项到与新簇中心点距离最近的簇中*/
14.                 }
15. CallTree_Creattion(child→rect, n, child);/*递归生成中间节点*/
16. return head;
end
```

在CU_RHbuilt算法中,要实现对海量数据的有效建树,首先必须对矩形样本集进行划分(Sample_rectangle_ divide())。把n个样本集划分成p个簇,则每个簇中数据量为$\frac{n}{p}$。每一个矩形作为一个单独的簇,利用CURE算法对每个簇聚类(UseCure($\frac{n}{p}$)),经过第一次聚类之后产生的新簇(RectClsuter)要进行删除孤立点(DetectOutlier())的操作。再次利用CURE算法进行第二次聚类,对产生的新簇第二次删除孤立点。最后计算外存中的点与矩形中心的距离,实现全局聚类(merge())。递归生成中间节点,完成整个R树的构建。经分区之后,可降低时间复杂度;相对于R树的动态加载(OBO),该算法可减少建树时间消耗,提升索引效率。

4 节点分裂算法

第3节给出了基于CURE算法的R树构建方法,当向R树中插入新的数据项时,如果超出了节点所能容纳的数据量,那么节点会分裂成两个。节点分裂的好坏直接影响到R树的索引效率。本节提出了一种基于CURE算法的节点分裂方法。根据聚类的特性,使空间中相近数据聚成一个簇,以保证增加的MBR最小,并且减少节点之间的MBR重叠,提高索引效率,减少磁盘访问量。进一步给出节点的分裂算法NodeSplit,如算法3所示。

算法3 NodeSplit(*DateSets*,2):

```
输入:一个n数据矩形 r_0, r_1, r_2, r_3, ···, r_{n-1};
输出:生成2个节点的MBR。
begin
1. 确定 r_0, r_1, r_2, r_3, ···, r_{n-1};/*每个矩形作为单独簇*/
2. While(DateSetsCluster. number>2){
3. for each rectangle r_i in DateSets do
4. dist(r_i, r_j)=(r_ix−r_jx)^2+(r_iy−r_jy)^2;/*r节点的i,j项和x,y维度*/
5. if(r_i==r_j. closest){/* 当r_i是r_j的最近邻时*/
6.     w=r_i∪r_j,
```

```
7.    w. mean=(|r_i|r_i. mean+|r_j|r_j. mean)/(|r_i|+|r_j|);
8.    w. rep=r_i+α*(w. mean−r_j);
9.    if(ClusterRep. number≤MaxRep. number){/*当簇中代表点
      个数少于代表点最大个数时*/
10.     ComputerMinDance(w. rep,r_i. rep);/*计算新簇中的代表点
        与其他单一簇代表点之间的距离*/
11.           return 4;}
12.     else {
13.       ComputerMaxDistance(w. rep,r_i. rep);
14.       d←MaxDistance(r_i,r_{i-1});/*将与上一个代表点距离最远
          的点作为新的代表点*/
15.         until ClusterRep. number=MaxRep. number;/*直到簇
          中代表点个数等于代表点个数的最大阈值*/
16.         w. rep=r_i+α*(w. mean−r_j);/*利用收缩因子再次求
          代表点*/
17.   return 2;}
18.             }
19.  }
20. if(PNode. number>RNode. number) {/*父节点溢出*/
21.         NodeSplit(PDateSets,2);}递归调用算法,调整父节点*/
22. return DateSetsCluster. number;
end
```

在 NodeSplit 算法当中，节点分裂之后的两个 MBR 会继承聚类的优点，簇与簇之间相似度低，从而距离较远，使矩形重叠度大幅度降低，达到提高索引效率的目的。采用 CURE 中的收缩因子，可以使数据项向簇中心靠拢，减少增加的 MBR 面积，同时降低孤立点的影响，避免一个数据项单独成为一个节点的情况，提高了节点的空间利用率。

5 实验结果与分析

本文详细研究了基于 CURE 算法的 R 树构建过程，并分别给出了在处理较小数据量和处理复杂海量数据时 R 树的构建算法：CU_RSbuilt()、CU_RHbuilt()。本节在配置 2GHz Pentium® T4200 CPU 和 2GB 内存的 PC 机上进行实验。操作系统为 Windows 7，程序用 C++、R 语言编写。

为了验证 CU_RSbuilt()算法的有效性，采用传统的 OBO 方法构建 R 树，通过插入二维空间的 100 个随机点生成 R 树。树高度为 3，节点最小容纳量为 4，最大容纳量为 8。从图 1 中可以看到，节点之间有很明显的重叠，会造成多路的插入和查找，对查询效率有很大的影响；而且当插入数据项的空间位置与 MBR 中心距离较远时，会造成 MBR 过大，从而使节点内的空间利用率降低，影响数据项的存储。

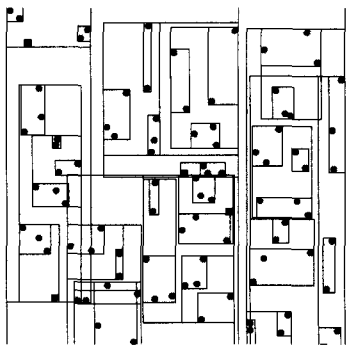

图 1 采用 OBO 算法构建 R 树

本节选取二维随机 spiral 数据 1000 组，其中前 500 组数据属性标示为 0，后 500 组数据标示为 1。采用取样和划分技术，编写 R 语言程序，利用 CURE 算法的数据集聚类。如 CURE 算法聚类之后呈树形结构，由子节点生成父节点，最后生成根节点。从图 2 中可知，随着树的高度不断增加，中间节点之间的距离会越来越大，相似度较低，从而达到节点不重叠的理想聚类效果。相比于图 1，由 CURE 算法构建的 R 树，可以使节点之间的 MBR 不重叠，从而提升了索引效率。在数据量比较大的情况下，通过取样和划分，有效地降低了算法的时间复杂度，使程序得以运行。另外选取 Iris 数据集进行实验，该数据集包含 4 个属性，共有 150 个数据对象，可以分为 3 类，3 种花的类别标示分别为：virginica，versicolor，setosa。将数据集投影到二维空间进行 CURE 算法聚类。如图 3 所示，随着簇的不断合并，树的层数不断增加，中间节点之间的相似度会越来越低。图中避免了一个节点下只有一个数据项的不足，通过控制代表点数量，进而使 R 树节点内数据存储达到最大，利用收缩因子使数据更加紧凑，有效地提高了节点的空间利用率。

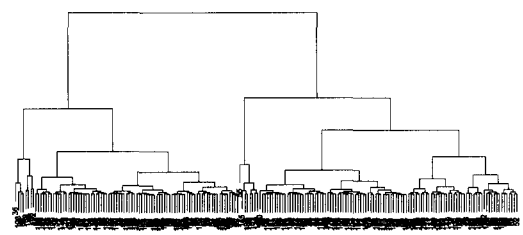

图 2 利用 spiral 数据集进行 CURE 算法聚类

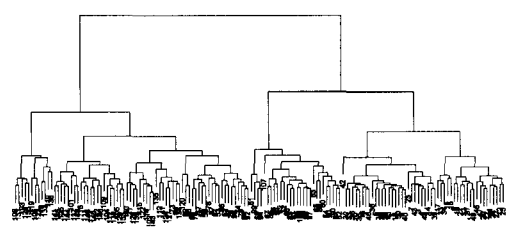

图 3 利用鸢尾花数据集进行 CURE 算法聚类

在图 4 中，采用 K-means 方法通过 R 语言实验了一个简单的数据聚类。与图 2 进行比较可知，在 R 树构建的过程中，图 2 的矩形重叠度会明显低于采用 K-means 方法的建树；而且图 4 部分中心点过于相近，利用 K-means 生成 R 树的 MBR 仍然会重叠，造成额外的查询时间消耗。由此可见，采用 CURE 构建 R 树，可以有效弥补利用 K-means 构建 R 树时的初始值选择和对孤立点处理上的不足，大幅度降低矩阵重叠，有效提升查询效率，提高空间利用率。分别选取两种比较有代表性的 R 树数据结构：Guttman's R 树和文献[11]提出的基于 k-medoids 算法构建的 R 树，将它们与本文所提出的基于 CURE 算法构建的 R 树进行比较分析。基于模拟数据集实验，选取的数据集是利用随机函数生成的从 5000 到 30000 个整型数据。从图 5 中建树的时间对比可以看出，利用 CURE 算法的建树时间消耗略低少基于 k-medoids 算法，明显少于传统的 OBO 算法。但是由于传统 R 树的节点 MBR 过度重叠，k-medoids 算法对 k 值的选择和处理大型数据时存在不足，基于 CURE 算法的 R 树查询效率要高于传统 R-树和基于 k-medoids 算法的 R 树。

从实验结果中可以明显看出，CURE 算法在处理数据集的过程中，簇与簇之间层次清楚，在 R 树生成的过程中，矩形基本不重叠，而且空间利用率较高，每一个簇中节点容纳的数据量可以控制到最大值，空间利用率接近 100%。分析实验结果可知，数据量越大，产生的树则越高，在查询叶子节点时，时间上的消耗会相对较大，磁盘访问次数也会相对较多。采用 CURE 中的划分技术，可以降低算法的时间复杂度，有效地处理海量数据集。

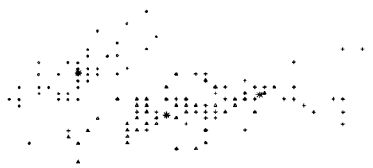

图 4　采用 K-Means 进行聚类

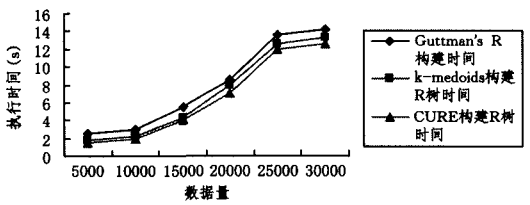

图 5　基于模拟数据集的 R 树构建时间对比

结束语　为了提升对海量空间数据信息和复杂空间关系查询的能力，本文针对传统 R 树建树的不足，提出了基于 CURE 算法的 R 树构建和节点分裂算法。利用聚类的特性，使 R 树节点的 MBR 几乎不重叠，提高了查询效率。根据 CURE 算法对孤立点处理具有健壮性的优点，提高了 R 树节点的空间利用率，使每一个节点存储的数据量达到可容纳的最大值。

基于 CURE 算法的 R 树不但可以处理任意形状的簇，而且能够更加有效地处理孤立点。在构建 R 树的过程中，采用 3 步聚类，最后使外存中的标记的数据项导入内存，使得中间叶子节点满足聚类特性，减少了 MBR 的重合，提高了插入和索引效率。进一步提出了基于 CURE 算法的 R 树节点分裂方法，通过使节点内相似度较高的数据项尽可能地整合到一个节点中，有效降低了增加的 MBR 面积；同时利用收缩因子有效减少了孤立点的影响，提高了节点的空间利用率。

未来的研究重点是基于本文所提方法对复杂空间数据查询技术（如连续近邻链查询[15]）和多维不确定空间拓扑关系查询技术进行研究。

参 考 文 献

[1] 李松，张丽平，孙冬璞. 空间关系查询与分析[M]. 哈尔滨：哈尔滨工业大学出版社，2011

[2] 张明波，陆峰，申排伟. R 树家族的演变与发展[J]. 计算机学报，2005，28(3)：289-300
Zhang Ming-bo, Lu Feng, Shen Pai-wei. The Evolvement and Progress of R-Tree Family[J]. Chinese Joural of Computers, 2005, 28(3): 289-300

[3] Guttman A. R-Trees A Dynamic Index Structure for Spatial Searching[C]//Proceedings of Annual Metting(SIGMOD'84). 1984: 18-21

[4] 李松，郝忠孝. 球面上最近邻空间关系处理方法[J]. 计算机工程，2010，36(6)：91-93
Li Song, Hao Zhong-xiao. Methods For Handing Nearest Neighbor Spatial Relation on Spherical Surface[J]. Computer Engineering, 2010, 36(6): 91-93

[5] 张栋良，唐俊. 基于路由机制的时变路网 k 近邻算法[J]. 计算机科学，2013，40(2)：30-34
Zhang Dong-liang, Tang Jun. k-Nearest Neighbor Algorithm in Dynamic Road Network Based on Routing Mechanism[J]. Computer Science, 2013, 40(2): 30-34

[6] 张丽平，李松，赵纪桥，等. 受限区域内的单纯型连续近邻链查询方法[J]. 计算机应用，2014，34(2)：406-410
Zhang Li-ping, Li Song, Zhao Ji-qiao, et al. Simple continuous near neighbor chain query in constrained regions[J]. Journal of Computer Applications, 2014, 34(2): 406-410

[7] Abdel R, Zaki M, Jizhe X, et al. Data-intensive Spatial Indexing on the Clouds[J]. Procedia Computer Science, 2013(18): 2615-2618

[8] Brakatsoulas S, Pfoser D, Theod Y. Revisiting R-tree Construction Principles[C]//Proceedings of the 6th East European Conferences on Advances in Databases and Information Systems. London: Springer Verlag, 2002: 149-162

[9] 黄继先，鲍光淑，夏斌. 基于混合聚类算法的动态 R-树[J]. 中南大学学报，2005，37(2)：366-370
Huang Ji-xian, Bao Guang-shu, Xia Bin. A dynamic R-tree index based on hybrid clustering algorithm[J]. J. Cent. South Univ, 2005, 37(2): 366-370

[10] 刘润涛，郝忠孝. R-树和四叉树的空间索引结构：RQQP_树[J]. 哈尔滨工业大学学报，2010，42(2)：323-327
Liu Run-tao, Hao Zhong-xiao. Spatial index structure based on R-tree and quadtree: RQQP_tree[J]. Journal of Harbin Institute of Technology, 2010, 42(2): 323-327

[11] 汪璟玢. 一种结合空间聚类算法的 R 树优化算法[J]. 计算机工程与应用，2014，50(5)：112-115
Wang Jing-bin. Optimization algorithm for R-tree combining with spatial-clusting[J]. Computer Engineering and Application, 2014, 50(5): 112-115

[12] 刘润涛，安晓华，高晓爽. 一种基于 R-树的空间索引结构[J]. 计算机工程，2009，35(23)：32-34
Liu Run-tao, An Xiao-hua, Gao Xiao-shuang. Spatial Index Struction Based on R-tree[J]. Computer Engineering, 2009, 35(23): 32-34

[13] Jesper J W. Constructing the R* Consensus Tree of Two Trees in Subcubic Time[J]. Algorithmica, 2013(66): 329-345

[14] Nikhil D K. P+ and R* Tree Indexing Techniques on Data Warehouses for Efficient Database Indexing[J]. International Journal of Engineering Associates, 2013, 2(5): 31-34

[15] 李松，张丽平，朱德龙，等. 动态受限区域内的单纯型连续近邻链查询方法[J]. 计算机科学，2014，41(6)：136-141
Li Song, Zhang Li-ping, Zhu De-long, et al. Simple Continues Near Neighbor Chain Query in Dynamic Constrained Regions[J]. Computer Science, 2014, 41(6): 136-141