高技术通讯 2010 年 第 20 卷 第 1 期:55 ~ 60

doi:10.3772/j.issn.1002-0470.2010.01.010

一种基于形状分析的 R 树节点分裂算法①

刘 焱***② 方金云* 韩承德*

(*中国科学院计算技术研究所集成应用中心地理信息事业部 北京 100190)

(**中国科学院研究生院 北京 100049)

摘 要 基于对最小边界矩形(MBR)形状的分析,提出了一种线性时间复杂度的 R 树空间索引节点分裂算法。该算法将节点及其记录的最小边界矩形按形状分类,并根据分类情况确定节点分裂策略。首先提出了一种基于形状分析的基本节点分裂算法,然后针对其可能产生的不平衡分裂结果,提出了一种分裂结果平衡算法。最后提出了一种考虑兄弟节点的联合分裂策略以进一步提升算法的效果。对比实验表明,无论在索引的创建过程还是查询过程,此算法效率都优于对比算法,并且具有易实现和适应性强等特点,可以应用于各种空间数据库和地理信息系统(GIS)。

关键词 地理信息系统(GIS),空间数据库,空间索引,R 树,节点分裂算法

0 引 言

随着空间数据在地理信息系统(GIS)和计算机辅助设计等领域的应用不断增强,对空间数据存储与管理的研究越来越受到关注,比如传统的空间数据库方法和近年来兴起的基于地理标记语言(geographical markup language, GML)的方法[1]。无论哪种方法,其研究的主要内容都不外乎多维数据表示和访问方法。多维数据索引是多维数据访问方法的重要研究内容,已有的索引包括网格文件[2],k-D-B 树[3],R 树[4],四叉树[5],Cell 树[6]等。其中 R 树由于其性能优势成为研究最为活跃的索引之一。

R 树最早由 Guttman 在文献[4]中提出,是 B 树在高维上的扩展。R 树由节点组成,每个节点包含一定数量的记录。每个节点上的记录个数有上限和下限。由于上限的存在,向一个满的 R 树节点中插入新的记录时需要使用节点分裂算法将该结点分裂,从而产生空位用以插入新记录。节点分裂算法是 R 树的核心算法[4]。分裂算法的好坏直接关系到最终生成的 R 树结构,从而影响空间查询的效率。因此 R 树节点分裂算法成为 R 树研究中的一个热点。

Guttman 在文献[4]中提出了三种 R 树节点分裂算法,分别为线性时间分裂算法、平方时间分裂算法和指数时间分裂算法(下文中分别称之为线性方法、平方方法和指数方法)。其中,指数方法的时间复杂度高达 $O(2^N)$,实际应用中不被采用。而线性方法则由于产生的 R 树效率不高,也没能被广泛采用。只有平方方法得到了较多的应用。文献[7]提出了改进的 R 树——R* 树,其中包括一种新的 R 树节点分裂算法。但这种算法计算量大,效率不高,索引建立时间长。文献[8]中提出了一种新的线性节点分裂算法(下文简称新线性算法),作者将节点中的记录按其外包距离节点外包的远近分类,再按此分类选择分裂轴和进行分裂。文献[9]证明了 Guttman 提出的指数方法可以在多项式时间复杂度内实现并由此提出了一种最优化分裂算法。该文作者又基于此进行了改进,提出了一种分裂算法的实现框架。近年来,国内也有学者进行了该方面的研究。文献[10]和[11]都采用了聚类的思想,将节点分裂过程看作是记录根据其空间位置聚类的过程。本文提出了一种基于形状分析的由基本算法、平衡算法和联合分裂算法组成的 R 树节点分裂算法。已有算法大都只考虑了索引的创建效率和查询效率中的一个方面,但是当前空间数据量不断增加,因而对这两个方面的要求都不断提高。而本文同时考虑了这两个方面,提出了一种基于形状分析,由基本算法、平衡

① 863 计划(2009AA12Z226)资助项目。

② 男,1982 年生,博士生;研究方向:空间数据库;联系人,E-mail: liuyan@ict.ac.cn

(收稿日期:2009-06-01)

算法和联合分裂算法组成的R树节点分裂算法,得到了不错的结果。

1 基本算法

R树产生以来,除了众多实验方面的分析之外,文献[12]和[13]中提出了R树查询效率的理论模型,如式

$$DA(q) = \sum_{j}\{\prod_{i=0}^{n}(s_{i,j} + q_i)\} \quad (1)$$

所示。其中,q 表示查询外包,$DA(q)$ 表示使用 q 查询时的磁盘访问次数,$s_{i,j}$ 表示第 j 个记录外包的第 i 维上的长度,而 q_i 表示查询外包第 i 维上的长度。由式(1)可知,除去查询外包的因素,查询的磁盘访问次数与面积和周长两个参数相关。

本文描述的R树节点分裂算法的主要思想是使得分裂所产生的节点尽量成正方形,以便使其周长之和较小,同时在分配记录时尽量使其面积增加最小。为方便论述,本文只讨论二维的情况。二维是最为常见的一种情况,另外该算法也能很自然地扩展到高维的情况。

为了便于下文论述,首先介绍j-Long的概念。这一概念用于描述矩形的形状。一个多维空间中的矩形 R,设其最长维度 j 上的长度为 L,其最短维度 i 上的长度为 l,选定一个大于1的常数 θ,如果 $L/l > \theta$,则称该矩形是j-Long的。

该算法的第一步是判断上溢节点的最小边界矩形(minimum bounding rectangle,MBR)是否为j-Long。如果是,则选定维度 j 为分裂轴,对该节点的分裂将垂直于该维度进行。判断MBR是否为j-Long的过程同时也求出了维度 j 本身。如果MBR不是j-Long,则需要进一步检查该节点中每一个记录外包的形状。在遍历节点记录的过程中,记下维度0长度大于维度1长度的记录个数和维度1长度大于维度0的记录个数,分别记为0-*count*和1-*count*。如果0-*count* > 1-*count*,则选定维度1为分裂轴,反之选择维度0为分裂轴。在求出分裂轴之后,垂直于分裂轴把外包等分为两个矩形,分别记为mbr1和mbr2,然后扫描每个记录,如果某记录完全落入mbr1或mbr2,则把它归入相应的新节点;如果它没有完全落入任何一个外包,则把它加入使得面积增加较小的那个节点;如果面积增加相等,则它加入引起的重叠面积小的新节点;如果重叠面积也相等,则把这加入记录个数少的新节点。此步骤完成之后,所有的记录都被分配到两个新节点之一,算法结束。

本算法中,分配记录的操作需要遍历节点的所有记录,故时间复杂度为 $O(N)$,其它操作时间复杂度均为 $O(1)$。因此该算法的总时间复杂度为 $O(N)$。

2 平衡算法

基本算法在运行过程中会产生分裂不平衡的状况,即会产生记录个数小于下界的节点。具体来说,当一个节点的记录分布不均匀,大部分记录聚集在某个角落时,对该节点使用此方法分裂将会产生不平衡的结果。为了解决此种情况,笔者为此设计了一个平衡算法,用来平衡分裂结果。

平衡算法首先尝试使用另外一个维度作为分裂轴进行节点分裂,如果分裂成功并产生了平衡的分裂结果,则将此结果作为最终的分裂结果。如果此种分裂也会产生不平衡的结果,则针对第一个分裂结果进行额外处理,即首先计算需要移动的记录个数,然后在记录较多的节点中选择距离另一个节点最近的记录,将其转移到另一个节点中。

本算法中,在一个节点中选择距离另一个节点最近的若干记录这一操作已被证明能够在线性时间内完成[14]。故整个算法的时间复杂度为 $O(N)$。

3 联合分裂算法

为了进一步优化分裂效果,笔者提出了一个联合分裂算法。即在分裂过程中选择合适的兄弟节点一起参与分裂。这样做的好处一是可以提高存储利用率,二是可以使得分裂后的节点更接近正方形。大量实验表明,如果待分裂节点的外包形状接近正方形,则较难产生比较好的结果。原因是不论选择哪个轴作为分裂轴,正方形分裂后产生的节点外包都不会接近正方形。故对于外包接近正方形的节点,将其记录与其兄弟节点的记录放在一起,然后统一分裂。然而也并不是所有的节点都适合与兄弟节点联合分裂。当待分裂节点与选定的兄弟节点位于对角线时,分裂产生的新节点与其它节点的重叠面积会非常大,从而破坏了整个R树的结构。因此需要在分裂之前检测这种情况。具体来说,算法首先检测最优点是否适合单独分裂,并检查最优和次优节点联合分裂是否会产生不良结果。对不满足条件的结点不进行联合分裂,而是采用基本算法进行分

裂。对于适合联合分裂的最优和次优节点,垂直于其总外包的最长轴将其总外包分割为 3 个矩形,再按基本算法中的方法向 3 个新节点中分派记录。

该算法类似于基本算法,只有分配记录的操作时间复杂度为 $O(N)$,其它操作都是 $O(1)$,故该算法总时间杂度也是 $O(N)$。

4　实验结果及分析

为了验证本文算法的效率,笔者设计了一系列实验。对比算法有 Guttman 的平方算法和新线性算法。实验在一台奔腾 4 计算机上运行,CPU 主频为 1.7GHz,内存为 1G 字节,操作系统为 Linux。实验中使用的数据既包括人工合成数据又包括实际数据,例如:

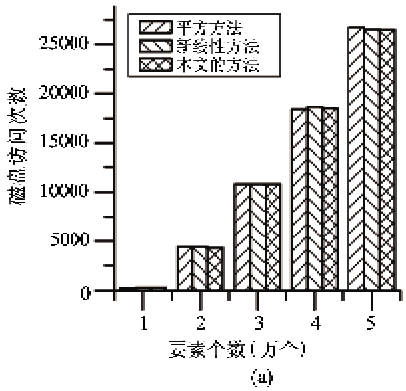
(1) 10 个人工合成正态分布的数据集,数据量从 1 万依次递增到 10 万。

(2) 从上海道路数据中提取的 10 个数据集,数据量从 1 万依次递增到 10 万。

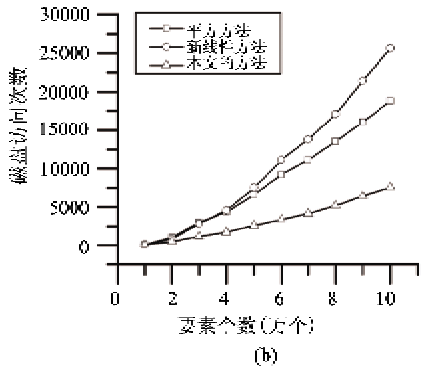
(3) 从 TIGER 数据库中美国长滩道路数据提取的 5 个数据集,数据量从 1 万依次递增到 5 万。

实验分为 3 部分,第一部分用来测试 R 树的创建效率,后两部分用来测试 R 树的窗口查询和最近邻查询效率。为了真实地反映实际应用环境,实验中采用了 LRU 节点缓存策略。

R 树创建效率测试在合成数据和真实数据上分别进行,其结果如图 1 所示。图 1(a)为对人工生成的正态分布数据建立 R 树时三种算法产生的磁盘访问次数对比,图 1(b)为对真实数据建立 R 树时三种算法产生的磁盘访问次数。



(a) 使用人工合成的正态分布数据集(为了描述问题清楚本图没有像其它图一样采用点-线图,而采用了柱状图)



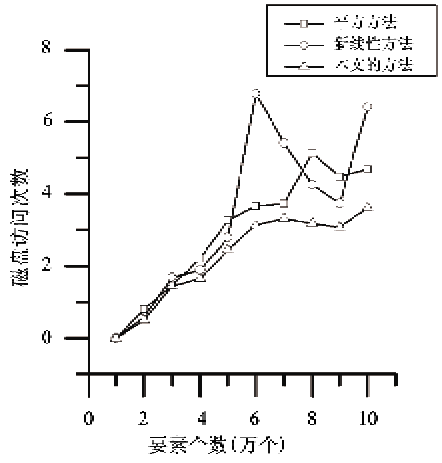
(b) 使用真实数据集

图 1　R 树创建效率对比

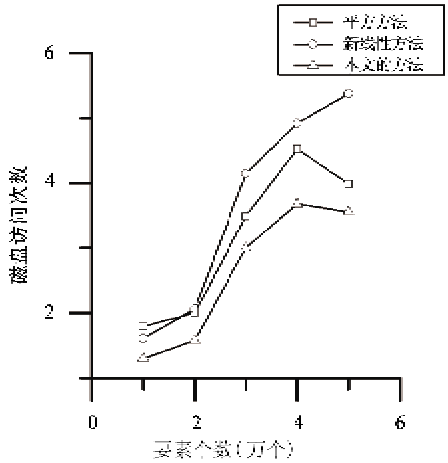
由于正态分布数据规律性强,分布均匀,三种算法在建立 R 树时所产生的磁盘访问次数都差不太多,本文算法在此项对比中虽有优势但优势不大。而对实际数据建立 R 树时本文算法效率上的优势则很明显。可以看出,在建立 R 树的过程中,相对于其它两种算法,采用本文中的算法可以大大减少磁盘访问次数。

第二部分实验用来验证本文算法所产生的 R 树的窗口查询效率。查询矩形的位置在数据集的外包范围内随机选定,其面积与总外包的比例从 1%,0.1%,0.01%,0.001%四个值中随机选取。针对每个数据集生成 100 个这种查询矩形,记录 100 次查询的平均磁盘访问效率。结果如图 2 所示。图 2(a)是由正态分布数据生成的 R 树查询效率测试,可以看到本文算法效率优于其它两个算法。在数据量小时三种算法生成的 R 树效率相差不大,但是随着数据增大,本文中算法的优势越来越明显。图 2(b)是对真实数据生成的 R 树进行查询的效率。可以看出,本文算法所生成的 R 树的查询效率明显优于其它两个算法。

第三部分实验是利用三种算法生成的 R 树进行最近邻查询的效率测试。针对每个数据集生成 100 个查询点,其位置在数据集外包范围内正随机选取。使用这些查询点对每个 R 树进行 100 次最近邻查询,计算其平均磁盘访问效率。其结果如图 3 和图 4 所示。图 3 描述了查询不同最近邻个数时磁盘访问次数与数据量的关系,全部使用真实数据测试。图 4 在不同数据类型和数据量情况下要求查询

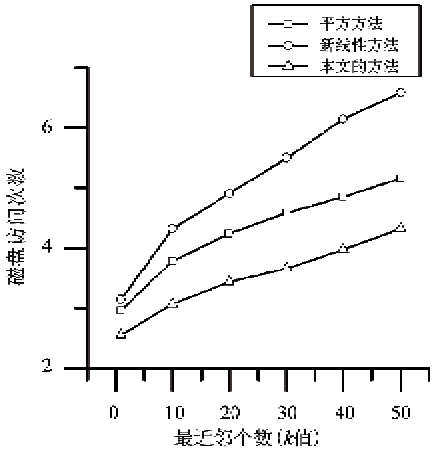


(a) 使用人工合成的正态分布数据集

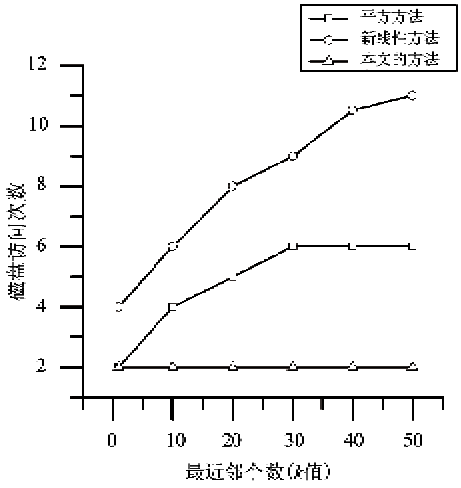


(b) 使用真实数据集

图 2 R 树窗口查询效率对比

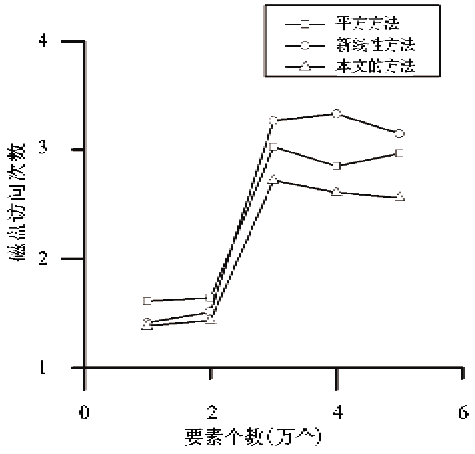


(a) 真实数据,要素个数 5 万

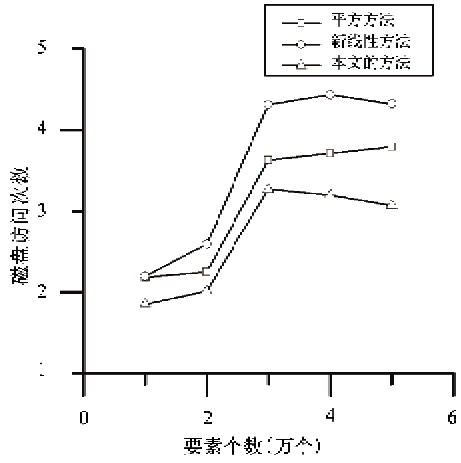


(b) 人工合成数据,要素个数 10 万

图 3 磁盘访问次数与最近邻个数的关系

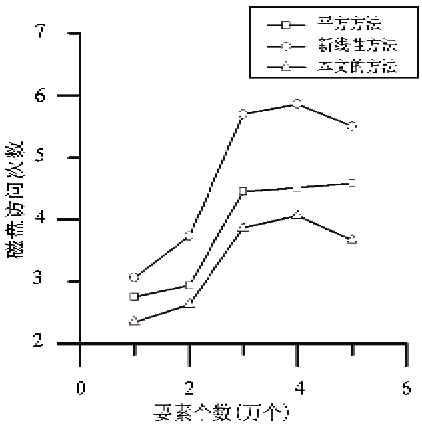


(a) 真实数据, $k = 1$

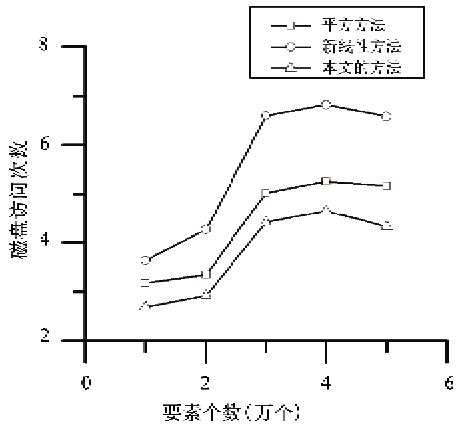


(b) 真实数据, $k = 10$

图 4 最近邻查询效率



(c) 真实数据,$k=30$



(d) 真实数据,$k=50$

图 4　最近邻查询效率

的最近邻个数与磁盘访问次数的关系,分别使用真实数据和合成数据测试。从图 3 和图 4 可以看出,采用本文方法所建立的 R 树在最近邻查询效率上都优于对比算法所建立的 R 树。

其中图 4(b)中,由于处理的数据是正态分布,本文的算法所建立的 R 树具有非常好的局部性,在 k 值小于节点的记录个数时,磁盘访问次数并没有随 k 值的增加而增大。

5　结　论

本文在对已有分裂算法充分调研的基础上,提出了一种新的线性时间复杂度的 R 树节点分裂算法。此算法由基本算法、平衡算法和联合分裂算法三部分组成。基本算法通过外包形状分析分裂溢出结点,平衡算法解决分裂不平衡的情况,而联合分裂算法可以用于在某些特定情况下优化分裂结果。在实际应用中可根据不同的情况选择不同的策略。通过使用合成数据和真实数据进行实验证明,该算法在 R 树创建效率和 R 树查询效率两个方面都优于对比算法,并且对不同类型的数据适应性较强,其实现也简单灵活,可以应用于各种实际系统。

尽管该算法展现出众多优势,很多工作仍然需要继续进行。对非平衡的分裂结果的处理还可以继续优化,可将空间聚类的思想纳入其中,将离群点找出后作单独处理。针对空间联接的效率分析也有待进行,这种查询与窗口查询和最近邻查询一样,是现代空间数据库的重要组成,近年来得到广泛关注。高维数据与两维或三维的情况有很大的区别,针对高维数据的效率分析也需要进行,以便验证该算法是否对高维数据同样有较好的效果。

参考文献

[1] 闫杰,方金云,韩承德等.GML 即时查询引擎研究与实现.高技术通讯,2008,19(11):1154-1160

[2] Nievergelt J, Hinterberger H, Sevcik K C. The grid file: An adaptable, symmetric multikey file structure. *ACM Trans Database Syst*, 1984, 9(1): 38-71

[3] Robinson J T. The K-D-B-tree: a search structure for large multidimensional dynamic indexes. In: Proceedings of the 1981 ACM SIGMOD international conference on Management of data. New York, NY, USA: ACM, 1981. 10-18

[4] Guttman A. R-trees: a dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD international conference on Management of data. New York, NY, USA: ACM, 1984. 47-57

[5] Samet H. The quadtree and related hierarchical data structures. *ACM Comput Surv*, 1984, 16(2): 187-260

[6] Günther O. The design of the cell tree: An object-oriented index structure for geometric databases. In: Proceedings of the 5th International Conference on Data Engineering. Washington, DC, USA: IEEE Computer Society, 1989. 598-605

[7] Beckmann N, Kriegel H P, Schneider R, et al. The R*-tree: an efficient and robust access method for points and rectangles. In: Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data. New York, NY, USA: ACM, 1990. 322-331

[8] Ang C H, Tan T C. New linear node splitting algorithm for R-trees. In: Proceedings of the 5th International Symposium on Advances in Spatial Databases. London, UK: Springer-Verlag, 1997. 339-349

[9] Garcia Y J, Lopez M A, Leutenegger S T. On optimal node splitting for R-trees. In: Proceedings of the 24rd International Conference on Very Large Data Bases. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 1998. 334-344

[10] 吴敏君,陈天滋.基于分割聚类技术的 R 树结点分裂方案.计算机应用与软件,2007,24(10):42-43, 55

[11] 黄继先,鲍光淑,夏斌.基于混合聚类算法的动态 R 树.中南大学学报(自然科学版),37(2):366-370

[12] Kamel I, Faloutsos C. On packing R-trees. In: Proceedings of the 2nd International Conference on Information and Knowledge Management. New York, NY, USA: ACM, 1993. 490-499

[13] Pagel B U, Six H W, Toben H, et al. Towards an analysis of range query performance in spatial data structures. In: Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. New York, NY, USA: ACM, 1993. 214-221

[14] Blum M, Floyd W, Pratt R, et al. Time Bounds for Selection. *Journal of Computer and System Science*, 1972, 7(4): 448-46

An R-tree node splitting algorithm based on shape analysis

Liu Yan[*,**], Fang Jinyun[*], Han Chengde[*]

([*] Geographic Information Department of Integration Application Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] Graduate University of Chinese Academy of Sciences, Beijing 100049)

Abstract

The paper proposes an R-tree node splitting algorithm based on the analysis of shapes of minimum bounding rectangles (MBRs). The algorithm classifies MBRs of nodes and records, and decides the splitting policy based on their shapes. At first a shape-based basic partition algorithm is described. Then to deal with possible uneven splitting results, a balancing algorithm is proposed. At last a joint splitting algorithm is developed using siblings to improve the splitting results. The comparison experiments showed that the proposed algorithm outperformed the comparison algorithms in both the creation process and the query process. Besides, the algorithm is easy to implement and adaptable to various types of data. It can be used in many kinds of spatial database systems and geographical information systems (GIS).

Key words: geographical information systems (GIS), spatial database, spatial index, R-tree, node splitting algorithm