基于 R+树的地图叠加分析双重循环算法

董 鹏 杨崇俊 刘冬林 芮小平
(中国科学院遥感应用研究所,北京 100101)

摘 要 地图叠加是非常重要的 GIS 空间分析功能之一,为此,提出了一种新的基于 R+树空间索引的矢量地图叠加分析双重循环算法,首先采用多边形穷举求交方法计算出线段相交点;然后运用引入、引出交点交替配对的叠加结果弧线段生成原则,进一步实现了面面叠加和线面叠加的双重循环算法;最后引入 R+树空间索引对空间数据的高效存取机制,对算法进行改进,进一步提高了计算速度. 实践结果表明,该算法快速、有效,具有较强的应用价值.

关键词 地理信息系统(420·3040) 空间叠加分析 多边形求交 空间索引 R+树
中图法分类号: TP391.4 P208 **文献标识码**: A **文章编号**: 1006 8961(2003)06-0703-08

A Dual Loop Map Overlay Algorithm Based on R+ Tree

DONG Peng, YANG Chong-jun, LIU Dong-lin, RUI Xiao-ping
(*Insititute of Remote Sensing Applications, CAS, Beijing* 100101)

Abstract Map Overlay is an important spatial analysis in Geographic Information Systems. This paper presents a novel dual-loop algorithm based on R+ tree to process vector map overlay analysis. The paper first figures out intersection points of line segments with enumerating method of polygon intersection, then uses the matching principle of "import-export intersection point" to form the result arcs and further realize the dual-loop algorithm of polygon-on-polygon and line-on-polygon vector overlay analysis; finally, perfects the algorithm through an efficient spatial access method—R+ tree index and it's efficient query and filter mechanism to spatial data, so excludes large amounts of unused graphics elements and advances the computational speed very much. This algorithm has been applied to the software development of the commercial geographic information systems. Performance tests confirm that the R+tree map overlay algorithm is quick and efficient, and has powerful practical merits.

Keywords Geographic information system, Spatial overlay analysis, Polygon intersection, Spatial index, R+ tree

0 引 言

空间叠加分析是将两层或多层地图要素进行叠加产生一个新要素层的操作,其结果是将原来要素分割生成新的要素,新要素综合了原来两层或多层要素所具有的属性[1]. 利用叠加分析,可以帮助我们进行专题分析,如将跨空间连续变化的雨量信息的降雨专题图与行政区划图叠加,可以形成各行政区划内的平均或最大降雨量图.

依据数据结构,有两种不同类型的叠加操作:栅格叠加和矢量叠加[2]. 栅格叠加是一个简单的在逐个地图像素或网格单元上的布尔算术操作,它比矢量叠加要相对容易和快捷. 矢量叠加是一个相当复杂的处理过程,是将多个拓扑关系按一定的叠加运算叠加在一起,生成新的拓扑关系,产生新的拓扑数据文件;它既涉及几何和属性的合并或交互作用,还包括相关的拓扑重构. 按数据类型[2],矢量叠加一般有点与多边形叠加、多边形与多边形叠加和线与多边形叠加 3 类. 一个矢量地图叠加操作的输入包括两个具有拓扑结构的图层,输出结果是一个新的图层,在这个图层中,基于输入图层产生了新的要

基金项目:国家自然科学基金(40071065);国际科技合作重点项目(ANFAS)
收稿日期:2002-09-03;改回日期:2003-01-27

素.矢量地图叠加的计算通常需要 3 个步骤[3]:一是计算来自于不同图层的边界(线段)之间的相交点;二、三步是进行拓扑关系的重构和属性值的分配.

1 地图叠加分析基础

1.1 点与多边形叠加

对于点(图层)与面(图层)的叠加,可以把此类问题归结为判断点图元是否位于面图元内部的问题,那么给定多边形 P 以及点 q,确定 P 是否包含 q 或者 q 是否在 P 内的求解方法是:先判断点 q 是否位于 P 的最小外接或包围矩形(Minimum Bounding Rectangle,简称 MBR)之内,如果不在,则可以判定 q 不在 P 内;如果在,则需要进一步判断.过点 q 作水平(或垂直)射线 l,通过计算 l 与 P 的边界线段的交点个数,判断其奇偶性,就可以判定点 q 是否在 P 的内部.具体地说,交点数为奇(偶)数时,点 q 在 P 的内(外)部[4].这种方法的优点是计算简单,缺点是当多边形的某条边与射线重合,或者射线正好通过多边形边界线段的端点时,就需要一些附加的判断.另外,对于带有岛(或洞)的复杂多边形的判断,需要依此方法逐环进行综合判断,当然还有其他一些方法可以处理点包含问题(即多边形 P 是否包含点 q)[5].

1.2 多边形与多边形叠加

也称多边形叠加.可以把此类问题归结为多边形(面图元)与多边形(面图元)的交、并和差的问题.

平面多边形可以看成是平面上线段的集合,判定两个多边形是否相交的问题可以转换为判断两个线段集合中的线段是否相交的问题[5].当通过某种方法算出两个多边形的全部交点后,下一步的计算就是截取其中一个多边形位于另一个多边形之内的部分.如图 1 所示,多边形 A 和多边形 B(二者都是带有内环的复杂多边形,内、外环方向相反),称 A 为叠加多边形,B 为被叠加多边形.由图 1 可以发现,A 与 B 的交点可以分为两种:一种是叠加多边形 A 的边界沿其走向经此交点进入被叠加多边形 B,把此类交点称为“引入交点”;另一种是 A 的边界经此点走出(离开) B 内部,称此类交点为“引出交点”.另外,从图 1 中还可以看到,作为叠加结果图元(图 1 中两个横纹区域)边界的一部分,A 的边界上位于 B 内部的部分弧段,都是始于“引入交点”,止于“引出交点”,并且这些“引入交点”和“引出交点”都是交替成对出现的.每一对交点之间的部分就是叠加图元位于被叠加图元之内的应予以截取的部分.据此就得到计算多边形叠加(交)的一般方法:

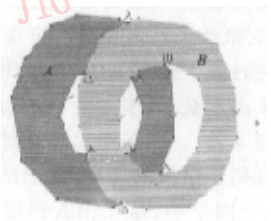

图 1 多边形与多边形的交

(1) 按一定方向记录多边形顶点坐标串(内、外环方向相反);

(2) 计算出全部交点,并识别“引入交点”和“引出交点”;

(3) 从叠加多边形边界中提取始于“引入交点”而止于“引出交点”的边界;

(4) 从被叠加多边形边界中提取始于“引出交点”而止于“引入交点”的边界;

(5)根据交点之间的连接关系,形成闭合多边形即为叠加结果多边形.

对于没有相交点的包含或被包含情况,处理方法比较简单,可通过两图元的最小包围矩形 MBR 之间的包含与被包含关系,以及顶点是否在多边形内部进行判断.

关于叠加分析中的多边形求交问题,在一些中外文献中提出过一些方法,如梯形交组成方法[5]、穷举方法[5]、平面扫描方法[6]、栅格化方法[7]等,这些方法考虑的仅仅是如何快速进行线段相交的判断,并求出交点,而没有很好地考虑求交之后,交点之间叠加结果弧段的判定以及拓扑连接重组,直到生成新的闭合多边形.为便于后续叠加操作的完成,特别是针对复杂多边形的求交问题,选择采用穷举方法进行多边形求交,并结合前面提到的引入、引出交点配对原则,进一步提出了矢量地图叠加分析的双重循环算法.

所谓穷举求交,就是依次取一个多边形的每条边(直线段)分别与另一个多边形的所有边(直线段)进行相交判断,如果相交,则求出交点.这是一个双重循环的过程.

2 地图叠加分析双重循环算法

下面以一个例子来较为详细地描述双重循环叠

加分析算法的程序(C++)实现.

图 2 中有两个多边形 A 和 B,对这两个都带有岛(洞)的复杂多边形进行叠加交操作.图中给每个多边形的边界点加上点序(索引)号,从首点为 0 开始,并用箭头表示环的方向.其中多边形 A 有 1 个外环(顺时针方向),2 个内环(逆时针).多边形 B 有 1 个外环(顺时针方向)和一个内环(逆时针方向).A 共有 41 个顶点(包含每个环的首尾重合点),B 共有 37 个顶点.其中 A 所有的环首尾重合点为 0(20)、21(31)、32(41),B 的环首尾重合点为0(23)、24(37).

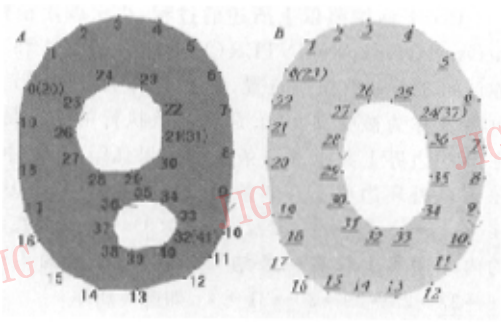

图 2　两个复杂多边形 A 和 B

(1) 首先分别从 2 个输入图层中获取面图元 A 和 B 的点集、环的个数、每个环的点数.

(2) 采用双重循环的方法,其中外层循环是从顶点 0 和 1 所表示的首线段开始,依次从 A 的边界中取直线段,如果 A 的边界线段完全位于 B 图元的 MBR 之外(可通过该线段的 MBR 与 B 图元的 MBR 的位置关系进行判断),说明该线段不与 B 图元的所有边界线段相交,则结束本次循环,执行下一次循环;内层循环同样从顶点0 和1 所表示的首线段开始,依次从 B 的边界中取线段,然后进行两线段相交的判定和交点的求解.

关于求两条直线段的交点,常用的方法是将两线段所在直线方程组成二元一次线性方程组,然后解这个方程组即可.但对于解得的交点仍需判断是否位于两线段的端点之间.为了减少计算量,可以先简单判断是否线段相交,然后再计算交点.具体方法可参见文献[4].

在这里采用两个动态数据结构 GFPoints 和 IndexVector. GFPoints 是一个用于存储多个顶点坐标的动态点集链表类.其意义可以是 1 条由多个顶点相连而成的弧线段.它提供了一系列成员函数,用于动态维护坐标点集.可用这个类的对象来存储线段交点.

```
class GFPoints
{.....
public:
~int32 size(); //点集中点的个数
FPOINT2D * getData(); //获取整个点集链表
//获取某个点(坐标)
FPOINT2D& pointAt(_int32 index);
//在尾部添加一个点
void addPoint(FPOINT2D& obj);
void addPoint(float x,float y);
//在链表某处插入点
void insertPointAt(FPOINT2D& obj, int index);
void removePointAt( int32 index); //删除某点
void removeAllPoints(); //删除所有点
...
};
```

typedef vector(int) IndexVector,是个存储 int 型变量的 vector 动态数据结构,这里用来存储 A 的线段和 B 的线段相交时,A 的线段的首端点序号,也即该线段的序号.

这样,每当线段相交时,就调用 GFPoints 类的成员函数 addPoint,将交点添加到交点点集中,并调用 vector 的 push_back 成员函数,添加此时 A 的边界线段的序号.

当循环进行到 A 的外环上所有线段均与 B 的所有环(包括外环和内环)上的所有线段进行完相交判别之后(如图 3 中所示),交点点集中添加的交点按顺序分别为①、②、③、④(图 3 中以小方斑表示交点),而与这 4 个交点对应的 A 的外环边界上的相交线段索引号分别为 4、6、9、12.

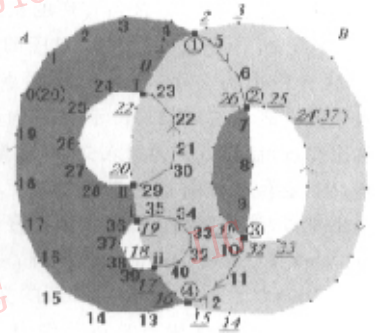

图 3　A 交 B(A 边界上的叠加结果弧线段)

(3) 成对地对交点点集和相交线段索引号进行处理. 前面已经提到引入交点和引出交点是交替成对出现的,因此可以通过判断 A 外环的首顶点(0)是否位于 B 内来确定引入、引出交点. 如果首顶点位于 B 内,那么就可以判定交点①为引出交点,从而可以判定交点②、④为引入交点,交点③为引出交点;如果首顶点不在 B 内,就可判定交点①为引入交点,从而可以判定交点②、④为引出交点,而交点③为引入交点. 这里为后一种情况.

接着,确定并生成叠加结果弧线段. 依据前面的判定方法,可以确定在 A 的边界上,交点①与交点②之间为叠加结果弧线段,包括界于这两个交点之间的 A 边界上的顶点,从点 4 到点 6,分别是点 5、点 6(不包括点 4). 那么这条叠加结果弧线段共由 4 个点组成,按点序依次为:①→5→6→②. 这一步的程序实现是:定义一个 GFPoints 类变量,并调用 GFPoints::addPoint(),依次存储这 4 个点. 同理,A 边界上交点③、④之间的弧线段也为叠加结果弧线段,包含的顶点界于点 9 至点 12 之间的 3 个点(点 10、11、12,不包括点 9),那么整个弧线段的点序依次为:③→10→11→12→④.

到此为止,已经将 A 的外环上的所有线段与 B 的所有环上的所有线段做了相交判断. 用以下的数据结构来对所有叠加结果弧线段进行存储.

```
typedef vector<GFPoints> GDotsVector;
GDotsVector NVector;
```

GDotsVector 的意义就是用于存储多个 GFPoints 变量(弧线段)的动态 vector. 调用 GDotsVector 的成员函数 push_back(),将这两条叠加结果弧线段添加进去.

(4) 依次类推,继续进行循环遍历,就可以求出 A 的第 1 个内环(21～31)与 B 所有环的交点,依次为交点Ⅰ、交点Ⅱ. 而与这 2 个交点对应的 A 的内环边界上的线段也为 2 个,它们的首端点分别为 23、28. 由于此内环的首点(21)位于 B 之内,因此交点Ⅰ为引出交点,而交点Ⅱ为引入交点,且首点(21)也即末点(31)必位于该叠加结果弧线段上. 从引入交点到引出交点,该叠加结果弧线段的点序依次为:Ⅱ→29→30→31(21)→22→23→Ⅰ,同理,可以得到 A 的第 2 个内环(32～41)与 B 所有环的交点为 i、ii,对应的 A 边界上的线段首端点分别为点 35、点 39,产生的叠加结果弧线段的点序为:ii→40→41(32)→33→34→35→i.

至此,循环结束. 在 A 的边界上共生成 4 条叠加结果弧线段.

把上面第 2 步～第 4 步的过程定义为一个函数 bool meOverlayNRegRegINTER(FPOINT2D * lpSPoints, _int32 * lpSPolyCounts, int SCount, FPOINT2D * lpTPoints, _int32 * lpTPolyCounts, int TCount, GDOTSVECTOR &NVector).

其中前 3 个参数代表叠加图元(A)的点集链表、各个环的顶点个数、环的个数,紧接着的是被叠加图元(B)的同样的 3 个参数,最后 1 个参数即为用于存储叠加结果弧线段的动态 vector. 此函数执行完后,NVector 中共存有 4 条叠加结果弧线段.

(5) 下面按照以上所述的过程,再次调用函数 meOverlayNRegRegINTER(),不同的是将 A 和 B 各自的 3 个参数互换位置,即把 B 作为叠加图元,而将 A 作为被叠加图元. 那么函数执行的结果是:在 B 的边界上共生成 4 条叠加结果弧线段. 其中,在 B 的外环边界上有 3 条,分别是(按点序):②→16→17→③,⑤→④,以及⑥→23(0)→1→①;在 B 的内环边界上仅有一条结果弧线段,点序为Ⅰ→26→27→28→29→30→31→Ⅱ,如图 4 所示.

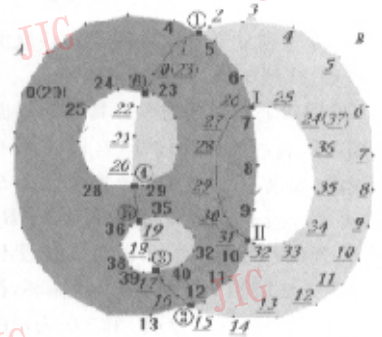

图 4 A 交 B(B 边界上的叠加结果弧线段)

需要提出的是,对于叠加结果弧线段⑤→④,因为这两个交点都是 B 外环边界上的线段(19→20)同时与 A 的 2 个边界线段(28→29)和(35→36)相交的结果,所以对应的线段索引号为 2 个 19. 如果按照上面所述的方法,叠加结果弧线段仅由这 2 个交点构成,弧线段的方向应该是④→⑤,而不是⑤→④. 这虽然并不影响结果,但事实上,这里是将交点④、⑤按照边界线段 19→20 的走向进行了排序,从而使叠加结果弧线段的方向与 B 的边界走向一致,变为⑤→④.

至此经过对函数 meOverlayNRegRegINTER 的两次调用，总共返回(4＋4)条叠加结果弧线段(如图 5 中所示).

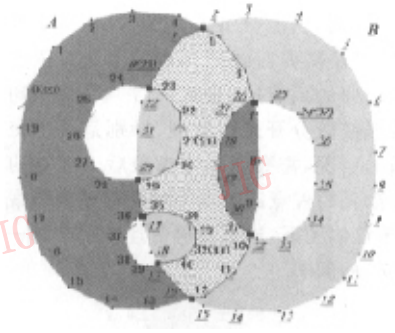

图 5　A 交 B(形成新的区图元)

(6) 执行叠加算法的最后一步，就是对这些叠加结果弧线段进行首尾拓扑连接重组，并生成新的区图元. 具体步骤是循环调用 vector::at()函数，从 NVector 中取出所有的叠加结果弧线段(4＋4)条.

```
/*循环变量 ii 的范围从 1 到 NVector.size()=8*/
GFPoints mdots=NVector.at(ii)
//往结果图层 nLayer 中添加叠加结果弧线段
  int LinNo=nLayer.AppendLine(
    mdots.getData(),mdots.size(),&iLin)
    /*其中 iLin 是 GILine 类对象，通过这个变量可以给
     该条弧段赋予新的属性标签*/
//进行拓扑连接重构，生成闭合区图元
  nLayer.Build(coTopologyPolygon);
```

如图 5 所示，网格阴影部分即为生成的区图元.

下面以区图元 S 与 T 为例，介绍叠加算法对叠加图元一条边界线段同时与被叠加图元的两条或多条边界线段相交这一特殊情况的处理. 如图 6 所示，其中 S 共 8 个顶点(0～7)，逆时针方向；T 共 12 个顶点(0～11)，顺时针方向.

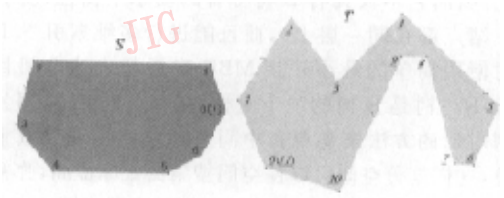

图 6　多边形图元 S 和 T

当 S 为叠加图元，而 T 为被叠加图元时，从图 6 可以看出，S 的边界线段与 T 的边界线段总共有 8 个交点(①～⑧). 其中，S 的边界线段 1→2 与 T 的所有边界线段共有 5 个交点(②、③、④、⑤、⑥). 与这 8 个交点对应的 S 边界线段的索引号，分别为 0、1、1、1、1、1、4、5. 如果不对这种情况进行特殊处理，那么按照前面所述的方法步骤，对这 8 个交点进行成对处理，将依次得到⑧→6→7(0)→①、②→③、④→⑤、⑥→2→3→4→⑦ 4 条叠加结果弧线段，而其中②→③和⑥→2→3→4→⑦并不是正确的叠加结果弧线段. 这说明前面描述的算法需要进行补充，就是针对叠加图元的一条边界线段同时与被叠加图元的多条边界线段相交的情况，也即在与交点序列对应的叠加图元边界线段索引号序列中出现重复序号的情况时，必须在确定叠加结果弧线段之前，对这些同位于一条边界线段上的交点，按照该条边界线段的走向进行排序，以保证得到的叠加结果弧线段的方向与叠加图元边界的走向一致. 一个有效的排序方法就是，如果判定该条边界直线段从首点到尾点，X(或 Y)坐标值是递增(减)的，那么也必须使这些交点按照 X(或 Y)坐标值递增(减)的方式排序，如果对②、③、④、⑤、⑥按照线段 1→2 的走向重新排序后，得到的这 5 个交点的新点序为④、⑤、⑥、③、②，那么整个交点的点序为①、④、⑤、⑥、③、②、⑦，而与之对应的 S 边界线段的索引号序列仍为 0、1、1、1、1、1、4、5，最后再按照前面的方法对它们进行成对处理，得到的 4 条正确结果叠加弧线段依次为⑧→6→7(0)→①、④→⑤、⑥→③、②→2→3→4→⑦(如图 7 所示).

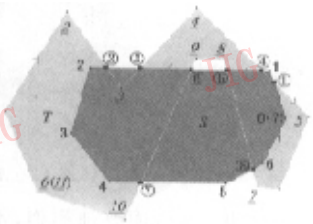

图 7　S 交 T(S 边界上的叠加结果弧线段)

同理将 T 作为叠加图元，S 作为被叠加图元，可以得到 T 边界线段上共 4 条叠加结果弧线段，分别是：①→3→②、④→③、⑥→⑤、⑦→⑧(如图 8 所示).

最后对这(4＋4)条叠加结果弧线段进行首尾拓扑连接重组后，就得到如图 9 中所示的 2 个新多边形(斜纹阴影部分).

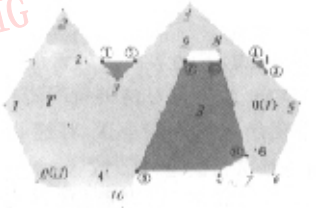

图 8 S 交 T(S 边界上的叠加结果弧线段)

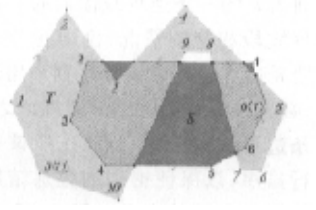

图 9 S 交 T(形成新的区图元)

以上介绍的是多边形叠加的交操作,那么对于多边形并和多边形差操作,同样可以用以上算法实现.但需要注意的是,对于叠加并或差操作,叠加结果图元边界中属于叠加图元边界部分的特点是:始于"引出交点",而止于"引入交点".而属于被叠加图元边界部分的特点是:叠加并操作中,是始于"引入交点",而止于"引出交点";在叠加差操作中仍是始于"引出交点",而止于"引入交点"(如图 10 和图 11 所示).

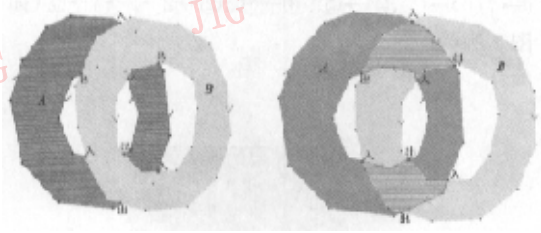

图 10 A 并 B

图 11 $A-B$

对于线面叠加,与面面叠加类似,不同之处在于:线可以是不闭合的,首点和末点不重合,且没有内环,而面可以看成是闭合的线,首点和末点重合,可以有内环;线与面相交,交点可以为奇数,而面面相交,交点必为偶数.注意,本算法把一条直线段的一个端点(首或末)位于另一条直线段上的相交情况,处理为得到两个相同的交点(即为该端点);把一条线段与另一条线段重合或部分重合的相交情况,处理为得到中间的两个端点.那么对于线面叠加问题,完全可以采用面面叠加的算法来实现.这里需要明确的是,当叠加线图元的首点和末点同位于被叠加区图元的内部或外部时,线面相交的交点个数仍为偶数,而当首、末点分处在区图元内外两侧时,交点个数才为奇数.对于这两种情况,仍可以通过"引入→引出"或"引出→引入"原则对这些交点进行配对处理.只是需要对线图元边界上从首顶点到第一个交点之间的弧线段,和从最后一个交点到末顶点之间的弧线段分开进行处理.特别是对于交点个数为奇数的情况,需要额外考虑最后未配对的一个交点与线图元的首点或末点间的叠加结果弧线段的生成.由于篇幅关系,这里不再赘述.图 12 为一个线面叠加(交)的例子.

图 12 线面叠加(交)

3 R+树空间索引结构

所谓空间索引就是指依据空间对象的位置和形状或空间对象之间的某种空间关系,按一定顺序排列的一种数据结构,其中包含空间对象的概要信息,如对象的标识、外接矩形及指向空间对象实体的指针.作为一种辅助性的多维数据结构,空间索引介于空间操作算法和空间对象之间,通过它的筛选,大量与特定空间操作无关的空间对象被排除,使空间操作能够快速访问操作对象,从而提高空间操作的效率[1,8].

目前空间索引应用最广泛的是以 R 树及其变种为代表的动态索引技术.R 树实际是对 B 树的扩充,因而它不仅具有 B 树特有的动态平衡性(所有叶结点都在同一层上),而且能进行多维索引[9].R 树使用简单的最小外接 MBR 来近似表达空间目标.R+树是 R 树的一个变种,在 R+树中,采用分割矩形的方法来避免在中间节点之间出现区域重叠,这样划分空间可以使空间搜索的效率提高,弥补了 R 树的不足.如图 13 为 R+树结点层次图.

R+树索引的算法包括 R+树的建立、节点的插入、删除和对 R+树的查询.具体算法参见文献[10].下面是用 C++编程实现的 R+树类.

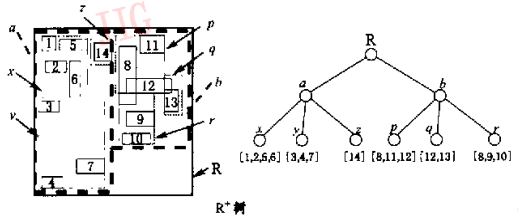

图 13　R＋树索引示意图

```
class CRTree
{
private:
bool root is data;
int num of data;
int num_of_dnodes;
int num_of_inodes;
……
public:
//创建索引
void Create(const char * distname,int pagelen);
void Open(const char *name); //打开索引
void Close(void); //关闭索引
void Insert (int id, float xmin, float xmax, float ymin,
float ymax);/*插入图元,id 是图元的标识符,后面是图元
的外接矩形 MBR 参数*/
void Delete(typrect rect);//删除矩形
void ExistsRegion(typrect rect); //查找指定矩形
void RegionCount(typrect rect); //返回矩形数
void point_query(FPOINT2D&. pnt); //点查询
void RegionQuery (typrect rect, int * result,
int querytype);//矩形区域或窗口查询(相交或包含)
……
}
```

其中矩形区域(或窗口)查询可以是通过在屏幕上设定一个矩形框,然后按照矩形框在 R＋树中检索与矩形框相交的元素,生成新的图层作为查询结果.

4　R＋树对叠加分析算法的改进

由于本算法采用的是循环遍历叠加图层中的每个图元,并分别与被叠加图层中的每个图元进行叠加处理,所以当图层中图元数目很多,数据量很大时,算法非常耗时. 如图 14 为中国行政区图与一地形图的叠加结果图. 但如果考虑利用 R＋树对整个图层的图元创建索引,也即将图层中所有图元的 MBR 按 R＋树的层次形式来组织,那么,就可以通过 R＋树本身的高效访问机制,在叠加操作之前,对空间数据进行快速查询和筛选,减少参与叠加处理的图元数目,提高空间操作的效率.

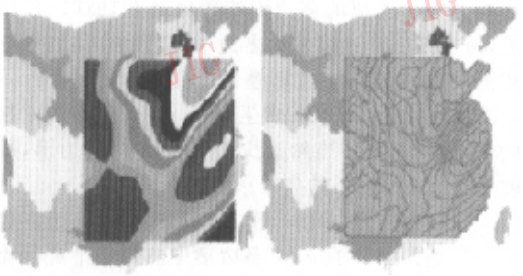

图 14　中国行政区图与一地形图的叠加

具体方法如下:

(1) 面图层与面图层叠加,以及线图层与面图层叠加

① 比较进行叠加分析操作的两个图层的图元数目,对图元较多的那个图层创建 R＋树索引(调用 CRTree::Create()和 CRTree::Insert());

② 遍历图元较少的图层中的每个图元,并用该图元的 MBR 作为查询窗口矩形,对索引图层进行R＋树窗口区域查询(调用 CRTree::RegionQueries()),这样,利用 R＋树索引快速地查找出仅与查询窗口相关(相交或包含)的索引图层中的图元;

③ 将作为查询窗口的图元(叠加图元)与查找出来的这些相对较少的图元(被叠加图元)进行遍历叠加操作.

(2) 点图层与面图层叠加

① 对面图层创建 R＋树索引;

② 遍历点图层中的每个点图元,并用该点作为查询点,对索引图层进行 R＋点查询(调用 CRTree::point_query()),这样就可以利用 R＋树索引快速地查找出查询点有可能落入的索引图层中的几个面图元;

③ 将作为查询点的点图元(叠加图元)与查找出来的这些相对较少的面图元(被叠加图元)进行循环遍历叠加操作.

5　计算复杂度与实验结果

叠加分析操作尤其是线面叠加和面面叠加操作,由于需要解决直线段对之间的交点判断问题,因此其计算量将非常大,而且耗时. 它是矢量 GIS 程

序中最复杂的操作之一.

很明显,弧段和多边形的数目以及所含坐标点对的数目直接影响所需的计算量.在一个多边形叠加操作中所生成的多边形的数目通常是难以预测的,特别是会产生很多碎多边形.实践表明,如果叠加图层中的 n_1 个多边形叠置于被叠加图层中的 n_2 个多边形之上,一般会产生(n_1+n_2)的 3 倍或 4 倍的结果多边形[2].当然最少是 n_1+n_2 个多边形,那就是在 2 个地图上的多边形根本不相交.

如果叠加图层的 n_1 个多边形中的每一个多边形与被叠加图层的 n_2 个多边形中的每一个进行比较,算法将具有 $O(n_1n_2)$的计算复杂度.

由于本算法利用了 R+树索引的快速访问机制,因此能够快速地筛选出索引图层(被叠加图层)所有图元中所有可能与未索引图层(叠加图层)的每个图元相关的图元,如果不考虑创建 R+树索引以及基于 R+树的窗口查询或点查询所耗的时间(与图元叠加的开销相比,可忽略),那么就构造了一个复杂度近似为 $O(n_1)$的算法.这意味着本算法是更高效的.表 1 所示是对本算法运用 R+树索引前后的测试比较.

表 1 使用 R+树前后的算法性能测试比较

	叠加图元个数	被叠加图元个数	执行时间(s)
未使用 R+树	24	31	5
	2 955	5 741	163
使用 R+树	24	31	2
	2 955	5 741	52

6 结 论

地图叠加分析是 GIS 空间分析中非常重要的功能,尤其是矢量叠加分析,在 GIS 应用的其他许多方面都得到了利用,比如开窗查询、缓冲区合并操作、多边形裁剪、空间连接查询等等.叠加分析算法的好坏,性能的优越与否直接影响到 GIS 整体功能的发挥.本算法,尤其是针对面面叠加和线面叠加问题,将多边形穷举求交与"引入、引出交点交替配对"的叠加结果图元边界生成原则有机结合在一起,并引入 R+树空间索引对空间数据的高效存取机制,实现了一种新的基于 R+树的双重循环叠加算法,很好地解决了叠加操作本身计算量大、复杂耗时的问题,为实现大数据量的地图叠加分析提供了一个很好的思路.

参 考 文 献

1 陈述彭,鲁学军,周成虎. 地理信息系统导论. 北京:科学出版社,1999.

2 Joseph MPlwowar. Introduction to Geographic Information System [EB/OL]. http://www.watleo.uwaterloo.ca/~piwowar/geog255/Overlay/Overlay.html,1999-10-08

3 Frank A U. Overlay processing in spatial information systems [A]. In:Eighth International Symposium on Computer Assisted Cartography[C],Baltimore,Maryland,USA,1982:12~31.

4 郭仁忠. 空间分析[M]. 武汉:武汉测绘科技大学出版社,2000.

5 周培德. 计算几何[M]. 北京:清华大学出版社,2000.

6 Nievergelt J, Preparata F P. Plane-sweep algorithms for intersecting geometric figures[J]. Communications of the ACM, 1982,25(10):739~747.

7 Frankin W R, Narayanaswami C, Kankanhalli M *et al*. Uniform Grids: A Technique for Intersection Detection on Serial and Parallel Machines[A]. In: Ninth International Symposium on Computer Assisted Cartography [C]. Baltimore, Maryland, 1989:100~109

8 刘东,李琦,承继成. 主存空间对象的索引方法. 环境遥感,1996,11(4):302~308.

9 李立,石树刚,郑振楣. 空间索引技术及其在 SamBase 中的实现. 交通与计算机,1995,13(3):42~47.

10 Sellis T, Roussopoulos N, Faloutsos C. The R +-tree: A dynamic index for multidimensional objects[A]. In:Proceedings of 13th International Conference on Very Large Data Bases[C], Brighton,1987:507~518.

董 鹏 1976 年生,2000 年获太原理工大学硕士学位,现为中科院遥感应用研究所博士研究生.主要从事网络地理信息系统的研究与开发.

杨崇俊 1954 年生,1990 年获法国遥感博士学位,现为中科院遥感所研究员,博上生导师.目前在网络地理信息系统方面开展研究和应用工作.

刘冬林 1971 年生,1998 年获中国矿业大学硕士学位,现在中科院遥感应用研究所工作.主要从事网络地理信息系统的研究与开发.

芮小平 1975 年生,2001 年获中国矿业大学硕士学位,现为中科院遥感所博士研究生.研究方向为网络三维地理信息系统.