图层级矢量地图裁剪计算模式与算法策略

王庆刚[1],田生军[2],范协裕[1],杨延青[1]

(1. 中国科学院遥感与数字地球研究所遥感科学国家重点实验室,北京 100094;
2. 北京中遥地网信息技术有限公司,北京 100101)

摘要:矢量地图裁剪是商业GIS软件平台重要的基础功能之一。而各种商业GIS平台的矢量地图裁剪效率存在较大差异,其中,ArcGIS效率较高。本文提出了一种矢量地图裁剪计算模式:首先,筛选与裁剪要素外接矩形框(MBR)相交或者包含于该矩形框内的被裁剪要素;然后,对筛选出的被裁剪要素构造四叉树索引,根据被裁剪要素的类型采用不同的计算模式完成裁剪;最后,采用线程池技术实现并行高效的裁剪计算过程。实验结果表明,本文提出的方法在矢量地图裁剪方面与ArcGIS 10平台的效率相当。

关键词:矢量地图;四叉树索引;图层裁剪;计算模式;线程池

DOI:10.3724/SP.J.1047.2013.00532

1 引言

地理信息系统矢量数据空间分析处理基本功能包括:缓冲、裁剪、相交、联合、合并及融合等。它们的执行效率对空间分析至关重要。目前,主流的GIS基础软件,如ArcGIS、MapGIS、SuperMap等均未完全公布各自所采用的矢量地图裁剪算法。当数据量较小时,不同GIS平台软件空间分析效率差距不大;但随着数据量的增加,它们之间的效率差距可能呈几何级数拉开。其中,ArcGIS的图层级矢量数据的裁剪效率较高。部分开源软件(如:Quantum GIS等)也实现了图层级矢量数据的裁剪,但与ArcGIS的效率差距也较大。因此,在方法上研究图层级裁剪的高效实现非常有必要。

图层裁剪的实现,实际上涉及到两个层面的研究工作。一是基础裁剪算法,二是裁剪计算模式。目前地理信息系统中最小颗粒度核心算法已经相当稳定和成熟,如:Weiler算法[1]、Vatti算法[2]、Greiner-Hormann[3]等算法。裁剪计算模式则主要研究技术集成和流程设计,最终实现高效率的图层裁剪功能。本文旨在推介一个新的裁剪计算模式,提供一种高效率的图层裁剪实现策略。

裁剪运算涉及两个图层:一个为裁剪图层(clip layer);另一个为被裁剪图层(clipped layer)。其中,裁剪图层必须为面图层或者多面图层,而被裁剪图层可以为点(point)、线(line)、面(polygon)、多点(multi-point)、多线(multi-line)和多面(multi-polygon)等任意图层。针对不同矢量数据类型,有表1所示的6种裁剪方式。

表1 图层关系

Tab.1 Relationship of layers

裁剪图层	被裁剪图层	结果图层
面	点	点
	线	线
	面	面
	多点	多点
	多线	多线
	多面	多面

本文针对不同的被裁剪数据类型,以及它们各自不同的图形和属性特点,分别采用不同的裁剪策略,实现了高效的图层级别矢量数据裁剪。

2 图层级矢量地图裁剪计算模式

图层级矢量地图裁剪计算模式实现顺序步骤如图1所示。

2.1 矢量地图裁剪计算模式

在进行矢量地图裁剪计算时,如果裁剪之后的

收稿日期:2013-01-14; **修回日期**:2013-04-24.

基金项目:国家"863"计划资助项目(2012AA12A401)。

作者简介:王庆刚(1984-),男,硕士生,研究方向为网络空间信息系统。E-mail:camelWQG@hotmail.com

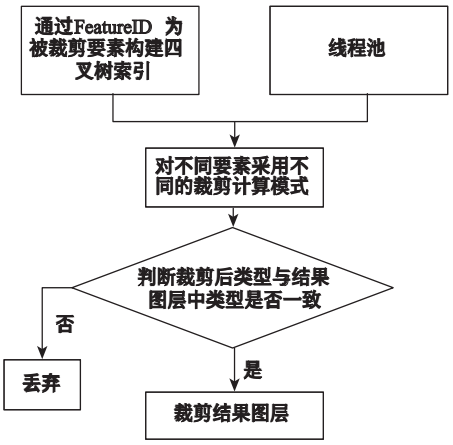

图1　图层级裁剪功能基本框架图

Fig.1　Framework of the clipping function at the layers' level

要素类型和被裁剪数据类型不一致,就不能将其放入结果图层中,故对裁剪后的要素需进行数据类型判断。表2详细列出了在裁剪过程中可能出现的特殊情况。

表2 处理特殊裁剪结果

Tab.2 Handling the exceptional cases

裁剪图层	被裁剪图层	裁剪结果	需要丢弃要素
	点	点	无
	线	点、线	点
	面	点、线、面	点、线
面	多点	多点	无
	多线	多线、多点、点	多点、点
	多面	点、多点、多面、线、多线	点、多点、多线、线

本论文针对点、线[4]、面,以及多点、多线、多面,采用不同的裁剪计算模式[5]进行裁剪,下面分别就这6种情况作详细的介绍,图2给出了图层级裁剪计算模式流程图。

2.1.1　点裁剪计算模式

点裁剪计算模式只需进行空间关系判断,不需进行裁剪操作,具体步骤如下:

(1) 查询并获取在裁剪范围内部的点要素;

(2) 将查询结果的图形信息和属性信息全部添加进Result-Layer。

2.1.2　线、面裁剪计算模式

线(或面)裁剪计算模式需根据裁剪要素与被裁剪要素之间空间关系,再进行相应裁剪操作,具体步骤如下:

(1) 查询与裁剪范围相交或者在裁剪范围内部的线(或面)要素;

(2) 完全在裁剪范围内部的线(或面)要素,直接放入Result-Layer;

(3) 裁剪与裁剪要素相交的线(或面)要素,将裁剪结果放入Result-Layer(需判断裁剪结果数据类型与结果图层是否一致,不一致的数据将丢弃)。

2.1.3　多点裁剪计算模式

多点裁剪计算模式,与点裁剪计算模式不同,需根据裁剪要素与被裁剪要素之间空间关系,再进行相应裁剪操作。具体步骤如下:

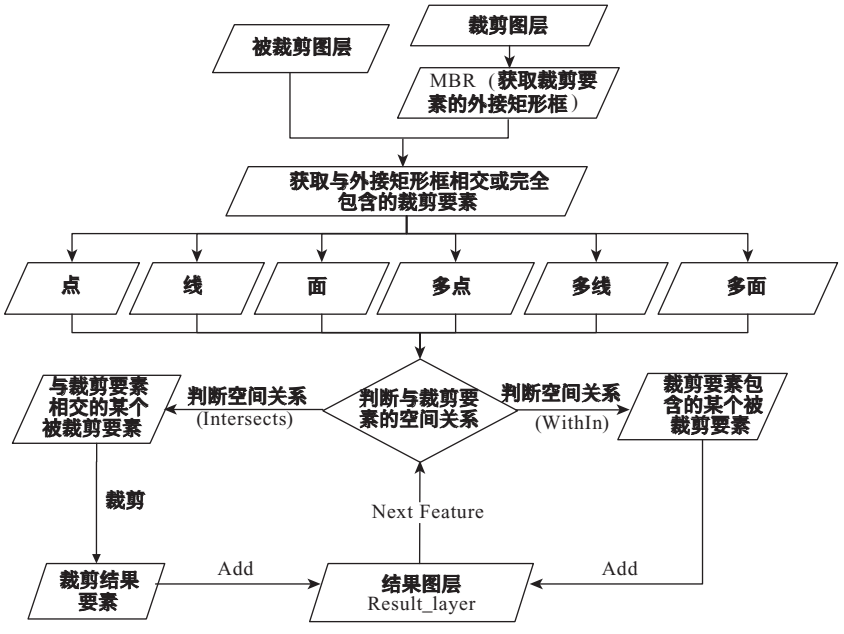

图2　图层级裁剪计算模式流程图

Fig.2　Flow chart of computing model at the layers' level

（1）查询与裁剪范围相交或者在裁剪范围内部的多点要素；

（2）将在裁剪要素范围内的多点要素直接放入Result-Layer；

（3）处理与裁剪要素相交的多点要素。分析多点要素中每个点要素与裁剪要素的空间关系，将包含于裁剪要素点要素逐个放入新创建的Multi-point要素中，最后再将Multi-point放入Result-Layer；

2.1.4 多线、多面裁剪计算模式

多线（或者多面）裁剪计算模式需根据裁剪要素与被裁剪要素之间空间关系，再进行相应裁剪操作。具体步骤如下：

（1）查询并获取与裁剪范围相交或者在裁剪范围内部的多线（或多面）要素；

（2）完全在裁剪范围内部的多线（或多面）要素，直接放入Result-Layer；

（3）裁剪与裁剪要素相交的多线（或多面）要素，将裁剪结果放入Result-Layer（需要判断裁剪结果数据类型与结果图层是否一致,不一致的数据将丢弃）。

Multi-Line（或者Multi-Polygon）要素中可能有一个几何图形（Geometry）或者多个几何图形Multi Geometry，如果只有一个Geometry，则直接进行裁剪并将裁剪结果放入Result-Layer中即可；如果有多个Geometry，则需要对每个Geometry单独进行裁剪，并将裁剪结果放入一个新创建的Multi-Line（或者Multi-Polygon）要素中，最终将该Multi-Line（或者Multi-Polygon）要素放入Result-Layer中。

2.2 改进后四叉树索引

空间索引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，按照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。一般而言，空间索引的遍历效率，即它的存取时间与数据集合大小成对数关系[6-7]。

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，通过筛选，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[8]。

当对数据进行处理时，构建合适的索引能够改善数据的处理效率。目前，常采用的索引主要有KDB树、G树、Cell树和R树等[9-10]，还有一些学者提出QR树[11]，DBP+树[12]等索引。

四叉树索引（如图3）是一种面向主存的空间索引技术[13]。由于其简单，四叉树通常用来加速对二维平面中的空间对象存取访问，尤其是在优化点查询和开窗查询等空间操作方面，四叉树具有独特的优势[6]。四叉树索引方法对一张地图逐步4等分，细分层次由用户视情况而定，细分的结果可再分成四叉树。

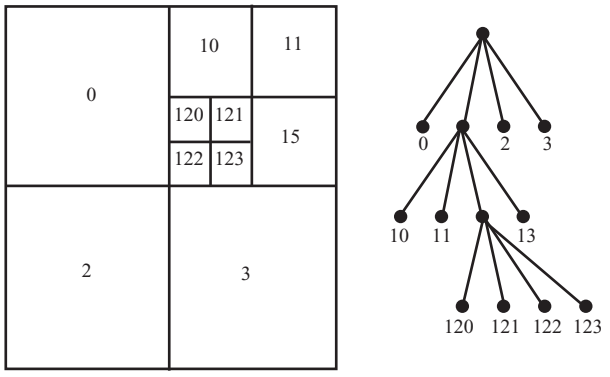

图3 四叉树

Fig.3 The quad-tree

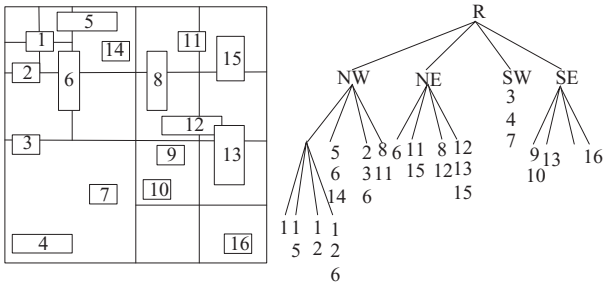

图4 传统四叉树

Fig.4 Traditional quad-tree

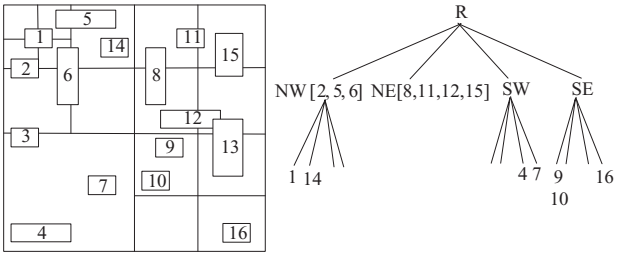

图5 改进型四叉树

Fig.5 Improved quad-tree

传统四叉树各节点有可能包含同一元素，本文所采用四叉树索引[6]是一种改进后的四叉树。其实现步骤如下：首先，根节点包含全部要素；其次，将

包含全部要素的外接矩形框进行四等分,也就是根节点下4个新叶子节点,每个节点只包含完全在该节点外接矩形框内部的要素,并将它们从父节点中去掉,如此就避免了不同节点包含相同要素的情况;接着,按照步骤2不断进行四等分,直到叶子节点为空才停止。

图4和图5中的阿拉伯数字是要素FeatureID,该值具有唯一性。通过观察图4和图5差异发现:在传统四叉树中要素分布偏向于叶子结点,改进后的四叉树很好地实现了各节点的负载均衡,这样减少了信息冗余,提高了索引树的搜索效率。

2.3 线程池

本文将被裁剪图层中要素逐个进行裁剪处理,并将裁剪结果放入结果图层中,由于被裁剪的各要素之间能够相互独立进行,不存在相互影响和制约的情况,故可以进行多线程并行运算。

线程池技术为线程创建、销毁开销和系统资源不足等问题提供了良好的解决方案,有效地提高系统响应速度和整体性能,具体有如下4个方面的优点:

(1)根据实验环境的需要,创建最适宜的线程数量。

(2)降低系统开销和资源消耗。通过将多个请求重用线程,线程创建、销毁的开销分摊到多个请求上。另外,通过限制线程数量,降低虚拟机在垃圾回收方面的开销[14]。

(3)降低由于不断内存申请和释放所带来内存碎片问题[15]。

(4)提高系统响应速度。

本文所使用线程池主要接口如下:

①创建线程池并指定线程池中线程个数为num_threads_in_pool

threadpoolcreate_threadpool(int num_threads_in_pool)

②给线程池中线程分配任务,通过from_me指定线程池,dispatch_to_here指定任务函数,arg指定任务函数参数。

int dispatch_threadpool(threadpoolfrom_me,dispatch_fndispatch_to_here,void *arg);

③销毁线程池对象,void destroy_threadpool(threadpooldestroyme);

表3中体现了是否使用线程池实现裁剪功能的时间比,t1,t2,t3分别为3次测试所消耗的时间。表3中"普通"指采用裁剪算法,但没使用多线程并行;"线程池"指采用本文裁剪算法的同时使用了多线程并行。

表3　线程池和非线程池裁剪性能分析(s)

Tab.3　Analysis of the performance of clipping between thread-pool and none thread-pool (s)

		裁剪要素	被裁剪要素	t1	t2	t3
点	普通	266 514	34	3.11	3.14	3.01
	线程池			2.11	2.22	2.09
线	普通	2 028 804	34	66.43	68.15	63.94
	线程池			57.05	57.30	58.50
面	普通	38 904	1188	546.74	556.67	557.56
	线程池			271.52	273.46	278.98

3　算法验证

本文图层级矢量地图裁剪计算模式所使用的实验环境:操作系统为Window7,2G内存,2.40 GHz主频;所用的软件开发集成环境为Microsoft Visual studio 10,开发语言为C++,最小颗粒度算法使用开源库GDAL[16]和GEOS[17];线程池中线程数量为15个;ESRI公司的ArcGIS 10.0版本;测试均在相同硬件环境下进行;测试表4列出了本次实验所用到的裁剪数据和被裁剪数据。

表4　裁剪数据和被裁剪数据说明

Tab.4　Illustration of the clip data and clipped data

	被裁剪数据	裁剪数据(面)
点	中国全图1:25万比例尺高程点数据	北京市行政区划图数据(图6)
线	中国全图1:25万比例尺高程线数据	北京市行政区划图数据(图6)
面	全球省级行政区划图	中国行政区划图数据(图7)

图6,图7是对测试所用到的部分数据进行的截图,表5列出了裁剪要素和被裁剪要素个数,并对本次裁剪计算模式的性能进行了详细对比。

如表5所示,本文分普通和多线程并行2种情况,与ESRI公司GIS平台软件相比,其中平均耗时是根据表3中的3次测试值计算得到,普通和多线程并行情况下均采用了本文算法,但普通没有采用多线程并行,具体性能如下:

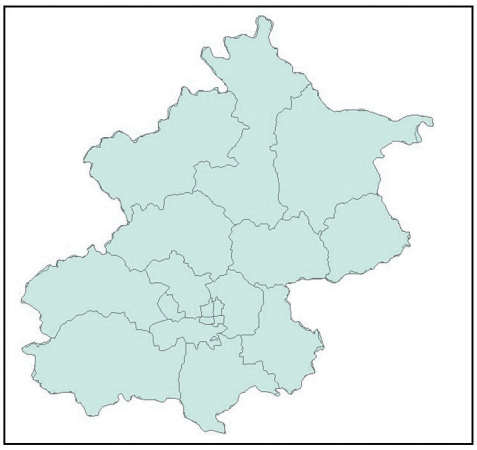

图6 北京市行政区划图

Fig.6 The administrative map of Beijing

图7 中国行政区划图

Fig.7 The administrative map of China

表5 裁剪时间分析(s)

Tab.5 Analysis of the clipping time (second)

	被裁剪要素个数	裁剪要素个数	普通平均耗时	多线程并行平均耗时	ArcGIS耗时
点	266 514	34	3.1	2.2	3.8
线	2028 804	34	66.2	57.6	90.3
面	38 904	1188	553.7	234.7	170.8

普通平均耗时情况下:点图层裁剪平均耗时减少0.7s;线图层平均耗时减少24.1s;面图层平均耗时多372.9s。多线程并行情况下:点图层裁剪平均耗时减少1.6s;线图层平均耗时减少32.7s;面图层平均耗时多53.9s。本文图层级裁剪计算模式优点是与ArcGIS 10.0裁剪效率相比本文点、线裁剪算法略高;缺点是采用多线程时面裁剪略微偏低,不采用多线程并行时面裁剪效率较低。这个计算模式的一个特点是CPU占用率较高。

4 结论

本文提出了一种新的图层级矢量地图裁剪计算模式。其特点是:针对点、线、面、多点、多线、多面,采用了不同计算策略;采用了改进后的四叉树;采用了线程池并行技术。采用北京市行政区划图数据、中国行政区划图数据、中国全图1∶25万比例尺高程点数据和全球省级行政区划图进行测试。最终测试结果表明,本文图层级矢量数据裁剪计算模式与ESRI公司的ArcGIS 10.0具有基本持平的裁剪效率,其中,点图层、线图层的裁剪效率略高,面图层略低。如何进一步降低CPU占用率,是后续的工作内容。在本文基础上,如果能够在云平台[18]或者分布式[19]系统上面实现海量数据[20]的图层级裁剪功能,将会为广大地学工作者带来较高的应用效率。

参考文献:

[1] Weiler K, Atherton P. Hidden surface removal using polygon area sorting[C]. The SIG GRAPH'77,New York,1977.

[2] Vatti B R.A generic solution to polygon clipping[J].Communications of the ACM,1992,35(1):56-63.

[3] Greiner G, Hormann K. Efficient clipping of arbitrary polygons[J]. ACM Transactions on Graphic,1998,17(2):71-83.

[4] Xie Z,Wei D Q,Wu L. Graph model of polygon clipping using simple vector data[J].ActaGeodaetica et Cartographica-Sinica,2009,38(4):369-374.

[5] 马丽娜.面向大规模空间数据的空间计算模式研究与实现[D].北京:中国地质大学,2011.

[6] 董鹏,杨崇俊,芮小平,等.一种基于改进四叉树的GIS空间选择查询算法[J].计算机工程与应用,2003,39(13):58-61.

[7] 董鹏,李津平,白予琦,等.基于改进四叉树索引的矢量地图叠加分析算法[J].计算机辅助设计与图形学,2004,16(4):530-534.

[8] 陈述彭,鲁学军,周成虎,等.地理信息系统导论[M].北京:科学出版社,1999.

[9] Theodoridis Y, Sellis T, Papadopolos A N, *et al.* Specifications for efficient indexing in spatiotemporal databases[C]. SSDBM'98, Capri, 1998.

[10] 余登峰.基于R树的空间数据索引技术研究与实现[D].北京:中国地质大学,2006.

[11] 郭菁,郭薇,胡志勇,等.大型GIS空间数据库的有效索引结构QR-树[J].武汉大学学报(信息科学版),2003,28(3):306-310.

[12] 唐继勇,白新跃,杨峰,等.基于DPS+-Tree的索引复制策略研究[J].计算机科学,2005,32(11):112-114.
[13] 朱庆,林珲.数码城市地理信息系统[M].武汉:武汉大学出版社,2004,38-44.
[14] 王华,马亮,顾明,等.线程池技术研究与应用[J].计算机应用研究,2005(11):141-142.
[15] 庞丽萍.操作系统原理(第三版)[M].武汉:华中科技大学出版社,2000.
[16] http://www.gdal.org/
[17] http://trac.osgeo.org/geos/
[18] 李玮顾,郝香山,程俊,等.GIS云服务平台体系架构与技术实现[C].2011年SuperMap GIS技术大会论文,2011.
[19] 杜书杰,林晓彤.分布式计算模式中影响算法性能的主要因素[J].计算机工程,29(16):162-164.
[20] 刘丽艳.基于数据网格的海量数据管理若干关键技术研究[D].北京:中国科学院研究生院,2005.

Clipping Computing Model for Vector Map at the Layers' Level

WANG Qinggang[1*], TIAN Shengjun[2], FAN Xieyu[1] and YANG Yanqing[1]
(1. *State Key Laboratory of Remote Sensing Science, Institute of Remote Sensing and Digital Earth, CAS, Beijing* 100094, *China;*
2. *Beijing Geobeans Information Technology Co. Ltd, Beijing* 100101, *China*)

Abstract: Clipping function is one of the fundamental functions in Geographic Information System. The efficiency of clipping function can greatly affect the overall performance of data processing in GIS application, especially when clipping large vector data. This paper proposes a new clipping computing model to effectively fulfill the clipping function for different types of clipped layers according to their different attribute and geometry characteristic, such as point, line, polygon, multi-point, multi-line, multi-polygon and so on. This clipping computing model consists of the following three steps: first of all, using clip-layer's Minimum Bounding Rectangle to select features from clipped-layer, where the features must intersect with or within the clip-layer's Minimum Bounding Rectangle; secondly, using Feature ID to build revised quad-tree index for the clip-layer features; lastly, using thread-pool to fulfill clip computing in parallel. A performance test was carried out using four different vector data layers. The result shows that the performance of clipping function based on this clipping computing model is as efficient as the one fulfilled in ESRI's ArcGIS 10.0. CPU-intensive is one drawback of the implementation of this model at its current form.

Key words: vector map; quad-tree index; layer clipping; computing model; thread-pool

***Corresponding author:** WANG Qinggang, E-mail: camelWQG@hotmail.com