球面三角区域四叉树*L*空间填充曲线*

袁 文 程承旗 马蔼乃

(北京大学遥感研究所, 北京 100871)

管晓静

(中国国家信息中心, 北京 100045)

摘要 球面三角四叉树中面片和结点的排列顺序直接关系到球面三角四分剖分模型组织和管理数据的效率. 在 Lee 编码模型基础上设计了 *L* 面片和结点空间填充曲线, 给出了面片寻址、结点 *L* 曲线生成、以及面片结点访问等主要算法. 同时, 基于位码运算提出了面片类型判别恒定算法时间优化算子, 可利用硬件来实现. 结点 *L* 曲线中大多数面片结点间距离分布在较低值范围内, 为数据高效存取提供了保证. 但是堂兄弟面片位置相邻, 结点地址却不连续, 少数面片结点间距离异常大, 导致平均结点间距离和遍历总距离的增大. 为解决该问题, 采用了 *m* 簇完备结点集作为 *n* 剖分簇结点存储基本单元, 每个 *m* 簇完备结点集重复存储公共结点, 从而避免了面片结点距离过大, 提高了节点访问效率.

关键词 全球格网 球面三角区域四叉树 **SQT** **QTM** 空间填充曲线

全球格网(global grid)主要研究如何将地球表面(或球面)剖分成一个等面积、等形状[1)]、具有多分辨率的层次结构[1~5], 它在全球性气候研究、资源管理与环境保护、动植物生态监测[1]、地球物理、分子物理化学[6]等领域中有着广泛的应用. 在众多全球格网剖分方案中, 只有基于内切正八面体(octahedron)和正二十面体(icosahedron)的球面三角剖分能近似满足等面积、等形状等的要求[1~4,7]. 球面三角剖分主要有两种方法, 即四分法和九分法, 其中四分法产生球面三角区域四叉树, 方法简单[1~3], 目前大多数研究采用了该法. 比较著名的球面三角区域四叉树剖

2003-04-19 收稿, 2004-03-24 收修改稿

* 国家 863 高技术计划重大课题资助项目(批准号: 2002AA783060)

1) 指每次剖分后每个单元的形状保持不变(比如三角形依然是三角形)以及每层中各个单元全等(边长相等、夹角相等)

分模型有Dutton基于正八面体构建的QTM模型[7,8]和Fekete基于正二十面体构建的SQT模型[9,10](如图 1 示). 现有大多数工作围绕球面三角区域四叉树的剖分方法[1~3,6,7,9~11]、面片(triangle facet)编码模型[7~10,12~15]、邻近面片搜索[9,10,12,15]以及不同剖分方法比较[1~4,11]等方面展开. 在面片排列顺序方面的研究中, 目前只有Lugo[14]做了工作, 提出了QTM面片排列顺序TQS并探讨了QTM与Morton码间的转换. 另外, 在结点性质的研究方面, 除Dutton[7,8]利用结点命名来计算面片编码(不涉及排列顺序)外, 尚无其他相关研究的介绍.

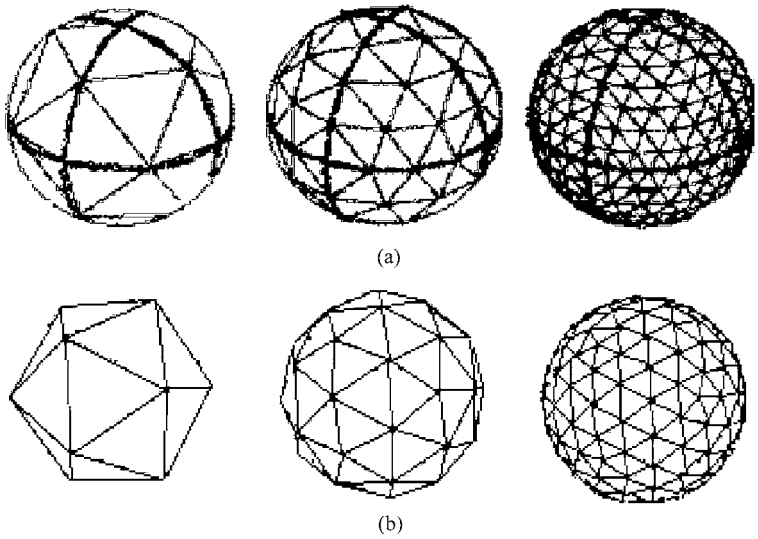

图 1　QTM(a)与 SQT(b)

由于QTM和SQT剖分三角几何形状相对规则, 近似等形状和等面积, 具有层次性、多分辨率、面片编码长度与数据精度相关联等特性, 可作为空间索引基础. Goodchild[5]和Dutton[8]认为在数字地球中可用之来取代经纬度坐标, 作为空间数据的表示、组织与空间索引的基础. 数字地球中的数据是海量的, 多分辨率的, 要求快速存取、处理和显示. 因此, 很有必要深入研究结点和面片的排列顺序, 充分考虑剖分的层次性特征, 最大程度地保证球面剖分面片和结点的空间几何分布和聚集性, 以有利于数据快速存取, 提高效率.

研究面片和结点的排列顺序也是面片和结点快速计算的要求. 在球面三角区域四叉树中, 每个三角面片包括 3 个结点, 每 6 个相邻面片共享一个结点, 内部每个结点与 6 个结点或 6 个球面三角相邻, 边界上的结点周围有 2 个或者 4 个结点以及 1 个或 3 个球面三角. 结点组成了三角面片的基本骨架. 球面三角区域四叉树的许多计算直接与结点有关, 如剖分网的形成等. 结点计算包括 2 个基本操作: (ⅰ) 访问面片的 3 个结点; (ⅱ) 遍历父面片的所有面片的结点集. 经过n深度剖分, 一个球面三角所产生的结点数为 $2^{2n-1}+3\times2^{n-1}+1$. 表 1 为一个球面剖分结

点和面片数目的统计情况. 可以看出, 随着剖分深度加大, 球面剖分结点数目急剧增长. 如此海量的结点数目必然对结点计算带来不可忽略的影响. 面片的数目也同样巨大(2^{2n}). 鉴于面片与结点排列顺序在实际应用中非常重要，本文将围绕该主题展开.

表 1 球面三角剖分结点数目统计

剖分深度 (n)	单独球面三角结点数	球面结点总数(QTM) (单独球面三角结点数×8)	对应地表分辨率 (QTM)
11	2,100,224	16,801,792	5 km
13	33,566,721	268,533,768	1 km
20	549,757,386,753	4,398,059,094,024	10 m
23	35,184,384,671,745	281,475,077,373,960	1 m
30	576,460,753,914,036,225	4,611,686,031,312,289,800	1 cm

面片和结点排列顺序实际是分布在球面上的一种空间填充曲线. 比较著名的空间填充曲线有行序、Peano曲线、Hilbert曲线以及Gray序[16~21] (如图 2). 通过空间填充曲线可将二维或多维空间的点依次转换到一维空间[16,17], 能较好地保持像素的空间几何形态. 目前所有填充曲线都基于二维或三维空间中的规则格子或者立体, 还没有看到对球面三角及其结点的相关研究.

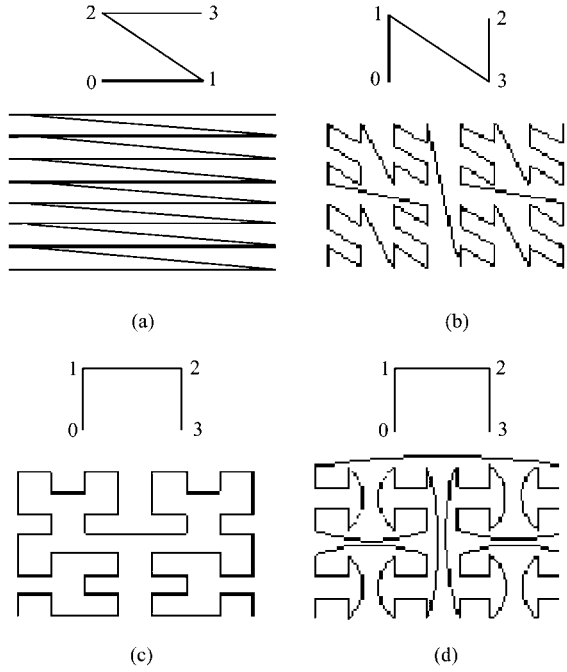

图 2 4 种空间填充曲线(从左到右)

(a) 行序, (b) Peano, (c) Hilbert, (d) Gray

本文采用了Lee[15]的面片编码模型, 直接把该模型编码的增长轨迹作为面片L空间填充曲线, 并在此基础上设计了结点L空间填充曲线(或称结点L序). 面片L曲线具有下面的性质:

(ⅰ) 兄弟面片编码连续.

(ⅱ) 编码基本反映了面片的空间分布.

(ⅲ) 少数堂兄弟面片编码差距很大, 以及编码连续的面片不相邻.

结点 L 序的性质如下:

(ⅳ) 兄弟面片的结点地址连续.

(ⅴ) 多数面片的结点距离较小, 少数面片很大.

与行序相比, 结点 L 序的性能并不如预期好, 因此提出了一种优化结点 L 序, 使得

(ⅵ) 所有面片的结点距离限制在可接受的范围内.

1　相关术语、面片编码模型及球面三角坐标系

1.1　术语定义

为论述方便, 首先定义若干术语:

(ⅰ) n簇剖分: 指对球面三角面n次四分剖分后产生的子三角面个数为 2^{2n}的球面三角网, $n\geq 0$.

(ⅱ) 基面片: n簇剖分中最小的球面三角面片. 在n簇剖分中, 按n-m深度的父面片的覆盖范围划分基面片可得到m簇剖分, 其中$m\leq n$且$m\geq 0$. m簇剖分包括2^{2m}个基面片.

(ⅲ) 面片方向: 构建球面三角四叉树剖分时, 球体内接 Plato 立体(如正八面体, 正二十面体等)的两个顶点分别与南北极重合, 0 级球面三角以及剖分过程中所产生的其他球面三角的一端或指向北极或指向南极. 对于前者, 称之为绝对正向, 后者为倒置向或绝对负向(参见图 3). 在剖分过程中, 父面片与子面片的方向不一定相同. 称与父面片方向相同的子面片为相对正向, 而相反的则为相对负向. 下面论述的所有操作都基于相对正向面片(相对负向面片可通过倒置转换为正向). 为简便计, 后文中所提及的正和负都指相对方向.

(ⅳ) 直系始祖: 从 m+1 (m>0)深度到 n 深度的所有父面片方向与基面片同向, 而 m 深度的父面片正好与之反向 (或者该深度不存在, m=0), 则称 m 深度的父面片为基面片的直系始祖.

(ⅴ) 完全父面片: 从 m+1 深度到 n 深度的 biBits (见 1.2 节)保持不变, 而在 m 深度发生变动, 称 m 深度的父面片为完全父面片. 例如 00-01-11-11-11(或 127)的完全父面片剖分深度为 2. 显然, n 深度 biBits 为 10 的基面片的完全父面片为其本身.

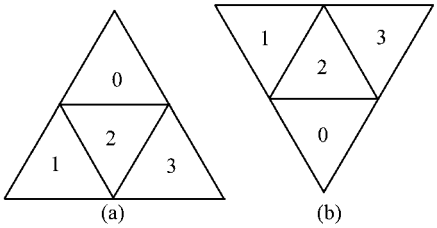

图 3 编码命名

(a) 正向, (b) 负向

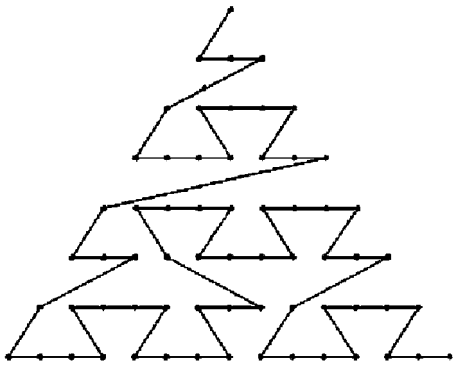

图 4 面片 L 序

n=4

1.2 面片编码模型

本文选择Lee[15]编码模型来命名三角面片(参见图 3). 对于正面片, 顶部子面片为 00(0)[1)], 底部左面片为 01(1), 底部中间面片是 10(2), 底部右面片为 11(3); 对于负面片, 底部面片为 00(0), 顶部左面片为 01(1), 顶部中间面片是 10(2), 顶部右面片为 11(3).

可以利用 64 位或者 32 位整型来表示编码(在这里, 不考虑 0 级剖分编码[2)]). 每一剖分深度占用 2 bit, 称之为biBits. 每次剖分后产生的biBits置于编码尾部. n 深度剖分产生的编码占有整型的后 $2n$个bit位, 从 1 级剖分biBits开始由高往低排列的第k个biBits为第k次剖分的编码值. 该编码模型具有下面的几个特点: (ⅰ) n簇剖分中所有面片的编码从 0 依次连续递增到 $2^{2n}-1$, 相邻编码差为 1. (ⅱ) 正负面片的子面片命名呈上下对称. 通过相互倒置, 两者可以统一为一种情况. (ⅲ) 父面片所包含的基面片的编码连续. 在该连续范围内的任何基面片都是该面片的孩子. (ⅳ) 编码为m的面片与m+1 面片相邻或者成中心对称. (ⅴ) 按照编码增大顺序连接所有的面片, 得到编码生长轨迹(见图 4), 可把该轨迹作为面片空间填充曲线. Dutton[7], Fekete[9,10]和Goodchild[12]等编码模型以中间的子面片作为起始点, 生长轨迹呈螺旋放射状, 并且存在自交现象. Lee编码模型生长轨迹总体上沿着从起始点经由最左边的面片向最右边的面片伸展的路线, 轨迹不自交. (ⅵ) 编码增长轨迹形状与剖分深度n无关. n+1 簇剖分与n簇剖分相比较, 编码为 $0\sim2^{2n}-1$ 的部分的轨迹一致, 而其余部分在底部(正面片)或顶部(负面片)一侧延伸. 本文直接采用面片生长轨迹作为面片L曲线(FLC).

1) 括号里的数字是对应的十进制值

2) QTM有 8 个 0 级剖分面片, SQT包括 20 个 0 级剖分面片

1.3　球面三角坐标系

L 曲线涉及两种球面坐标系, 包括面片坐标系和结点坐标系.

面片坐标系的原点为编码等于 0 的面片, Y轴穿越了n簇剖分最左一列上的面片, X轴沿着同Y值的面片伸展(如图 5(a)). n簇剖分的任意面片(X, Y)满足关系$\{(X, Y)|\ Y\in[0, 2^n-1], X\in[0, 2Y], n\geq 0\}$.

结点坐标系原点与编码为 0 的基面片的第一个结点重合, y轴与剖分簇左边界弧边重合, x轴穿越相同y值的所有结点(如图 5(b)). 对于n簇剖分的任意结点(x, y)满足关系$\{(x, y)|\ y\in[0, 2^n], x\in[0, y], n\geq 0\}$.

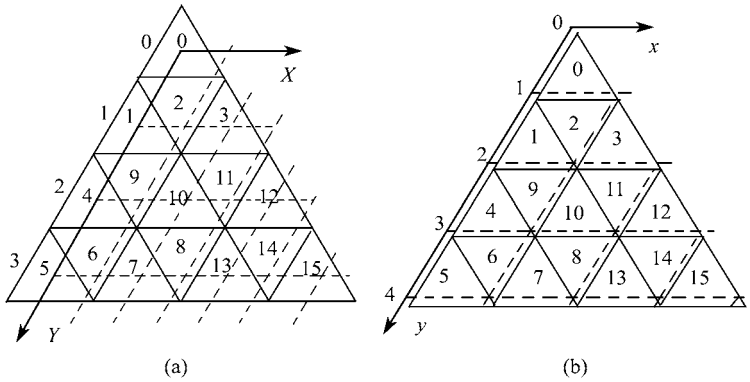

图 5　球面坐标系图

(a) 球面三角面片坐标系, (b) 球面三角结点坐标系

2　面片 *L* 序寻址

本文直接用编码增长轨迹作为面片空间填充曲线. $n=1$ 的剖分簇的轨迹形同“L”(见图 6, 负向为倒置的“L”). 因此称这种面片空间填充曲线为面片L空间填充曲线或L序. 在面片L序中, 面片按照编码从 0 到 $2^{2n}-1$ 依次排列. 面片L序的寻址包括计算面片(X, Y)在面片L序的偏移地址(即面片编码)以及其逆运算.

2.1　面片(X, Y)在面片 *L* 序的偏移地址

面片(X, Y)在面片 L 序的偏移地址等于该面片编码值, 该过程等同于计算该面片的编码值过程.

整个过程逐层计算面片从 1 级深度到 n 深度的 biBits. 每次计算后, 需根据该深度所在大面片的第 1 个基面片的位置更新坐标(X, Y). 该算法包括下面两步:

(ⅰ) 判断面片在m深度父面片的 4 个部分中的相对位置: 若$Y<2^{(n-m)}$, 则处于 00 子面片内, 令该深度对应biBits为 00 并转入(ⅱ); 若$X\leq 2Y-2^{(n-m+1)}$, 则在 01 子面片内, biBits为 01 并令$Y=Y-2^{(n-m)}$, 转入(ⅱ); 若$X<2^{(n-m+1)}$, 则在 10 面片内, 令 biBits为 10 以及$X=X-2Y+2^{(n-m+1)}-1$, $Y=2^{(n-m+1)}-Y-1$, 并转入(ⅱ); 其他情况则都在

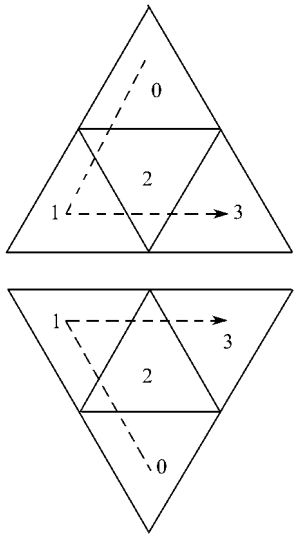

图 6 面片遍历顺序

11 子面片内, 令biBits为 11, $Y=Y-2^{(n-m)}$以及 $X=X-2^{(n-m+1)}$, 并转入(ⅱ).

(ⅱ) 若 $m=n-1$, 则转换过程终止并退出; 否则 $m=m+1$, 并跳至(ⅰ)进行下一深度分析.

2.2 偏移地址转换为面片三角坐标(*X*, *Y*)

偏移地址即编码 *code* 的每个 biBits 标示了面片在父面片的相对位置, 通过解析各个有效 biBits 即可计算出面片在面片三角坐标系中的坐标. 本文采用逐层解析法, 对每一层首先找到当前的父面片并计算第一个基面片的坐标, 并决定下一深度剖分的起始坐标. 编码 *code* 转换为 *n* 深度的面片三角坐标包括下面 3 步:

(ⅰ) 获取当前剖分深度 *m* 的 biBits 值 *segment*.

(ⅱ) 如果*segment*为 00, 则直接进行(ⅲ). 如果*segment*为 01，当相对方向[1)]为正时, 则令$Y=Y+2^{(n-m)}$; 当为负则令$Y=Y-2^{(n-m)}$且$X=X-2^{(n-m+1)}$. 如果*segment*为 10, 若为正向令$X=X-1+2^{(n-m+1)}$, $Y=Y-1+2^{(n-m+1)}$, 方向置负; 若方向为负则令$X=X+1-2^{(n-m+1)}$, $Y=Y+1-2^{(n-m+1)}$, 方向置正. 如果*segment*为 11, 当方向为正则令$X=X+2^{(n-m+1)}$, $Y=Y+2^{(n-m)}$; 当为负向时, 则令$Y=Y-2^{(n-m)}$.

(ⅲ) 如果 $m=n-1$, 则转换过程终止退出; 否则 $m=m+1$, 并跳至(ⅰ)进行下一深度分析.

3 结点 *L* 填充曲线及基本算法

结点 *L* 曲线的轨迹由面片 *L* 序以及结点取舍情况决定(见图 4 和 7).

3.1 结点 *L* 填充曲线

本文采用了如下结点遍历模式(如图 8 示): 正向面片中, 首先从顶部结点 0 开始, 然后经由底部左结点 1 到底部右结点 2; 负向面片中, 首先从底部结点 0 开始, 然后经由顶部左结点 1 到顶部右结点 2. 在正向面片, 结点遍历轨迹如同"*L*", 而负向面片呈倒置的"*L*". 因此, 称这种空间填充曲线为结点 *L* 填充曲线或结点 *L* 序. 图 7 为深度为 4 的结点 *L* 曲线轨迹.

1) 初始相对方向都为正

图 7　结点 L 曲线(n=4)

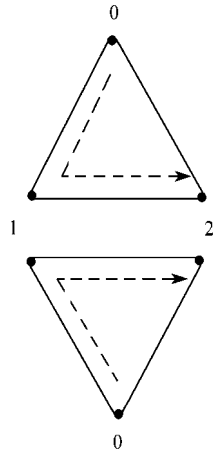

图 8　结点遍历顺序

3.2　结点访问规则

结点在 L 曲线中的地址唯一, 任何结点在遍历过程中只能被访问一次. 我们采取最小码优先访问规则来避免重复访问被相邻面片间共享的结点. 沿着面片 L 填充曲线, 编码值小的面片优先访问共享结点, 而该共享结点为后面的面片所不可访问. 优先访问结点的面片编码总是该结点邻接的所有面片编码最小的.

根据最小码优先访问规则, 可将面片分为 4 种类型(如图 9 示[1)]), 其中TYPE_ONE的 3 个结点都可访问, TYPE_TWO则只访问结点 1 和 2, TYPE_THREE中可访问结点 3, 而TYPE_FOUR所有三结点均不可访问.

基面片属于哪一种类型, 取决于面片在直系始祖中的相对位置以及直系始祖相对方向. n 深度的 biBits 为 10 的基面片直系始祖为其本身, 3 个结点都已被访问过, 属于 TYPE_FOUR 类型. 其他面片则分属 4 种类型.

TYPE_ONE 类型的面片为 n 簇剖分的第 1 个面片, 即当且仅当面片编码为 0. TYPE_TWO 类型面片位于 n 簇剖分的左侧第一列, 其 X 坐标等于 0 且不是 10 型基面片, 即 TYPE_TWO 类型面片编码各个有效, biBits 只能是 01 或 00, 但至少有一个是 01. TYPE_THREE 与 TYPE_FOUR 分布在 n 簇剖分内部和右侧. 区分两者的主要依据是直系始祖相对方向和面片与直系始祖相对位置. 如果直系始祖相对它的父面片方向为负, 说明最底部结点全部已被访问. 如果该面片正好位于直系始祖最底部, 则该面片为 TYPE_FOUR. 其他情况则都为 TYPE_THREE.

1) 图 9 和 10 中黑点表示该结点可访问, 非黑点结点为不可访问. 注: 在处理过程中面片方向始终保持相对正

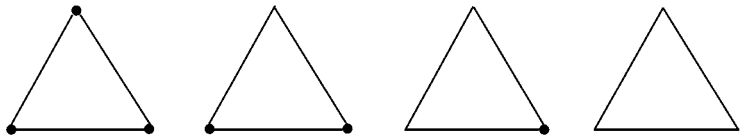

图 9 面片类型

(a) TYPE_ONE, (b) TYPE_TWO, (c) TYPE_THREE, (d) TYPE_FOUR

3.3 直系始祖及相对方向

直系始祖是与基面片保持同向距离最近的父面片. 00, 01 以及 11 型子面片方向与父面片一致, 而 10 子面片则与父面片相反. 因此, 离 n 深度最近的 biBits 为 10 的深度对应该面片的直系始祖, 而从该深度到 n 深度间的各个深度的 biBits 或为 00, 或 01 或 11. 例如编码 11-10-00-11-01-11(或 3607)的直系始祖深度为 2, biBits 为 10. n 深度的 biBits 为 10 的基面片的直系始祖为该基面片. 各个深度的 biBits 都不等于 10 的面片的直系始祖为 0 级原始面片. 显然, 当直系始祖深度 biBits 为 10 时, 直系始祖相对于它的父面片方向发生了倒置, 相对方向置为负; 若直系始祖为 0 级原始面片, 其方向与 0 级面片同向, 置正值.

3.4 面片与直系始祖相对位置计算

主要考虑基面片与直系始祖最底部一行面片的相对位置. 面片处于直系始祖最底部的充要条件是自直系始祖剖分以下深度开始, 该基面片的剖分位置或为底部左面片或为底部右面片, 即从直系始祖深度到 n 深度的所有 biBits 值或为 01, 或为 11. 不满足该条件的基面片都不在直系始祖最底部.

3.5 结点 L 曲线生成

Morton码可利用格子坐标(X, Y)交叉排列而得[19,20]. 但是结点L曲线比较复杂, 较好的方式是利用一维索引表来记录结点坐标(x, y)在L曲线中的偏移地址. 该索引表采用结点行序.

按照面片L序, 逐一遍历基面片的可访问的结点, 即可获得结点L曲线. 遍历每个结点的同时初始化索引表中对应值. 遍历从$code=0$ 的面片开始, 此时 (x, y)=(0, 0). 对于方向相对于 0 级面片为正的基面片, 3 个结点坐标分别为(x, y), (x, y+1)和(x+1, y+1), 而负面片则分别为(x, y), (x−1, y−1)和(x, y−1). 若该面片为 TYPE_ONE, 则依次遍历 3 个结点; 若面片为TYPE_TWO, 则只访问第 1 和 2 结点. 对于TYPE_THREE类型面片, 则只访问第 2 结点. 如果面片为TYPE_FOUR 基面片, 不访问任何结点. 结点(x', y')在行序中的偏移地址为 $\frac{(1+y')\times y'}{2}+x'$, 其

索引表对应项的值等于访问该结点前的已访问结点总数. 完成该面片遍历后, 需要确定下一面片起始结点的坐标值以及面片方向[1]. 假设目前所访问结点坐标为 (x, y).

(ⅰ) TYPE_ONE 面片: 下一个面片起始结点(x, y)为$(0, 1)$.

(ⅱ) TYPE_TWO 面片: 如果 n 深度 biBits 为 00, 则 $x=0$; 否则(x, y)保持不变但面片方向置负.

(ⅲ) TYPE_THREE面片: 设该面片的完全父面片对应剖分深度为k, 若$k\neq 0$且n深度的biBits为 11 以及k深度的biBits为 00, 则$x=x-2^{n-k}$; 如果n深度biBits为 00, 则令$x=x-1$; 其他情况下(x, y)保持不变但面片方向置反.

(ⅳ) TYPE_FOUR 面片: 如果方向为正, 则 $x=x+1$, $y=y+1$; 否则 $y=y-1$, x 保持不变. 两者方向都需置反.

基于单纯一个面片遍历效率较低. 在实际利用中, 可把 m 簇剖分作为遍历单位. 与基面片遍历相似, 根据结点访问模式可以把 m 簇剖分分为 4 种情况. 图 10 是 $m=1$ 时的 4 种情况. 采用 m 簇剖分不影响具体的算法.

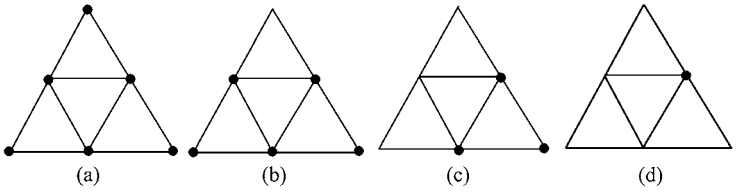

图 10　1 级面片类型

(a) TYPE_ONE, (b) TYPE_TWO, (c) TYPE_THREE, (d) TYPE_FOUR

3.6　面片结点的访问

3.6.1　基面片访问

结点 L 曲线不直接支持访问面片的 3 个结点. 比较方便的访问方式包括下面 4 步: (ⅰ)首先将 *code* 转换为球面三角面片坐标(X, Y); (ⅱ) 根据(X, Y)找到 3 个结点在球面三角结点坐标; (ⅲ) 将球面三角结点坐标系坐标转换为行序索引; (ⅳ)通过行序索引在索引表中找到结点地址, 从而访问 3 个结点.

3.6.2　访问$\boldsymbol{m}$深度父面片$\boldsymbol{P}$[2]的结点

结点L曲线中, 兄弟面片的编码是连续的. 因此, 首先求得P在n簇剖分中的第 1 个基面片编码 *start* 和以及底部最右基面片的编码 *end*, 其中 $start = P.\mathrm{SHIFT\_LEFT}.(2(n-m))$, $end = start + 2^{2(n-m)}$. 从*start*开始, 递增遍历所有基面片,

1) 初始化面片方向为正

2) P为父面片在m深度剖分簇中的编码

直到*end*为止.

4 *L* 序恒定时间算法

我们在Michael Lee[15]的基础上设计了基于基本位算子的优化算法.

4.1 基本算子

Schrack[22]利用AND和OR等基本位运算提出了邻近面片搜索时间恒定算法. Lee在其基础上定义了ABIDXY算子, 其中*A*, *B*, *X*和*Y*分别为 0 或 1, 该算子将编码中所有为*AB*的biBits置换为*XY*, 而其他biBits置为 00. 下面是摘自Lee的 00ID11 算法, 其他ABIDXY算子, 如 01ID11 等, 请参见相关文献.

```
path_array procedure MAKE_IDMASK_00ID11(p)
begin
  value pointer location_code P;
  path_array 00ID11;
  00ID11←OR(SHIFT_RIGHT(CODE(P)),CODE(P));
  00ID11←XOR(00ID11, EVENTBITMASK);
  00ID11←AND(00ID11, EVENTBITMASK);
  00ID11←OR(00ID11, SHIFT_LEFT(00ID11));
  return (00ID11);
end;
```

在此基础上, 提出了面片类型判断恒定算法时间优化算子. 恒定时间算法有利于硬件实现.

4.2 TYPE_TWO 判别算子

TYPE_TWO算子用于判断非零*code*是否为TYPE_TWO类型, 包括 3 步: (ⅰ) 对*code*进行 00ID11 操作, 标示出*code*里所有为 00 的biBits, 得到*a*; (ⅱ) 对*code*进行 01ID11 操作, 标示出*code*里所有为 01 的biBits, 得到*b*; (ⅲ) 令$c = a+b$. 如果c .EQ.MAX[1)], 表示该面片为TYPE_TWO类型.

表 2 为剖分深度为 5 的两个示例. 00-01-00-01-01(或十进制 69)的*c*等于 11-11-11-11-11($2^{10}-1$), 因此该面片为TYPE_TWO类型, 而 11-10-11-01-00(或 948)则不是.

表 2 TYPE_TWO 算子示例

code	00-01-00-01-01	11-10-11-01-00
(ⅰ)	11-00-11-00-00	00-00-00-00-11
(ⅱ)	00-11-00-11-11	00-00-00-11-00
(ⅲ)	11-11-11-11-11	00-00-00-11-11

1) *n*深度剖分的MAX等于 $2^{2n}-1$

4.3　TYPE_THREE 和 TYPE_FOUR 判别算子

TYPE_THRE 和 TYPE_FOUR 识别算子包括下面几步: (ⅰ) 对 *code* 进行 10ID10 操作, 得到 *a* 值. 如果 *a* 为 0, 说明直系始祖是 0 级原始面片, 面片底部结点可访问, 该面片为 TYPE_THREE 型, 结束该运算过程, 否则进行第 2 步; (ⅱ) 对 *a* 减 1, 即 *b*=*a*−1; (ⅲ) 对 *b* 取反码, 即 *c*=.NOT.*b*; (ⅳ) 对 *a* 和 *c* 进行或运算, 即 *d*=*a*.OR.*c*; (ⅴ) 得到直系始祖顶部第一个基面片编码, 令 *e*=*d*.AND.*code*; (ⅵ) 通过 *f*=*code*−*e* 得到直系始祖剖分深度以下的 biBits 部分; (ⅶ) 对 *f* 进行 00ID11 操作, 得到 *g*; (ⅷ) 令 *h*=*g*−*d*; (ⅸ) 将直系始祖剖分深度以下的 biBits 部分 00 位段置为 11, 即对 *h* 进行 11D11 操作得到 *j*; (ⅹ) 如果 *j*=0, 说明直系始祖剖分深度以下的 biBits 部分或为 01 或为 11, 可以判断该面片位于直系始祖底部且类型为 TYPE_FOUR; 如果 *j*≠0, 表明在直系始祖深度以下至少有一个 biBit 为 00, 该面片不在直系始祖的最底部, 因此面片类型为 TYPE_THREE.

表 3 为 3 个实例, 其中 11-01-00-01-01(或十进制 837)在第 1 步时 *a* 为 0, 该面片为 TYPE_THREE 类型, 从而提前结束计算过程; 而第 2 个编码 10-00-10-01-11(或十进制 551)第 9 步计算结果 *j* 值为 0, 该面片为 TYPE_FOUR 类型; 第 3 个编码 10-00-10-00-11(或十进制 547)的 *j* 值为 3, 因此该面片为 TYPE_THREE 类型.

表 3　TYPE_THREE 算子示例

步骤		*code* 输入		
		11-01-00-01-01	10-00-10-01-11	10-00-10-00-11
(ⅰ)	*a* = 10ID10 (*code*)	00-00-00-00-00	11-00-11-00-00	11-00-11-00-00
(ⅱ)	*b*=*a*−1	—	11-00-10-11-11	11-00-10-11-11
(ⅲ)	*c*=.NOT.*b*	—	00-11-01-00-00	00-11-01-00-00
(ⅳ)	*d*=*a*.OR.*c*	—	11-11-11-00-00	11-11-11-00-00
(ⅴ)	*e*=*d*.AND.*code*.	—	10-00-10-00-00	10-00-10-00-00
(ⅵ)	*f*=*code*−*e*	—	00-00-00-01-11	00-00-00-00-11
(ⅶ)	*G*=00ID11 (*f*)	—	11-11-11-00-00	11-11-11-11-00
(ⅷ)	*h*=*g*−*d*	—	00-00-00-00-00	00-00-00-00-11
(ⅸ)	*j* =11D11 (*h*)	—	00-00-00-00-00	00-00-00-00-11

5　结点 *L* 曲线与结点行序编码的比较

本文对结点 *L* 曲线与结点行序进行了对比.

面片结点排列距离反映了结点排列效率, 它等于从面片 3 个结点在填充曲线中的第一个结点到最后结点的结点个数. 行序按逐行从左到右顺序排列结点(见图 11). 行序排列的面片结点距离等于从第 0 结点按行序扫描至第 2 结点 (或者从第 1 结点扫描至第 0 结点)的结点个数. 同一行的面片的行序结点距离相等. 假如一个面片 3 个结点 *y* 坐标分别为 *i* 和 *i*+1, 则行序结点距离等于 *i*+2. *n* 簇剖分的基

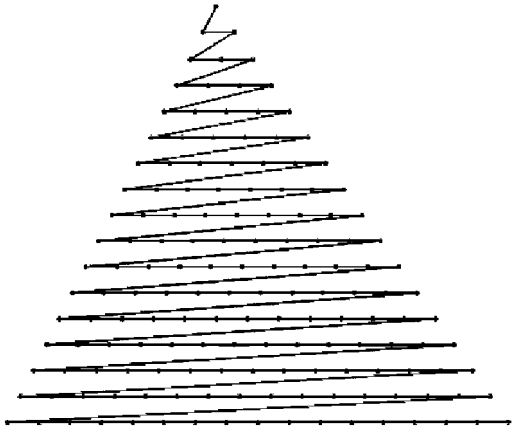

图 11 结点行序

面片结点行序距离和为

$$\sum_{i=0}^{2^n-1}(2i+1)(i+2).$$

图 12 和 13 分别为面片结点距离频率百分比统计以及结点距离频率累计百分比统计. 从图中可看出, 在结点 L 曲线中面片集中在低结点距离范围内, 而行序中具有低结点距离的面片数量较少而且面片数量随着距离增加呈线性增长, 距离越大, 面片的数量越大. 深度为 7 的结点 L 曲线中, 距离在 42

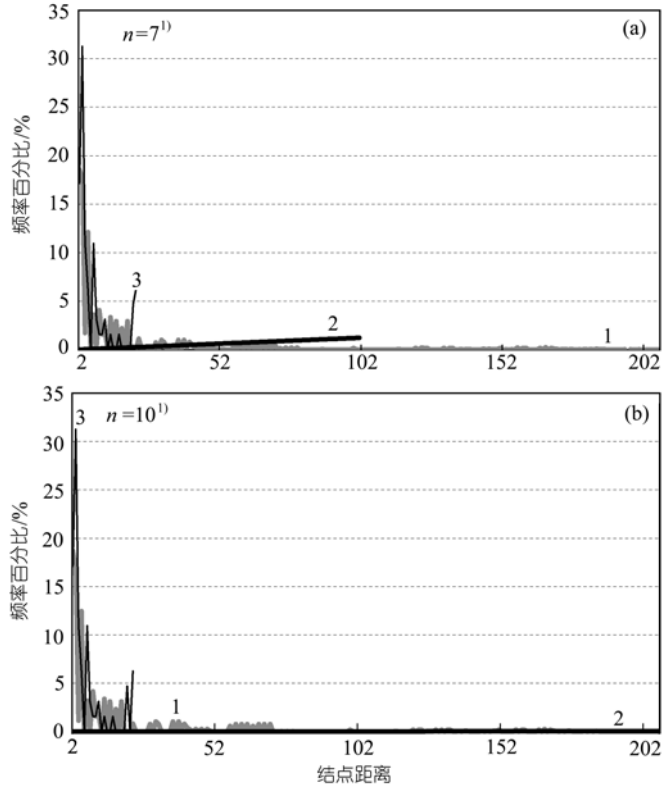

图 12 面片结点距离频率百分比统计

1 示L序, 2 示行序, 3 示基于 3 完备剖分簇的优化L序, n为剖分深度

1) 图中仅反映了结点距离在 0~207 以内的数据

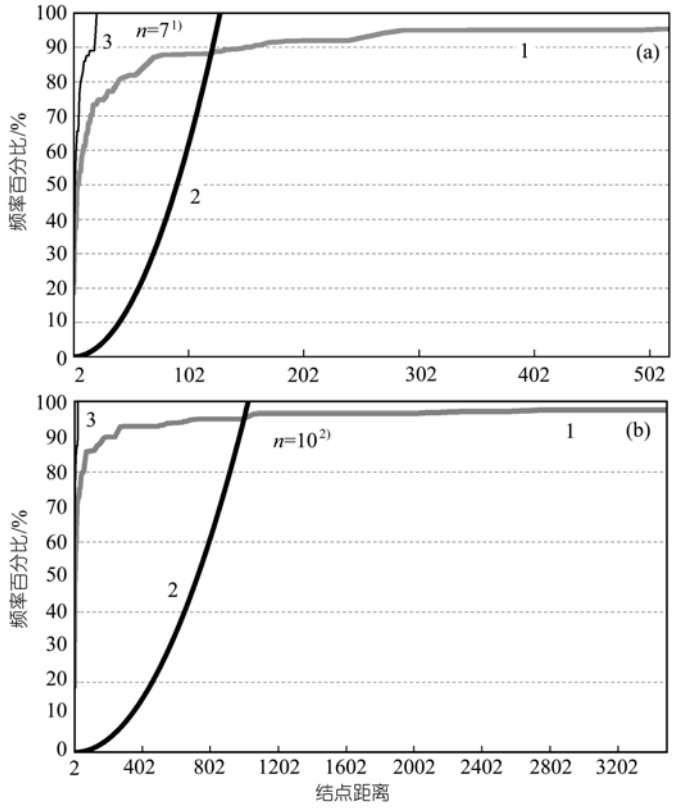

图 13　结点距离累计百分比统计

1 示L序, 2 示行序, 3 示基于 3 完备剖分簇的优化L序, n为剖分深度

以内的面片占整个数量的 80%, 而行序仅为 10%. 深度为 10 的结点 L 曲线中, 距离在 50 以内的面片占整个数量的 80%, 而行序仅为 0.22%. 因此相对行序, 结点 L 曲线具有低结点距离的特点.

图 14 反映了结点平均距离随剖分深度n的增长趋势. 当n=1, 2 时, 行序与结点L序的结点平均距离相等; 当n=3 时, 结点L曲线(6.6)小于行序(6.8); 但当n>3 后, 结点L曲线反而大于行序并且随着n增大其差距急剧加大. 因此, 从平均结点距离来看, 行序反而优于结点L曲线. 主要原因是有些堂兄弟面片编码不连续或者编码连续但面片呈中心对称分布, 相隔太大, 导致结点距离过大, 从总体上加大了结点平均距离. 比如, 在n=10 的剖分簇中, 0 级剖分中 00 面片与 10 面片边界相邻的基面片, 编码差至少为 2^{18}(262144), 最大为 2^{19}(524288), 最大结点距离达

1) 图中仅反映了结点距离在 0~519 以内的数据

2) 图中仅反映了结点距离在 0~3492 以内的数据

263167; 而行序相应的最大结点距离为 1025. 随着剖分深度加大, L曲线该

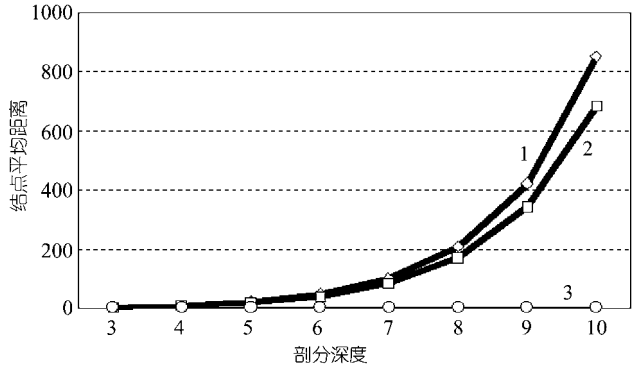

图 14 结点平均距离随剖分深度 n 增长趋势

1 示 L 序, 2 示行序, 3 示基于 3 完备剖分簇的优化 L 序

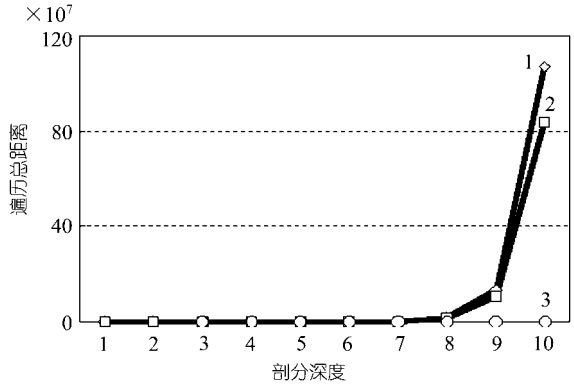

图 15 面片结点遍历总距离随剖分深度 n 增长趋势

1 示 L 序, 2 示行序, 3 示基于 3 完备剖分簇的优化 L 序

现象增多且差距加大, 从而导致平均结点距离超过行序. 这也是面片遍历总距离[1)]大的原因(参见图 15).

一方面结点 L 曲线面片普遍具有低结点距离, 另一方面由于少数面片的超大结点距离导致平均距离大于行序, 因此, 实际应用不能直接使用结点 L 曲线, 有必要对 L 曲线进行优化. 通过以 m 剖分簇作为独立的基面片结点排列基本单元($m<n$), m 剖分簇间的公共结点在各个 m 剖分簇中分别存储, 可将最大结点距离限制在 m 剖分簇内部, 从而保证了整个 n 簇的平均距离维持在稳定的水平上. m 剖分簇中的结点按照 L 曲线排列, 都存储一次, 而不论是否在其他 m 剖分簇中已经存储, 从而构成一个完备结点集. m 剖分簇的排列遵循面片 L 序, 以保证优化 L 曲

1) 沿着面片编码生长轨迹访问面片的 3 个结点所移动的距离总和，包括面片内部的结点距离和面片间的距离

线与 L 曲线的兼容. 本文试验了 m=3 完备结点簇, 相关性质如图 12~15 中的细线所示. m=3 完备结点簇的优化 L 曲线中, 最大结点距离为 23, 平均结点距离为 6.6, 面片遍历总距离随着剖分深度的增长趋势远远低于行序和 L 曲线. 因此, 通过优化结点 L 序, 可以有效地提高结点访问效率.

6　结论

通过上面的论述, 了解了 L 曲线的优缺点: 普遍面片结点距离小, 同一父面片的子面片结点距离连续; 但结点距离变异大, 导致平均结点距离和面片遍历总距离等总体性能指标反而不如行序. 基于 m 剖分簇完备结点集的优化 L 曲线通过重复存储公共结点, 减少了结点距离变异, 从而避免了 L 曲线的缺陷, 同时保持了 L 曲线优点.

但是完备结点集中边界结点的重复存储存在存储空间的浪费. n剖分簇m剖分完备结点集, 总结点数量为$(2^{2m-1}+3\times2^{m-1}+1)\times2^{2(n-m)}$. 考察n=10 的情况. 当m=0 时, 结点总数与n剖分簇结点数比例达到 6 倍. 随着m增大, 该比例逐渐减少. 当 m=1 时, 该比例降为 3; m=2 时接近 1.9; m=3 时比例约为 1.4; m=4 时比例约为 1.2; 当m=n时, 该比例降为 1. 因此, m的增大对额外存储空间的需求影响逐渐降低, 当m由 0 到 1 过渡时, 效用最大. 因此, 选择m值时需要同时考虑存储空间耗费与结点距离随m变化的边际效用.

利用优化 L 曲线作为全球 DEM 数据组织基础, 构建了一个虚拟三维地球系统, 完备结点集选择 m=3, 取得了较好效果. 将在后续论文中作相关介绍.

参　考　文　献

1 White D, Kimerling J, Overton W. Cartographic and geometric components of a global sampling design for environment monitoring. Cartography and Geographic Information System, 1992, 19(1): 5 ~ 22

2 White D, Kimerling J, Sahr K, et al. Comparing area and shape distortion on polyhedral-based recursive partitions of the sphere. Int J Geographical Information Science, 1998, 12(8): 805 ~ 827[DOI]

3 Kimerling J, Sahr K, White D, et al. Comparing geometrical properties of global grids. Cartography and Geographic Information Science, 1999, 26(4): 271 ~ 288

4 Sahr K, White D. Discrete global grid systems. Computing Science and Statistic, 30. Weisberg S, ed. Farfax Station: Interface Foundation of North America Inc, 1998

5 Goodchild M. Discrete global grids for digital earth. International Conference on Discrete Global Grids. California: Santa Barbara, 2000

6 Saff E B, Kuijlaars A B. Distributing many points on a sphere. The Mathematical Intelligencer, 1997, 19(1): 5~11

7 Dutton G. A hierarchical coordinate system for geoprocessing and cartography. Lecture Notes in Earth Science 79. Berlin: Springer-Verlag, 1998. ISBN 3-540-64980-8

8 Dutton G. Universal geospatial data exchange via global hierarchical coordinates. International Conference on Discrete Global Grids, California: Santa Barbara, 2000

9 Fekete G. Rendering and managing spherical data with sphere quadtrees. Proc Visualization'90, New York: ACM, 1990. 176 ~ 186
10 Fekete G. Sphere quadtrees: A new data structure to support the visualization of spherically distributed data. SPIE, Extracting Meaning from Complex Data: Processing, Display, Interaction, 1990, 1259: 242 ~ 253
11 Song L, Kimerling J, Sahr K. Developing an equal area global grid by small circle subdivision. International Conference on Discrete Global Grids, California: Santa Barbara, 2000
12 Goodchild M, Yang S. A Hierarchical coordinate system for geoprocessing and cartography. CVGIP: Graph, Models Image Process, 1992, 54(1): 31 ~ 44
13 Otoo E J, Zhu H. Indexing on spherical surfaces using semi-quadcodes. Advances in spatial Databases. Proc 3rd Int Symp SSD93, Singapore, 1993. 510 ~ 529
14 Lugo J, Clarke K C. Implementation of triangulated quadtrees sequencing for a global relief data structure. Proc Auto Carto 12 ACM/ASPRS, 1995. 147 ~ 156
15 Lee M, Samet H. Navigating through triangle meshes implemented as linear quadtrees. ACM Transactions on Graphics, 2000, 19(2): 79 ~ 121[DOI]
16 Gaede V, Günther O. Multidimensional access methods. ACM Computing Surveys(CSUR), 1998, 30(2): 171 ~ 231
17 Sagan H. Space-Filling Curves. New York: Springer-Verlag, 1994. ISBN: 0-387-94265-3
18 Samet H. The quadtree and related hierarchical data structure. ACM Computing Surveys, 1984, 16(2): 187 ~ 260[DOI]
19 Samet H. The design and analysis of spatial data structures. Reading, MA: Addision-Wesley, 1989
20 Samet H. Applications of spatial data structures. Reading, MA: Addision-Wesley, 1990
21 Samet H, Webber R E. Storing a collection of polygons using quadtrees. ACM Trans Graphics, 1985, 4(3): 182 ~ 222[DOI]
22 Schrack G. Finding neighbors of equal size in linear quadtrees and octrees in constant time. CVGIP: Image Underst, 1992, 55(3): 221 ~ 230