

利用双重索引快速构建道路网络连通拓扑

撒志恒, 芮小平*, 宋现锋, 刘真余, 王 静

(中国科学院大学资源与环境学院, 北京 100049)

摘要:路网拓扑关系的生成是进行最优路径规划的基础。本文针对 ISO GDF4.0 模型对道路连通拓扑的定义, 结合最优路径规划对道路网络连通拓扑的要求, 提出一种使用 R-tree 空间索引和 B-tree 索引双重索引方式快速生成道路连通拓扑的算法。连通拓扑快速构建算法包括新道路生成和网络拓扑提取两部分, 新道路生成过程中, 首先, 自上而下地打断道路形成直线段集并求交点, 然后, 自下而上地重构直线段集以生成新道路。在打断道路求交点过程中, 对道路建立 R-tree 空间索引, 显著提高了几何要素的查找速度。在网络拓扑提取过程中对序列化数据建立 B-tree 索引, 使得其查找速度大大加快。通过对双重索引算法的时间复杂度分析与验证表明, 本文提出的拓扑生成算法具有较高的执行效率。

关键词:道路网; 连通拓扑; 双重索引; R-tree; B-tree; 有效算法

DOI: 10.3724/SP.J.1047.2013.00498

1 引言

最优路径规划是交通地理信息系统的研究热点, 其核心是最短路径算法^[1], 其实质是寻找道路网络中两点之间的“最短路径”(即以一种或多种指标计量的耗费最小的路径)^[2]。执行最优路径规划的前提是道路网络已经建立正确的拓扑, 因此, 构建道路网络拓扑是实现最优路径算法的关键。

国际标准化组织(International Standardization Organization, ISO)制定的导航地理数据概念模型国际标准草案(ISO GDF4.0)中, 对道路网络拓扑关系给出了明确的定义^[3]。GDF4.0 根据不同的应用需求, 定义了道路网基本表达单元(结点、边与面)之间3种不同的拓扑关系, 即道路网完全拓扑、连通拓扑和非显式拓扑。完全拓扑使用面表示多边形要素, 明确定义道路网中所有点线面基本表达单元之间的拓扑关系; 连通拓扑使用结点和边表示零维对象(结点)和一维对象(边), 明确定义结点和边的拓扑关系, 不定义面和边之间的拓扑关系; 非显式拓扑不定义基本表达单元之间的拓扑关系, 空间要素间的拓扑关系只能通过坐标值计算得到^[4-5]。根据 GDF4.0 对连通拓扑的定义, 道路网络中所有要

素使用点和边表示, 道路的交汇表示为零维的点, 道路抽象表示为一维的线, 道路的长度、通行能力、车道数等特性作为属性存储, 这种道路表达模型是本文提出算法的基础。

国内外学者对道路的多边形拓扑构建算法研究较多^[6-8], 主要研究拓扑的完备性。多边形作为完备拓扑结构中的基本单元, 在空间查询、分析方面有重要作用。但在进行路径规划时, 主要使用道路连通拓扑信息, 即边与结点之间的拓扑关系, 而不是多边形拓扑提供的信息, 故高效快速地构建道路的连通拓扑即可满足大部分最优路径规划的需要。

目前, 对连通拓扑的研究集中在构建算法上, 何超英在构建导航数据库时主要采用连通拓扑, 并补充部分完全拓扑^[9], 其他研究者在构建道路网络拓扑时多采用类似连通拓扑的结构^[5,9-13]。这些研究多是从构建算法出发, 并没有考虑算法的执行效率问题, 也没有提出针对连通拓扑构建过程的加速技术。针对算法执行效率的改进问题, 孙棣华提出三类典型的道路的相交模型, 然后对实际的道路相交情况进行类型划分, 统计类别信息作为处理道路交点的辅助信息, 以此来提高处理速度^[2]。使用有关交点类型的辅助信息可以简化求交点算法的复杂

收稿日期: 2013-01-21; 修回日期: 2013-03-15.

基金项目: 国家科技支撑计划项目课题(2012BAC25B01)。

作者简介: 撒志恒(1988-), 男, 河南平顶山人, 硕士生, 研究方向为地理信息科学应用。E-mail: ronald.han7@gmail.com

*通讯作者: 芮小平(1975-), 男, 江苏苏州人, 博士, 副教授, 研究方向为地理信息科学理论与应用。E-mail: ruixp@yahoo.com.cn

度,但计算道路网中交点与典型相交模型的相似性又会增加时间开销。陆守一提出使用外包矩形判断道路相交的可能性,再对道路进行求交^[14]。外包矩形相交判断能降低求交点运算的次数,但并没有减少一条道路的外包矩形与其他道路外包矩形的比较次数。徐立对道路建立格网索引加速连通拓扑构建^[15]。对道路建立空间索引加速相交判断是合理的思路,格网索引能在一定程度上加快查询速度,但整体而言,并没有明显降低查询的时间复杂度。在空间索引中,相比格网索引,R-tree可以更低的时间复杂度完成查询几何对象的任务。本文根据R-tree索引和B-tree索引的特点,在构建道路拓扑结构时,采用R-tree索引和B-tree索引的双重索引算法提高构建道路拓扑的效率。

2 采用双重索引算法快速构建路网拓扑结构

构建道路网拓扑中关键是将道路在交点处打断,传统的算法通过遍历所有道路求出一条道路与其他道路的交点,在对大规模路网进行道路求交时,道路数目巨大,往往需要耗费较长时间才能完成求交处理。改进措施主要是对道路建立索引,提高查询速度。常用的索引包括B-tree索引、格网索引、四叉树索引及R-tree系列索引^[16-19]等。采用道路的外包矩形对两条道路是否可能相交进行初步的判定,此种方式可以在一定程度上降低求交点运算执行的次数,但没有减少一条道路的外包矩形与其他道路外包矩形的比较次数,查询几何对象的时间复杂度为 $O(n)$ (n 为几何对象的数目)。格网索引将研究区域按照一定的间隔划分为大小相同的 $m \times n$ 个格网,格网记录落入其中的几何对象的信息,通过查询格网记录的几何对象信息即可搜索出所要的空间目标。格网索引是一种多对多索引,一个几何对象可能存储在多个格网中,一个格网也可能存储多个几何对象,格网索引的查询效率不优于 $O(mn)$ 。四叉树索引在构建过程中对数据空间进行四等分,当数据分布不均匀时,产生的空间索引四叉树是严重不平衡的,树的高度会比较高,查询效率达不到 $O(\log(n))$ ^[20]。

B-tree是一种多叉平衡树,适用于重复率低的序列化数据的查询,度为 d 的B-tree,其高度上限为 $\log_d((N+1)/2)$ (N 为索引的数据量),检索任意值的

时间复杂度为 $O(\log_d(N))$ 。B-tree可以高效完成一维空间查询,但并不适合线段或多边形这类二维或多维数据的查询,R-tree索引更适合多维数据的查询,它是B-tree在高维空间的扩展,也是一种多叉平衡树,其查询效率为 $O(\log_d(N))$ 。

R-tree索引通过设计虚拟矩形目标,将空间目标包含在相关的矩形内作为空间索引,其组成包括叶结点和非叶结点,数据结构为 $Entry(I, identifier)$,其中, I 表示一个 n 维空间的最小外包矩形,在二维空间中 $I=(I_x, I_y)$, $I_x=(X_{\min}, X_{\max})$, $I_y=(Y_{\min}, Y_{\max})$, $identifier$ 是标识符,在叶结点中指向其包含几何对象的实际存储位置,在非叶结点中表示指向孩子结点的指针。如下图1所示,L1和L2为两条道路,L1由A、B、C三条直线段构成,L2由D、E、F三条直线段构成。每条直线段的外包矩形如图1(a)所示,构建R-tree索引过程中需要依据直线段的外包矩形生成互不重叠的上一层最小外包矩形,如图1(b)中G、H、I所示,图1(c)为建立的R-tree索引。

在构建拓扑的线段求交过程中,主要有两类查询,其一是几何对象的查询,用于判定一条线段与其他线段是否相交,其二是几何对象编号的查询,用于判定求交的两条线段编号是否相同,这两类查询的效率,直接决定了算法效率的高低。对几何对象的查询,如果不对线段采取任何加速措施,查询线段的时间复杂度为 $O(n)$,对线段构建R-tree索引可以将查询时间降低到 $O(\log(n))$;线段编号是一种序列化数据,能够唯一标识一条线段,对序列化数据建立B-tree索引则将在此类数据上的查询时间降低到 $O(\log(n))$,由此可以提高算法执行效率。双重索引算法构建拓扑主要包括2个部分:“自上而下”拆分原道路;“自下而上”构建新道路。

(1)“自上而下”拆分原道路

一条道路一般由两个以上点组成,如图2(a)所示为一条原始道路,图2(b)中的蓝色点表示道路存储过程中使用的点,这些点依次连接形成图2(a)中原始道路的形态。“自上而下”拆分原线段是将道路按照结点存储顺序,两两连接,将道路细分为一组只有两个点组成的直线段。如图2(a)所示的道路,存储这条道路使用了8个点,将这8个点依次两两连接形成7条直线段,如图2(c)所示。拆分后,所有的线段都是直线段,原有道路细分为直线段集合。一条直线段与另外一条直线段(不同的直线段)之间最多有一个交点,这大大简化求交点使用

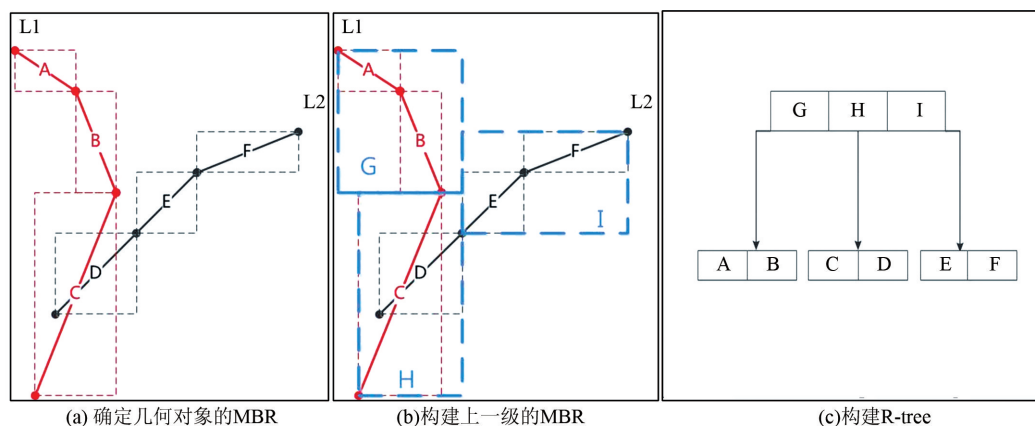


图1 构建R-tree过程

Fig.1 The procedure of building R-tree index on roads

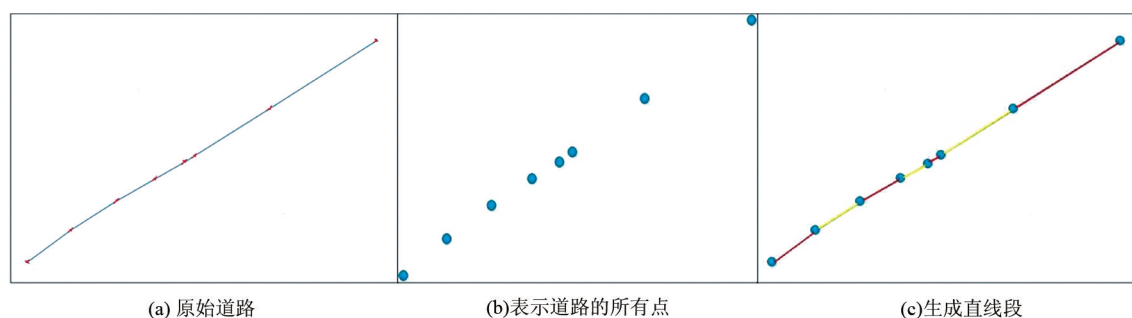


图2 将道路线拆分为直线段

Fig.2 Break the road into line segments

的算法,也提高了拓扑构建算法的效率。

(2)“自下而上”构建新道路

对拆分后形成的直线段建立R-tree索引,然后判断直线段是否相交,如果相交则返回交点,否则返回空值。直线段求交过程中,首先判断两个直线段的外包矩形是否相交,利用已建立的R-tree索引,将其中一条直线段的外包矩形作为R-tree查询的输入,如果查询结果不为空,则两条直线段的外包矩形相交,否则直线段的外包矩形不相交,不再进行求交点判断;如果外包矩形相交,再进行直线段的相互跨立判断,如果通过,则两条直线段必有交点,

调用求交点算法即可获取交点坐标,求交点算法详见文献[21]。对每条道路的交点情况作统计,使用道路的起始结点和结束结点以及其与其他道路的交点,将道路在交点处打断,依次连接起始结点、交点和结束结点形成新的道路。如下图3(a)所示,原始的两条道路相交,图3(b)表示在完成“自上而下”拆分原线段之后两条道路相交的形态,此时红色的交点是由两条道路上的直线段相交产生,记录交点的坐标,以及相交的两条直线段在原道路拆分后形成的直线段集合中的索引,便于对道路的交点进行排序。图3(c)表示将道路按照排序后的交点打断

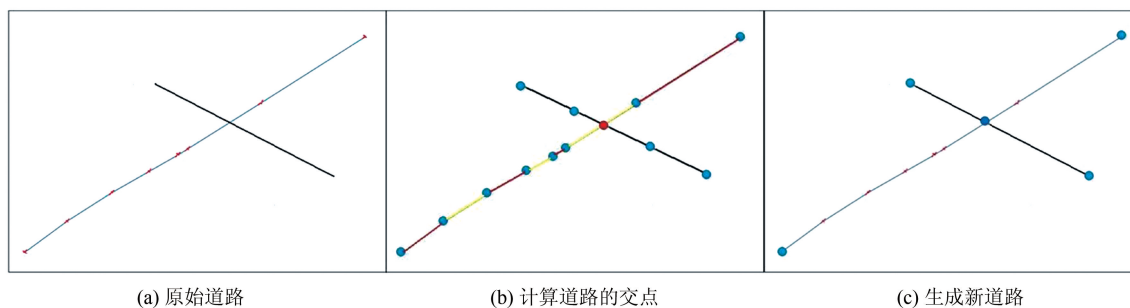


图3 根据道路交点生成新道路

Fig.3 Reconstruct new roads with intersections

形成新的道路。

2.1 算法定义

为了描述方便,有如下定义:

- (1) L_i : 道路图层的第 i 条道路。
- (2) $TypeFlag$: L_i 是否为平面道路的标记。
- (3) Ps : 组成 L_i 的点的集合。
- (4) N_i : L_i 存储点的个数。
- (5) I_i : L_i 所产生的交点集合。
- (6) Ts : 临时点结构体数组, 点的定义如下, Ts 表示为 $Point[]$ 。

(7) NLi : 新生成的道路中的第 i 条道路。

(8) 定义点结构体如下:

Struct Point

```
{
    Double x; Double y;
};
```

2.2 算法详细过程

(1) 新道路生成算法设计

算法的步骤和伪代码如下所示:

① 获取 L_i , 解析出表示 L_i 所存储点的坐标, 为 N_i 赋值, 并将点坐标依次存放入点结构体数组 Ps 中;

② 遍历 Ps 数组, 读取当前点和下一个点的坐标, 生成直线段;

③ 所有道路均拆分为直线段后, 对所有直线段创建 R -tree 索引;

④ 对直线段两两求交点, 以直线段的外包矩形是否相交作为判断条件, 降低进行求交点运算的次数, 对直线段进行查询时使用 R -tree 索引, 提高查询效率。通过判断两条直线段产生交点集合是否为空, 将外包矩形虽然相交, 但没有产生交点的记录剔除。通过判断直线段所属道路编号是否相同, 将简单线段与自身相交的记录剔除;

⑤ 统计 L_i 所有的交点, 并且按照交点所属直线段在 Ps 中的索引进行升序排列;

⑥ 获取 L_i 的起始结点和终止结点, 查询 L_i 的交点记录, 创建临时点结构体数组 Ts 。判断 $TypeFlag$ 的类型, 如果为平面类型, 将 Ps 数据复制到 Ts , 依次判断起始结点、交点、终止结点是否在 Ts 中, 如果不在, 按照交点指示的索引, 将点插入到 Ts 中, 并使用数组 $vIndex$ 记录插入点的位置; 如果为非平面类

型, 则只将起始结点和终止结点插入到 Ts , 在 $vIndex$ 中记录插入点的位置。

⑦ 遍历 $vIndex$, 获取当前记录所指示的点的索引和下一条记录所指示的点的索引位置, 从 Ts 中读取两个索引之间的点(包括索引指示的点), 将这些点数据生成新道路并存储, 完成新道路的构建。

算法流程图如下所示:

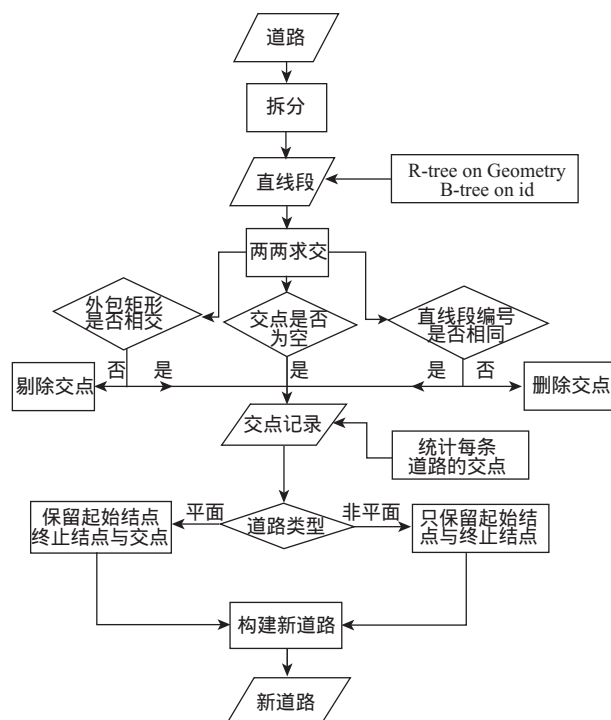


图4 算法流程

Fig.4 Flow chart of the algorithm

(2) 道路网络拓扑提取算法

道路在交点处打断生成新道路后, 为了构建道路网络拓扑还需要为道路网络中的结点赋予唯一的编号, 赋予编号之后即可通过分析点与边的连接关系建立点-边拓扑关系和边-点拓扑关系。算法执行需要维护一个临时表用于存放编号和点坐标。

构建道路网络拓扑的算法步骤如下:

① 遍历 NLi , 增加2个字段, 分别存放新道路的起始结点编号和终止结点编号。

② 获取 NLi , 检测 NLi 的起始结点和终止结点在临时表是否有记录, 如果没有记录, 为结点分配编号, 如果有记录, 将表中记录的点的编号赋予检测点, 获得结点编号后, 获得起始结点和终止结点编号后, 将其记录到 NLi 的属性中。具体过程通过以下示例说明, 图5中有A、B、C、D四条道路, 结点包含道路的起始结点和终止结点。

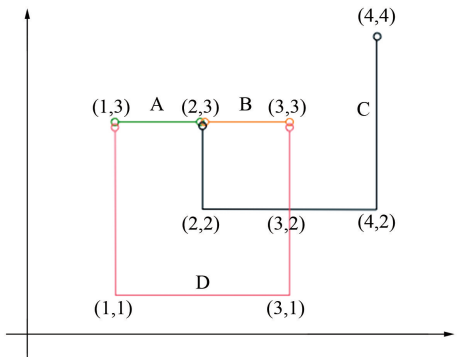


图5 简单道路示例
Fig.5 An example of simple road

分配结点编号时取出道路A,起始结点为(1,3),临时结点表没有此点的记录,为这个点分配编号为1,并将记录添加到临时结点表中。道路A的终止结点为(2,3),临时结点表没有此点的记录,为这个点分配编号为2,并将记录添加到临时结点表中。对于道路B,起始结点为(2,3),临时结点表有此点的记录,因此,道路B的起始结点编号为2,依次执行直至遍历所有道路。同时,为道路新添加的两个字段用Source和Target表示,为每一条道路的起始结点和终止结点分配编号,如表1所示。

表1 起始/终止结点编号 Tab.1 IDs of start/end vertex		
EdgeId	Source	Target
A	1	2
B	2	3
C	2	4
D	1	4

道路网络拓扑通常建立边-结点关系来表示道路的连接关系,表中存储每条道路起始结点与终止结点编号,以及表示道路的所有点的坐标。而在进行最优路径规划时,需要搜索一个结点所连接的边的信息,建立结点-边关系表(如表2所示),表中记录每个结点所关联的边的数量及编号,在搜索与特定结点关联边时,只需要在表中找到此结点即可获取关联的所有边,大大减少了遍历时间。

建立结点-边关系表时,统计某一结点连接的边的编号等需要遍历临时结点表和起始/终止结点编号表,并且主要集中在对结点编号和道路编号的查询,这类编号具有序列化的特征,并且重复率较低,因此,对这些编号建立B-tree索引,可以获取较高的查询效率。

表2 结点-边表

Tab.2 Vertex-Edge table

VertexId	结点编号
ConnEdgesNum	连接的边的数量
ConnEdgesId	连接的边的编号
Point(几何类型)	点的地理坐标

③ 遍历NLi,按照表2的结构,提取各字段的值。

3 算法时间复杂度分析

生成新道路算法主要的时间耗费在直线段求交,设经过拆分操作生成的直线段的数目为 n ,算法需要将任一直线段与其他所有直线段进行比较,判定其外包矩形是否相交,假设建立R-tree索引使用的最少子节点数为 m ,最坏情况下的查询时间为 $m \log_m(n)$,直线段求交的时间复杂度为 $nm \log_m(n)$ 。实际中,为直线段创建R-tree索引耗费的时间远少于直线段求交,按照文献[22]中提出的R-tree构建算法,插入新结点最多需要遍历至当前叶子结点,时间复杂度不超过 $\log_m(n)$ 。

其他步骤,如步骤一、二和步骤七需要访问道路网络存储的所有点,假定道路网络存储点数目为 N ,步骤一、二和步骤七的时间复杂度为 $O(N)$,考虑其他步骤的时间复杂度,整个算法复杂度为 $3O(N)+(nm+1)\log_m(n)$,拆分后的直线段数目 n 和存储点数目 N 并无确定的数值关系,在实际测试中发现 $O(N)$ 远小于 $\log_m(n)$ 。故算法总时间复杂度可以简写为 $(nm+1)\log_m(n)$ 。

设新生成道路图层中道路数目为 v ,道路网络中结点数目为 p ,则 p 的取值范围为 $[v,2v]$,分配编号的时间复杂度不超过 $(\log_m 2)v \log_m v$;第二步中,以建立结点-边表为例,需要搜索每个结点所连接的边的编号。现实中每个结点连接的数目不同,设定点的平均连接数为 c ,在有B-tree索引情况下,建立结点-边表的时间复杂度为 $c \log(v)$,一般而言 $c < v$,故道路拓扑构建算法的整体时间复杂度为 $v \log(v)$ 。

4 算法的应用测试

软件环境: Microsoft Visual Studio 2005, Windows XP Professional (32位/Service Pack 3)。

硬件环境: Intel Core™ i5-2540M 2.60GHz, 4G

DDR3 内存, TOSHIBA 640G 硬盘 7200 转/分钟。

使用文中算法对北京市道路进行拓扑关系构建测试, 原图为 shapefile 格式, 来源于项目资料, 包含快速主干道、国道、环路和 3 级道路。道路属性信息包含道路类型, 根据道路是否离地将所有道路标记为平面道路或非平面道路, 分别用 0 和 1 标记。从道路网络中提取出不同规模子网, 对子网执行双重索引算法, 结果如图 6 所示。

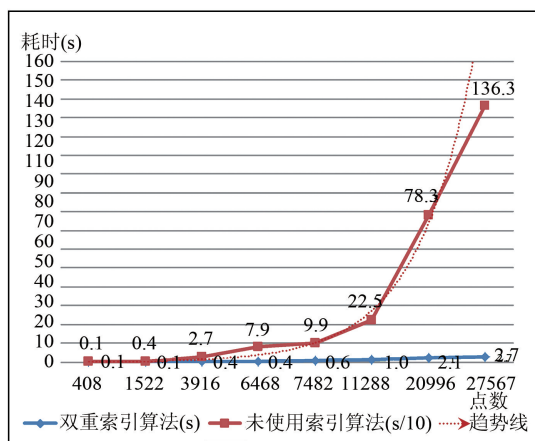


图 6 测试结果表

Fig.6 Experiment results

在图 6 中, X 轴表示算法处理的道路所包含的点数, Y 轴表示算法耗时, 单位为 s。由于未使用索引算法耗时较长, 为将其与双重索引算法进行对比, 将未使用索引算法耗时除 10。处理后, 图中虚线表示未使用索引算法耗时的增长趋势线, 拟合曲线为二阶函数。从图 6 中可以看出, 随着处理点数的增加, 两种算法耗时也逐步增加, 但增长速率并不同。在点数量较小时两种算法耗时差距并不太大, 大体处于同一量级, 随着点数不断增长, 两种算法耗时差距迅速增大, 未使用索引算法耗时增长呈平方阶, 而双重索引算法耗时增长平稳, 基本符合对数阶曲线。可见, 基于 R-tree 索引和 B-tree 索引的双重索引算法可以极大提高构建拓扑时间效率。

5 结论

近年来, 以道路网为基本信息衍生的各种交通服务应用方兴未艾, 如路径规划系统、物流配送系统、各类应急系统等。这些应用服务均以完善准确的道路网络拓扑作为数据基础, 而大规模道路网络拓扑构建通常耗时较长, 不能满足实时动态路径规

划等应用的需求。本文提出了一种改进的道路网络拓扑快速构建算法, 将道路打断形成直线段集, 对直线段集建立 R-tree 索引, 利用 R-tree 空间索引提高道路求交的效率。对序列化数据构建 B-tree 索引, 提高拓扑关系生成效率。针对高架桥、天桥、涵洞等非二维实体对拓扑关系构建的影响, 算法将所有道路进行类型标记, 在构建拓扑时根据道路标记类型不同分别进行处理, 能建立准确可靠的道路连通拓扑, 适合于较大规模网络道路连通拓扑的构建。对算法的时间复杂度进行理论分析, 证明本文算法的时间复杂度为 $O(v \log(v))$, 较传统算法的时间复杂度 $O(n^2)$ 而言, 时间效率有了明显改善。本文通过使用不同规模的道路网络进行实证表明, 改进算法的耗时随道路网络规模的扩大呈稳定平缓增长, 与时间复杂度理论分析结论一致。

本文实验发现, 改进算法需要进行数据预处理, 消除未及、过伸和道路不相交等数据错误, 不能自动处理自相交的道路, 这需要修改算法以提升健壮性。此外, 本文构建的拓扑是静态的, 现在的应用往往需要频繁修改道路的连通属性, 如何通过对道路网的分区实现道路拓扑的实时动态更新, 有待深入研究。

参考文献:

- [1] 李英姿, 张飞舟, 林耀海. 智能交通系统中地理信息系统的研究[J]. 中国公路学报, 2000, 13(3): 97-100.
- [2] 孙棣华, 肖锋, 廖孝勇, 等. 基于预处理的都市路网拓扑结构构建算法[J]. 计算机工程与应用, 2008, 44(23): 233-235.
- [3] 蒋捷, 韩刚, 陈军. 导航地理数据库[M]. 北京: 科学出版社, 2003.
- [4] 何超英, 蒋捷, 韩刚, 等. 基于 GDF 的道路网完全拓扑生成算法[J]. 地理与地理信息科学, 2004, 20(2): 30-33.
- [5] 赵斌. 导航地理数据生产系统及其关键技术研究[D]. 郑州: 解放军信息工程大学, 2006.
- [6] 张小国, 王庆, 王宁, 等. 电子地图道路网模型及其自动生成算法研究[J]. 中国图象图形学报, 2001, 6(5): 481-485.
- [7] 李杰, 张文栋, 张樾. 一种多边形道路网络拓扑生成算法的设计与实现[J]. 电子学报, 2006, 34(8): 1396-1400.
- [8] 王杰臣. 多边形拓扑关系构建的栅格算法[J]. 测绘学报, 2002, 31(3): 249-254.
- [9] 熊少非, 赵丕锡, 李军. MapInfo 中城市道路网络拓扑结构的自动生成[J]. 兵工自动化, 2005, 24(3): 67-71.
- [10] 宋维佳, 张丽芬, 王晓华, 等. 一种基于骨架化的道路拓扑生成算法[J]. 交通与计算机, 2004, 22(3): 37-40.
- [11] 张小国, 王庆, 万德钧. 基于物理统一存储大规模数字导

- 航地图道路网拓扑自动生成算法[J].中国惯性技术学报,2009,17(6):669-676.
- [12] 吴长彬,闫国年.线面拓扑和度量关系的细分描述和计算方法[J].计算机辅助设计与图形学学报,2009,21(11):1551-1557.
- [13] 邓敏,刘文宝,黄杏元,等.空间目标的拓扑关系及其GIS应用分析[J].中国图象图形学报,2006,11(12):1743-1749.
- [14] 陆守一.地理信息系统[M].北京:高等教育出版社,2004.
- [15] 徐立,孙群,杨帅,等.地图矢量数据拓扑关系生成算法[J].地理与地理信息科学,2012,28(4):18-21.
- [16] 史文中,郭薇,彭奕彰.一种面向地理信息系统的空间索引方法[J].测绘学报,2001,30(2):156-161.
- [17] 胡久乡,何松,钟瑜.空间数据库网格索引机制的最优划分[J].计算机学报,2002,25(11):1221-1230.
- [18] 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(13):58-61.
- [19] Theodoridis Y, Stefanakis E, Sellis T. Efficient cost models for spatial queries using R-trees[J]. IEEE Transactions on Knowledge and Data Engineering, 2000,12(1):19-32.
- [20] 谷红亮,史元春,徐光祐.智能空间位置感知的伺候式服务模式[J].清华大学学报(自然科学版),2006,46(4):584-587.
- [21] 张宏,温永宁,刘爱利,等.地理信息系统算法基础[M].北京:科学出版社,2006.
- [22] Guttman A. R-Trees: Adynamic index structure for spatial search[J]. ACM SIGMOD Record,1984,14(2):47-57.
- [17] 胡久乡,何松,钟瑜.空间数据库网格索引机制的最优划

An Efficient Algorithm to Construct Connectivity Topology for Road Network Using R-tree and B-tree Indexes

HAN Zhiheng, RUI Xiaoping*, SONG Xianfeng, LIU Zhenyu and WANG Jing

(College of Resources and Environment, University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: A road network with topology is the basis for optimal path finding. According to the definition of connectivity topology by ISO GDF4.0 (Geographic Data File) model and the requirements of road network connectivity topology during the optimal path finding procedure, this paper presents an efficient algorithm to construct connectivity topology for road network using R-tree and B-tree indexes. The efficient algorithm contains two parts: the first part breaks the roads at intersections and generates anew road network; the second part constructs connectivity topology based on the new network. The procedure of building new road network breaks the roads into line segments and gets intersections at first, then reconstructs new roads with the line segments set and the intersections. During the procedure of getting intersections of any two line segments, the algorithm builds R-tree spatial index on the line segments to improve query operation efficiency significantly. What's more, our algorithm also builds B-tree index on serialized data of the roads such as the identity codes while constructing connectivity topology to improve the query operation efficiency on serialized data. This paper also tests the time complexity of the algorithm using road networks with different scales. Experiment results show that our algorithm builds connectivity topology of large-scale road network in very short time. (For a road network that contains about 30 thousands line segments and 25 thousands points, the quick algorithm completes connectivity topology construction in no more than 3 seconds). The time cost of our algorithm increases slightly while the traditional algorithm increases tremendously, to be exact, our algorithm's time cost grows at the rate of $O(v \log(v))$ while the traditional one grows with the speed of $O(v^2)$, so we concluded that the quick algorithm has high efficiency and is valuable to practicable application.

Key words: road network; connectivity topology; R-tree; B-tree; efficient algorithm

*Corresponding author: RUI Xiaoping, E-mail:ruixp@yahoo.com.cn