基于预处理的城市路网拓扑结构构建算法

孙棣华[1],肖 锋[1],廖孝勇[1],赵 敏[1],吴宏伟[2],唐 亮[2]
SUN Di-hua[1],XIAO Feng[1],LIAO Xiao-yong[1],ZHAO Min[1],WU Hong-wei[2],TANG Liang[2]

1.重庆大学 自动化学院,重庆 400044
2.重庆市交通委员会,重庆 401147
1.College of Automation,Chongqing University,Chongqing 400044,China
2.Chongqing Municipal Commission of Communications,Chongqing 401147,China
E-mail:xiaofeng6712@163.com

SUN Di-hua,XIAO Feng,LIAO Xiao-yong,et al.Algorithm based on pre-processing for constructing topological structure of urban road network.Computer Engineering and Applications,2008,44(23):233-235.

Abstract: This paper focuses on the construction of road network topology,which is the foundation of optimal path planning. Considering the deficit of topology in MapInfo electronic maps,inefficiency and low precision in existing topology construction algorithm,a pre-processing method of buffer analysis technology and the calculation of the regional particle is proposed to classify the primitive irregular road network structure and supplement the road information before the construction of road network topology.On the basis,the road section map and node map layer are created,moreover,the road network topology can be built. Through application of the presented algorithm,the construction of road network topology is achieved in the VB6.0 development environment with MapX control.The algorithm is proved capable of improving the precision and efficiency obviously.
Key words: road network;topological structure;MapX;MapInfo

摘 要:路网拓扑结构构建是最优路径规划的基础。针对 MapInfo 数据格式电子地图不具备拓扑结构,且现有拓扑结构构建算法精度低、效率差等不足,提出在路网拓扑结构构建前,应用缓冲区分析技术和计算区域质点等预处理方法,对原始路网不规则的关系进行分类和道路信息补充,以此为基础创建路段和节点图层,建立路网拓扑关系。应用该算法,在 VB6.0 开发环境和 MapInfo 二次开发控件 MapX 支持下,实现了重庆市路网拓扑结构的构建。实验结果表明,该算法构建精度和效率明显提高。
关键词:路网;拓扑结构;MapX 控件;MapInfo 数据格式
DOI:10.3778/j.issn.1002-8331.2008.71 **文章编号**:1002-8331(2008)23-0233-03 **文献标识码**:A **中图分类号**:TP301.6;P285

1 引言

最优路径规划是智能交通系统中应用广泛的关键技术,其核心是最短路径算法, 实质是计算路网中两顶点间的最短路径。采用适当的方法构建并正确描述路网拓扑结构,是具体实现最短路径算法的必要前提。因此,构建路网拓扑结构是最短路径算法实现的基础,没有路网拓扑结构,最优路径规划无从谈起[2]。

MapInfo 是目前各种地理信息系统(GIS)中较流行的桌面型应用平台,具有空间信息与属性信息能很好结合,图形能力强,专题地图制作多样化,空间查询方便等优点,因此,许多城市电子地图均采用 MapInfo 数据格式。同时,MapInfo 公司向用户提供了具有强大地图分析功能的 ActiveX 控件 MapX, 该控件是基于 Windows 操作系统的标准控件,具有很好的易开发性和开放性。但不足是 MapInfo 本身不具备构建对象拓扑结构的能力[2]。

在实际 MapInfo 格式的电子地图中,各条道路线是一个整体,在交叉路口未断开生成各个路段,也未建立路网中顶点和弧段的关系,因此,无法描述路网的拓扑结构关系。此外,由于电子地图制作不规范,一些道路衔接部分没有反映道路间实际拓扑关系。例如,两条相交道路在电子地图上未相交(图 1(a));在交点处线路过长(图 1(b));多条道路端点在交叉口未交于一点(图 1(c))。

(a)相离 (b)交叉 (c)交叉口处相离

图 1 电子地图中 3 种道路相邻的情况

基金项目:重庆市科技公关计划(the Key Technologies R&D Program of Chongqing City,China under Grant No.CSTC,2005AC6037)。
作者简介:孙棣华(1962-),男,教授,博士生导师,主要研究方向为控制理论与控制工程、系统工程、智能交通系统;肖锋(1982-),男,硕士研究生,主要研究方向为 GIS 在智能交通中的应用。
收稿日期:2007-10-12 **修回日期**:2008-01-09

目前,路网拓扑结构构建算法对上述问题主要有两种解决方式：第一种是在 MapInfo 平台上应用节点抓取功能手动处理[2-3]。其操作简单,但会面临两个难题:(1)一个城市道路成百上千条,由人工对每条道路进行节点抓取处理,工作量太大,且质量没有保证;(2)应用节点抓取处理需要设置最后节点公差参数,但参数设置过大会破坏道路的几何特征,参数设置过小,道路交叉口处的多条道路不能相交在一起。

第二种是在程序中设置“捕获距离”,当节点间、或者节点与线间距离小于此值后,即自动连接[3]。该方法较适合于简单情形,如果遇到复杂情形(如图 2(b))或者更多条道路在交叉口相离、交叉,在进行道路连接时原始道路的几何特征变动很大,而且处理精度也会下降;此外,在处理过程中需要在道路图层中对道路进行频繁的删除和创建。若一条道路中有多个交叉口出现图 1 所示的 3 种情形，则该道路需进行多次删除和创建,将导致图层中原始道路的几何特征发生变化,且处理过程明显耗时和效率低下,并影响道路拓扑结构构建的精确度。

为此,针对城市电子地图实际情况和 MapInfo 以及现有路网拓扑结构构建算法的不足,提出一种通过路网信息预处理技术提高城市路网拓扑结构构建效率和精度的算法。

2　新算法的提出

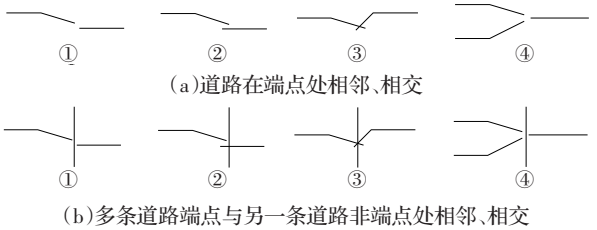
本文提出算法的基本思想是:通过在路网拓扑结构构建前建立道路缓冲区、分别对道路首尾节点进行分析、求多点所组成区域的质点及质点在相邻道路上的垂点等预处理,对原始道路图层中不规则的路网关系进行分类和信息补充,以提高拓扑结构构建的精度。现有算法需要对道路进行频繁删除和创建才能完成对原始道路的预处理[2],本算法则首先进行信息的补充,然后只进行一次创建路段即可完成对原始道路的预处理和路段图层的构建,这将提高算法的运行效率。

算法分为 3 步:原始路网预处理、路段图层创建、路网拓扑结构构建。

(1)原始路网预处理是应用缓冲区分析技术和计算区域质点等预处理方法对原始路网不规则的关系进行分类,主要分为三种典型情形,见图 2。针对不同的情形应用不同的方法收集和补充道路信息,并将信息写入各道路的属性表内。该预处理方法只在道路属性表内补充必要的道路信息,不对原始道路的几何特征进行修改。

(2)路段图层创建是根据写入到属性表内的信息和使用 MapX 对象属性提取的道路节点、形状点信息新建路段图层。

(3)路网拓扑结构构建是在路段图层的基础上创建节点图层,然后从路段图层和节点图层中提取拓扑信息写入路段表和节点表内。



(a)道路在端点处相邻、相交

(b)多条道路端点与另一条道路非端点处相邻、相交

(c)两条道路在非端点处相交

图 2　电子地图上原始道路拓扑关系典型情形

3　算法设计

3.1　定义

为便于描述算法,给出如下定义:

(1)*Li*:在道路图层遍历的第 *i* 条道路。

(2)道路集合 *T*:道路 *Li* 缓冲区搜索到的道路集合。

(3)道路集合 *S*:以道路 *Li* 首节点为圆心搜索到的道路集合。

(4)道路集合 *E*:以道路 *Li* 尾节点为圆心搜索到的道路集合。

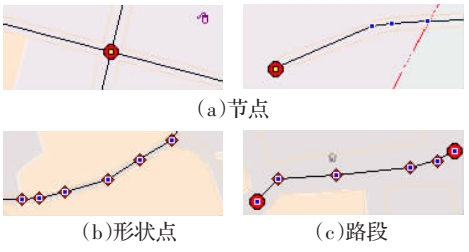
(5)*SD1j*、*SD2j*:*SD1j* 为 *S* 中 *Sj* 的首节点到 *Li* 首节点的距离;*SD2j* 为 *S* 中 *Sj* 的尾节点到 *Li* 首节点的距离。

(6)*SR*、*MR*:*SR* 为图 2(b)中的特殊道路;*MR* 为图 2(c)中的道路。

(7)*P*1、*P*2:*P*1 为道路节点和形状点集合;*P*2 为原始路网预处理所补充的节点集合。

(8)*PC*:道路的路段点集数组,*PC*(*i*)表示第 *i* 个点集(*i*=1,…,*n*)。*n* 表示该道路可分为 *n* 个路段。

(9)节点:道路与道路的交叉点或道路的端点(图 3(a));形状点:指同一条非直线道路上的转弯点(图 3(b));路段:两节点间的一段,用于表示分段道路(图 3(c))。



(a)节点

(b)形状点　(c)路段

图 3　路网中的几何元素

3.2　算法设计

3.2.1　原始路网预处理算法设计

(1)在原始道路图层的属性表中添加所需属性字段,用于存放预处理过程中收集和补充的道路信息。

(2)判断道路图层中道路 *Li* 与相邻道路的关系。

①选取适当的阈值为缓冲区宽度创建 *Li* 的缓冲区域,搜索与该缓冲区域相交的道路,写入 *T* 中。

②从 *T* 中找出与 *Li* 的关系属于图 2(c)中的道路 *Tj*。

③分别以 *Li* 的首、尾节点为圆心,选取适当的数值为半径画圆,搜索与该圆相交的道路,并分别写入 *S*、*E* 中。

④若 *S* 非空,分析 *Li* 与 *S* 中 *Sj* 之间的关系:首先分别计算 *S* 中 *Sj* 的首、尾节点到 *Li* 首节点的距离 *SD1j*、*SD2j*,然后通过判断 *SD1j*、*SD2j* 与阈值的关系即可得知属于第几种情形,把图 2(b)中的特殊道路记为 *SR*(*E* 中道路与 *Li* 关系的判断方法同理)。

(3)针对 *Li* 与相邻道路 3 种情形的不同处理方法:

图 2(a)情形:①计算 *S* 中所有道路与 *Li* 首节点相邻节点组成区域的质点 *P*。②将 *P* 的经、纬度写入 *Sj* 的属性表内。

图 2(b)情形:①计算 *S* 中所有非特殊道路与 *Li* 首节点相邻节点组成区域的质点 *P*。②计算 *P* 在特殊道路 *SR* 上的垂点 *V* 及垂点 *V* 在 *SR* 中的位置次序。③将 *V* 的经、纬度添加到*SR*、*Sj* 的属性表内,并把 *V* 的位置次序添加到 *SR* 的属性表内。

图 2(c)情形：①计算 *MR* 与 *Li* 的交点，然后分别求出交点 *MR* 在 *Li* 和中的位置次序。②分别将交点与位置次序写入 *Li* 和 *MR* 的属性表内。

3.2.2　路段图层创建算法设计

(1)遍历原始道路图层，提取 *Li* 的节点和形状点，依次存放于点集合 *P*1 中；并从的属性表内提取出所补充的交叉点位置信息，依次存放于点集合 *P*2 中。

(2)将 *P*2 中的节点依次顺序插入到 *P*1 中。

(3)从 *Li* 的属性表内取出首、尾节点位置信息 *PS*、*PE*，然后将 *PS*、*PE* 依次顺序插入到 *P*1 中。

(4)将 *P*1 的所有点按节点断开后写入到路段点集数组 *PC* 中，使 *PC*(*i*)中只是首、尾点为节点，其余点为形状点。

(5)根据 *PC* 新建路段图层。

3.2.3　路网拓扑结构构建算法设计

路网拓扑结构利用节点表达路段与路段之间的连通性，因此构建路网拓扑结构主要内容是：提取和处理节点、路段信息，从而建立拓扑结构。具体方法如下。

(1)设计节点表和路段表结构，见表 1，表 2。

表 1　节点表

NodeID	节点编号
NodeLong	节点经度
NodeLat	节点纬度
NodeArcsNum	相连通路段个数
NodeArcsID	相连通路段编号

表 2　路段表

ArcID	路段编号
RoadName	道路名称
RoadAreItem	道路路段号
StartNodeID	首节点编号
EndNodeID	尾节点编号
StartArcsNum	首节点相连通路段个数
StartArcsID	首节点相连通路段编号
EndArcsNum	尾节点相连通路段个数
EndArcsID	尾节点相连通路段编号
ArcLength	路段长度

(2)遍历路段图层，提取每条路段的首、尾节点，判断首、尾节点在节点图层是否已存在，如果不存在，则在节点图层中建立点对象，然后在路段图层中搜索以该点为节点的路段，并把搜索得到的路段数和路段编号写入节点表和路段表内，最后把节点表和路段表没有赋值的字段分别赋值；如果该节点已存在，则提取下一个节点。

3.3　路网拓扑结构的数据结构表示

路网拓扑结构的表示方法有多种，比较常用的方法有邻接矩阵法：用一个二维数组存放顶点间关系(边或弧)的数据；前向星型结构(Forward Star Representation)[5-6]法：使用两个数组表示网络的拓扑关系，一个数组存储和弧段相关的数据，称为弧段数组，另一个数组存储和节点相关的数据，称为节点数组。

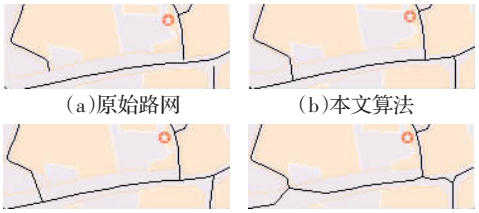
为了提高搜索效率，可以根据最短路径算法，采用不同的数据结构表示网络的拓扑关系。对于 Dijkstra 算法，为了缩短算法运行时间，采用先进的数据结构和搜索方法可以改进节点选择和标号更新这一“瓶颈”，从而将时间复杂度由二阶降到对数阶。例如：引入一个新的数据结构-基数堆(radix heap)。采用一阶形式基数堆的 Dijkstra 算法，其时间复杂度为 o(m+$n$log$C$)；采用二阶形式的基数堆，其时间复杂度为 o(m+$n$log$C$/loglog$C$)；其中，C 为非负整数弧权的上界。

本文的重点是构建路网的拓扑结构，具体采用何种数据结构存储路网拓扑结构，需要依据最短路径算法和所要求的搜索时间。

4　实验结果

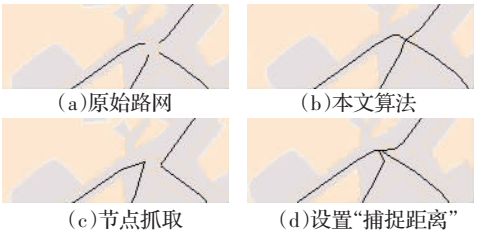
为验证算法，依据重庆市公交电子地图道路图层中主要的 489 条道路，在 VB6.0 环境下进行了实验。

图 4 和图 5 分别是原始路网及由不同方法预处理后获得的路网拓扑结构图，包括本文算法、节点抓取方法和设置“捕捉距离”方法。



(a)原始路网　(b)本文算法　(c)节点抓取　(d)设置“捕捉距离”

图 4　道路端点与相邻道路相离时不同方法预处理对比图



(a)原始路网　(b)本文算法　(c)节点抓取　(d)设置“捕捉距离”

图 5　多条道路的端点在交叉口处相离时不同方法预处理对比图

图 4(c)、图 5(c)是使用 MapInfo 节点抓取功能处理后的结果。从图 4(c)看出，虽然道路首、尾节点与相邻道路进行了连接，但首、尾节点附近的弧段也发生了变化。图 5(c)中，由于最后节点公差参数设置过小(40 m)，道路交叉口处的多条道路没有相交在一起；如果参数设置过大，将会破坏首尾节点附近弧段的几何特性。

图 4(d)、图 5(d)是采用设置“捕获距离”方法处理后的结果。图 4(d)中，虽然道路首、尾节点及附近的弧段都没有变动，但与该道路相邻道路的内部几何特性却改变了。

图 4(b)、图 5(b)是采用本文算法预处理后的结果。图 4(b)中，除了道路的首、尾节点与该节点在相邻道路的垂点进行了连接外，该道路与相邻道路的几何特性都没有改变，也不需对相邻道路进行频繁的删除和创建。图 5(b)中，道路首、尾节点附近弧段的几何特性没有发生变化。

原始城市路网中每条道路与相邻道路在衔接处都没有反映道路间的拓扑关系，若采用节点抓取或设置“捕捉距离”方法进行预处理，预处理后每条道路的几何特性都发生了变化，在此基础上构建的路网拓扑结构的精度将难以满足最优路径规划的需要。而本文算法减少了不相关的操作，只是对原始道路增加或更改节点，没有改变原始道路的几何特性，因此算法不仅节约了人力，而且明显提高了精确度和运行效率，适合实际应用的需要。

经算法处理后，构建的重庆市主城区主要 489 条道路的拓扑结构如图 6 所示。建成后的路网拓扑结构从原 489 条道路中抽取 883 个道路节点，分割得到了 1 121 条路段。

构建的路网拓扑结构表达方式如图 6 中节点表和路段表所示。例如，节点表中 NodeID=129，即编号为 129 的节点，其 NodeArcsNum=3，即连通的路段有 3 条，编号分别为 138、139、777。通过路段编号可以在路段表中依据 ArcID 找到相应的路

(下转 248 页)