

基于矢量地图的路径寻优算法

鲍远慧 冯三强 徐敏

摘要 本文介绍一种基于矢量地图的最优路径搜索算法。首先描述矢量地图库的存储结构, 然后提出针对这种特定存储结构的最短路径搜索算法, 并对算法的收敛性加以讨论。算法采用启发式代价树的广度优先搜索法, 其设计思想对于其它类型存储结构的矢量图的路径寻优问题仍有意义。本算法在合肥, 厦门等城市的矢量化电子地图的基础上成功实现。

关键词 矢量, 地图, 最短路, 最优化算法

1. 背景介绍

电子地图作为一种新型空间信息管理技术, 在城市规划、交通管理等方面得到广泛的应用。而在地图中寻找任意两点间的最短路径这一古老的问题具有很大的现实意义, 比如降低能源消耗, 缩短路途时间。优化的路径带来的将是巨大的经济效益。

计算机不可能从 Bitmap 中寻找路径, 因为位图仅保存了点的信息, 点与点之间拓扑关系没有体现。拓扑关系的抽取和存储是实现智能化电子地图的关键, 也是各种矢量形式的差异所在。仅停留在孤立点信息的地图, 不足以实现路径寻优等实时决策。文中将介绍一种面向对象结构的矢量存储结构。针对这种结构, 文中提出的寻优算法得到了很好的实现。

2. 矢量图的定义及其基本操作

地图矢量化的目标是得到一个带权的无向图, 即网(Network), 它的形式化定义为 $NetWork = (V, R)$, 其中 $V = \{x | x \text{ dataobject}\}$, $R = \{VR\}$, $VR = \{(x, y) | P(x, y) (x, y \in V)\}$ 。网中的数据元素是由经矢量化处理之后得到的点; VR 是两个顶点之间的拓扑关系集合。无序对 (x, y) 表示 x 和 y 之间的一条边, 谓词 $P(x, y)$ 则表示 x 到 y 的一条通路。网中的顶点 v 按度分为两类 $V1, V2$, $V1 = \{x | TD(x) \leq 2, x \in V\}$, $V2 = \{x | TD(x) > 2, x \in V\}$, 划分的意义是区分出道路网的交叉口。若 Network 是连通图, 对任意 $v_i, v_j \in V2, i, j$, 则至少存在一条从顶点 v_i 到 v_j 的路径, 途中必经 $n - 0$ 个顶点 $v_k \in V2, 0 \leq k < n$, 先后历经的 v_{i-1}, v_i 之间, $1 \leq i \leq n+1$, 必由 $m - 0$ 个顶点 $v_m \in V1$ 连接。为方便起见, 对于任意连接 $v_i, v_j \in V2, i, j$ 的路径(path), 若途中历经的任一顶点均属于 $V1$, 称其为桥(bridge)。显然, 对于矢量图中任意两顶点间的路径, 必由起点、终点附近若干条边和中间若干架桥组成。不难看出, 对于路径寻优类的问题, 大部分的操作可以在 bridge 层进行, 仅对起点、终点的操作需在边(edge)层上。对 bridge 操作可大大节省时间和空间开销。因此, 基于 Network, 生成由顶点集合 $V2$ 和联接桥构成的 Bridge-Network 尤其重要, 它的形式定义为 $Bridge-Network = (V, R)$, 其中 $V = \{x | x \in V2\}$, $R = \{VR\}$, $VR = \{(x, y) | P(x, y) (x, y \in V)\}$ 。

为使矢量地图具有可操作性, 上述 Network 和 Bridge-Network 必须提供以下基本操作:

- (1) GET_BRIDGE(v) 求所属桥函数。确定所有包含顶点 v 的桥。
- (2) NEXT_VERTEX(v_n, v_b) 求邻接顶点函数。根据现在所处和上次历经的顶点 v_n, v_b , 决定下一步将历经的顶点。
- (3) NEXT_BRIDGE(B_n, B_b) 求邻接桥函数。根据现在所处和上次历经的桥 B_n, B_b , 决定下次所有可能历经的桥。
- (4) GET_ADJ_BRIDGE(x) 求最近邻桥函数。已知图上任意一点, 求最近邻的桥。
- (5) GET_ADJ_VERTEX(x) 求最近邻顶点函数。已知图上任意一点, 求最近邻的顶点。
- (6) INS_VERTEX(v) 插入顶点操作。添加一个顶点 v 。
- (7) INS_EDGE(v, w) 插入边操作。添加一条顶点 v, w 之间的边。
- (8) DEL_VERTEX(v) 删除顶点操作。删除顶点 v 。
- (9) DEL_EDGE(v, w) 删除边操作。删除一条顶点 v, w 之间的边。
- (10) DETACH_BRIDGE(B, v) 顶点分离操作。分离顶点 v 与桥 B 之间的从属关系。

在以上操作中, (1) 至 (5) 在路径寻优算法中用到; (5) 至 (10) 在矢量图的生成和编辑

时用到。

3. 矢量图的存储结构

上述对矢量图的抽象定义最终必须对应具体的数据结构，这里采用分块链式映象结构(见图.1)。矢量库所要的基本信息是一簇离散点的位置信息，而点点间的拓扑关系(边和桥的信息)，则是在矢量化过程将各点归属到某一个或某几个桥上得到的。实际处理过程总是从一个交叉点搜索到另一个相邻的交叉点，然后将当前这对交叉点间的搜索路径用一组顺序的关键点取代，这簇点就构成了一个桥，因为生成先后顺序不同，各点和桥便有了唯一的索引，若将所属桥的索引信息添加到点信息中，便建立起了 Bridge-Network。又因桥的关键点的索引以顺序结构保存，所以 Network 已经得到。图源不同，信息量就不同，库的大小自然也就不同。综合时间和空间的考虑，采用分块链表保存信息，其存储结构的形式描述如下：

```
CONST blocknum=..., {每块存储的信息单元数}
      membernum=..., {信息单元所含成员链的块划分 }
TYPE  blockptr=1..blocknum;    memberptr=1..membernum;
      member = ARRAY [ membernum] OF integer ;
      memberptr= membernode;
      membernode = RECORD
          nextmember:memberptr;
          memberdata:member;
      END;
      nodedata = RECORD
          noddata : ... ; {结点信息}
          memptr :memberptr ;
          ... : ; {与结点有关的其它信息和操作}
      END;
      block = ARRAY [ blocknum] OF nodedata ;
      blockptr= blocknode;
      blocknode = RECORD
          nextblock:blockptr ;
          blockdata:block;
      END;
```

Network 和 Bridge-Network 均采用了上述存储结构，参数的意义有以下两点不同：

1> Network 的 Nodedata.noddata 保存的是顶点的位置，而 Bridge-Network 中保存的是桥的索引；此外 Nodedata 中保存的操作不同。

2> Network 的 membernode.noddata 保存的是所属桥的索引，而 Bridge-Network 中保存的是构成桥的顶点索引。

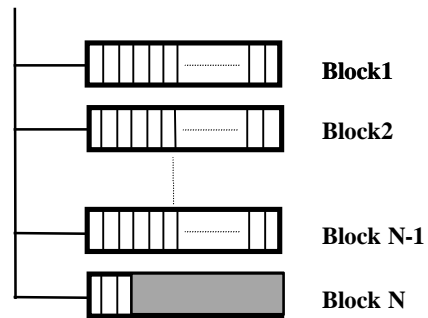


图 1.分块链式结构

通过上述定义，Network 与 Bridge_Network 之间已具备双向可操作性，既可由已知顶点找到所属的桥，也可由已知桥得到包含的所有顶点。不难想象，两者结合可以实现上述(1)至(5)的基本操作。

4. 最短路径搜索算法

算法的基本思想是：假设求从顶点 v_i 到 v_j 的最短路径。首先建立两个链表 OpenList、

CloseList,分别存放待展开和已展开顶点。链表结点的形式描述如下：

```
BridgePtr = BridgeNode
BridgeNode= RECORD
    nextbridge : BridgePtr ;
    BridgeID ,LastBridgeID : integer ;
    Length : integer ;
    Preseat : integer ;
END
```

其中, BridgeID 为上次历经的桥索引, LastBridgeID 为当前所在的桥索引；

Length 为从起始桥到达当前桥的当前路径长度；

Preseat 为从起始桥到达当前桥的当前路径的上一段桥保存在 CloseList 中位置。

将起始桥以上述 BridgeNode 的结构压入 OpenList 中 Length = 0 , Preseat = -1 , LastBridgeID = 0 , BridgeID = 起始桥索引。展开 OpenList 的第一个结点, 即将次结点从 OpenList 中取出, 压入 CloseList, 返回压入位置, 然后找出展开结点两端的邻接桥, 以 BridgeNode 结构按 Length 的大小顺序地压入 OpenClose 表, 使 OpenClose 表中结点 Length 保持从小到大的顺序, 从而每次展开结点的 Length 总是当前最小的。当展开结点已在 OpenList 中存在时, 比较两者的 Length, 若待压入的结点 Length 小, 则取代已存在结点, 否则舍去。重复上述步骤, 直至 OpenList 的第一个结点 BridgeID=终止桥索引。

以上是算法的基本思路, 我们自然会对算法的收敛性和正确性提出疑问。下面陈述的事实可打消这种顾虑：

A> OpenList 中结点的 Length 总保持增长的趋势, 即便某一结点被较小 Length 的展开结点所取代, 但仍将呈增长趋势, 因为目标点尚未发现, 结点还要继续展开下去。而出发点到目标点的路径长度是有穷, 随着 Length 的不断增长, 最终定有一条路径收敛于终止桥。

B> OpenList 总是先展开 Length 最小的结点, 而且即便展开的子结点已存在于 OpenList 中, 仍需进行一次插入操作, 只有插入位置位于表头, 才能最终认定。从而保证了路径的最优。

注意到最短路径的走向总是朝着终止桥的方向, 依此对上述算法加以优化：

如果将寻优的估价函数由 $f(x)=g(x)$ 改为 $f(x)=g(x)+h(x)$, 必可大大提高收敛的速度, 其中 $g(x)$ 表示从起始桥到当前桥的实际走过的长度, $h(x)$ 是从当前桥到终止桥的直线距离。因此在以前的 Length 中添加预估长度, 作为 OpenList 排序的依据, 将很大程度上提高算法的时间和空间效率。这种优势在稠密网中体现的非常明显。

图 2. 给出优化后算法的工作流程图：

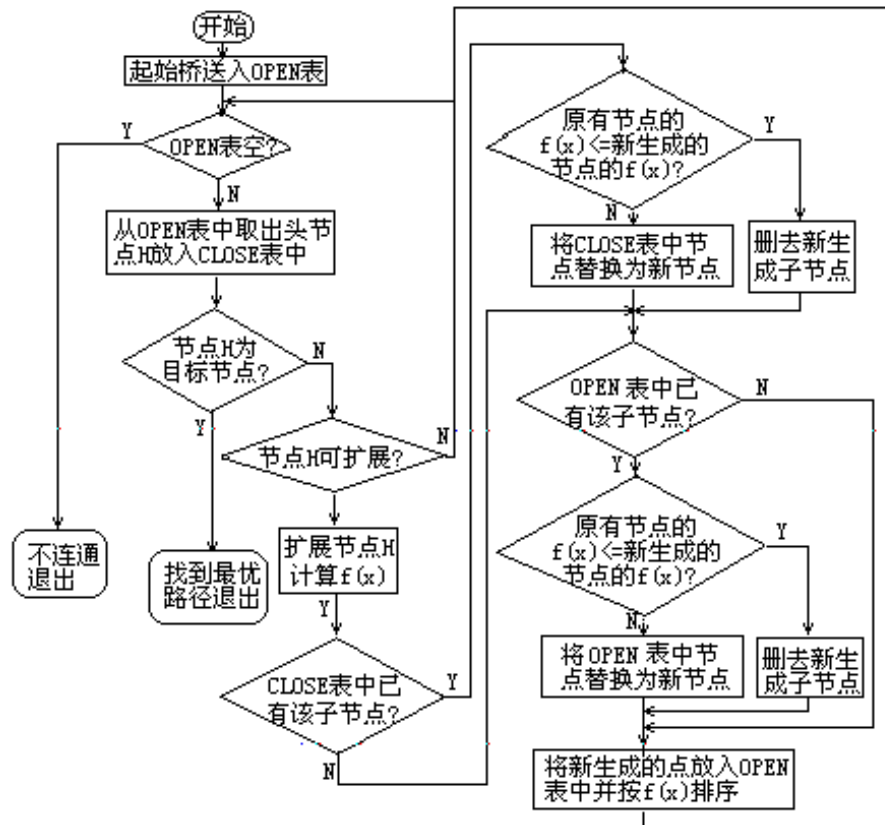


图 2. 寻优算法流程图

图 3 给出由顶点 A、B、C，及交叉点 1~6 构成的拓扑结构，桥的权值已给出，考虑从 A 点到 B、C 点的最优路径选取过程中 OpenList 的顶点序列，见下表 1。

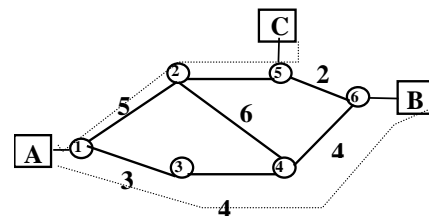


图 3 简单拓扑结构

表 1

结点展开次数	当前所在顶点	OpenList 中的顶点序列	
		A B	A C
1			
2			
3		()	()
4			(找出路径)
5		() (找出路径)	

注: (...) ... 表示当展开的新结点已存在于表中时, 按 Length 的大小决定取代与否及排序插入。

从表 1 中看出, 前 4 次展开的过程对 A B, A C 完全一样, 因为上述过程没有将预估值考虑进去。

5. 举例

依据上述设计思想, 我们成功地生成了合肥、厦门等城市的矢量地图库, 并实现了上述寻优算法。图 4 以合肥市矢量地图为例说明其拓扑结构; 图 5 为路径选优的一个实例, 找出顶点 1 与 10 之间的最短路径, 路径以反相显示出来, 图中各交叉点已标出序号。

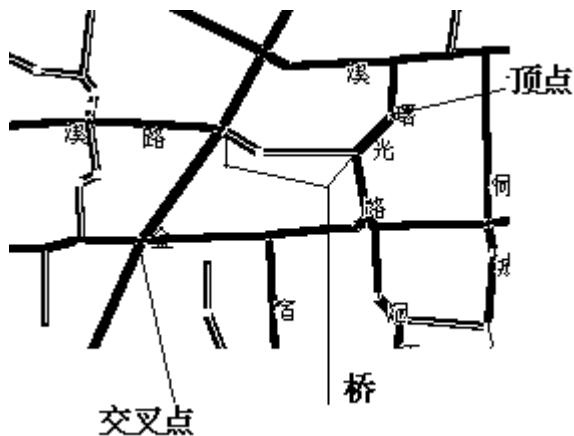


图 4. 合肥市矢量地图



图 5. 路径选优实例

参考文献

- [1] 严蔚敏, 吴伟民. 数据结构. 清华大学出版社, 1995.155-197
- [2] Haotao Wang, Zhenwang Yao and LingyuDuan. " Automatic Generation of Vector-Map ", Proceedings of 1997 China-Japan Joint Symposium on Advanced Energy and Transportation Engineering, 392-397.
- [3] 林春蔚等. C 环境下地图图像矢量化及图形编辑技术和实例. 海洋出版社, 1995.
- [4] 卢开澄. 算法与复杂性. 高等教育出版社, 1995

作者简介 鲍远慧 女 1955 年生 ,1981 年毕业于合肥工业大学电气系。现为合肥工业大学电气系自动化教研室讲师。研究方向为微机控制, 计算应用及软件优化。

冯三强 男 1973 年生 ,1994 年毕业于成都科技大学。现为中国科学技术大学自动化系研究生。主要研究方向是 CT 误差修正与图象重建, 基于 DICOM 标准的 PACS。

徐敏 女 1977 年生 , 现为中国科学技术大学自动化系本科生。主要研究方向是 GPS 在车辆监控系统中的应用, 地理信息系统的开发与应用。



An Algorithm for Optimum Path Based on the Vector Map

Bao Yuanhui , Feng Sanqiang, Xu Min

ABSTRACT

In this paper, an algorithm for seeking optimum path, based on the vector map, was introduced. We begin with discussion of a kind of data structure for the vector map, which will be applied in the realization of the algorithm. This algorithm introduced scope priority searching method by elicitation cost tree. Its design method will still make sense for other types of vector map. This kind of algorithm was successfully achieved on the vector maps of HeFei City and XiaMen City.