

# 矢量地图下最短路径问题的研究

## Research on Shortest Path Problem of Vector Map

夏冰, 鲍远律(中国科学技术大学自动化系, 安徽合肥 230027)

XIA Bing, BAO Yuan-lu (Dept. of Automation, Univ. of Sci. And Tech. of China, Hefei 230027, China)

**摘要** 本文讨论了矢量地图下的最短路径问题, 就矢量地图下最短路径寻优算法的实现进行了深入的研究, 并应用于具体的城市道路环境中进行检验, 取得了较好的结果。

**关键词** 最短路径; 最短路径寻优算法; A\* 算法; 矢量地图

**ABSTRACT** In this paper, we discuss the shortest path problem of vector map, and study the algorithms of shortest path-Finding, and achieve good result when using it in city road maps.

**KEYWORDS** :Shortest Path ;Shortest Path-Finding Algorithms ;A\* Algorithms ;Vector Map

中图分类号: TP301.6 文献标识码: A

### 1 引言

GPS(全球卫星定位系统)和GIS(地理信息系统)技术在车辆导航和监控领域有着广泛的应用。在车辆导航和监控系统软件中, 一个基本的辅助决策功能即寻找两点间的最短路径。在复杂的城市道路网络中, 如何快速准确地找到出发点和目标点之间的最短路径, 取决于不同的路径寻优算法和矢量地图的数据结构。

本文主要研究基于矢量地图的最短路径问题和路径寻优算法的实现。

### 2 最短路径问题的定义

最短路径问题(Pathfinding or Shortest Path Algorithms)是一种计算机图形搜索算法, 即在出发点和目标点之间找出总代价最低的路径。路径寻优算法一方面要完成探索最低代价的路径, 另一方面要做到尽可能快, 尽可能少占用内存, 即尽可能降低算法的时间复杂度和空间复杂度。

### 3 路径寻优算法

最短路径的寻优有很多不同的实现方法, 下面介绍几种主要的路径寻优算法。

#### 3.1 Dijkstra(迪杰斯特拉)算法

下面给出 Dijkstra 算法的简洁描述:

输入为有向图  $G=(V, E)$ , 源顶点  $S \in V$ , 这里  $V$  是顶点的集合,  $E$  是图  $G$  的边集合。边的权值函数  $W(u, v) \geq 0, (u, v) \in E$ 。算法将顶点集合  $V$  划分为两类,  $V = R \cup Q$ 。集合  $R$  包含的顶点是那些已经确定的到离源顶点的最近路径的顶点, 集合  $Q = V - R$ 。  $Q$  又称为打开(Open)集合,  $R$  又称为关闭(Close)集合。当一个顶点从集合  $Q$  转移到集合  $R$ , 我们就称它已经“退休”。对于每一个顶点  $v$ , 我们都保存当前最短路径的估计值  $g[v]$  并为每一个顶点保存一个“入边”  $ed[v]$ , 即在到当前顶点的最短路径中, 那个直接由邻点到当前顶点的边。这样, 我们就可以根据这个信息, 从目标点反向回溯重建这条最短的路径。如果有一条以上的最优路径, 只有最近发现的路径被记录下来。

算法就是不停地展开具有当前最小代价估计  $g[v]$  的顶点  $v \in Q$ 。因为所有边的代价都为非负, 在某一顶点展开后得到的最短路径, 不可能比当前顶点得到的最短路径还短。因此, 我们已经确定了某一顶点  $v$  的当前最短路径后, 就可以将其“退休”至集合  $R$ 。然后我们检验, 当前这条路径向相邻顶点展开出去, 是不是会有比那些相邻顶点当前具有最短路径更短的路径。如果有, 我们就更新最短路径的估计以及相应的相邻顶点的入边。

#### 3.2 BFS(最好优先)算法

最好优先算法是一种启发式的搜索算法, 也就是它将地图的一些知识带入了搜索标准中。它类似 Dijkstra 算法, 但它总是先走向离目标点最近的顶点, 而不是逐渐走向离出发点越来越远的顶点, 然后一步步地逼近。BFS的关键就是“面向问题”寻找可以用来做决策的标准。标准选择的好坏直接关系到搜寻的结果, 甚至关系到是否有解。A\* 算法就是建立在典型的 Dijkstra 算法和 BFS 算法之上的。

#### 3.3 A\* 算法

A\* 算法是一种非常有效的路径寻优算法, 它比 Dijkstra 算法和 BFS 算法都要快。算法的整体框架是一个图的遍历搜索算法, 但它不同于大多数图的搜索

算法,采用了启发函数来估计地图上的任意点离目标点的远近程度。通过这种启发式,可以协助选择最好的搜索方向。 $A^*$ 算法希望通过将搜索方向偏向目标点,从而提高搜索的效率。最基本的思想是,取代 Dijkstra 算法采用的  $g[v]$  排序优先队列,用  $[v]=g[v]+h[v]$  作为新顶点插入队列的排序标准,这里  $g[v]$  表示源点到当前顶点的实际代价, $h[v]$  是当前顶点的启发值。理论上已经给出证明,如果  $h[v]$  与从  $v$  到  $t$  的实际最小代价相比,是一个低估值(Underestimate),那么  $A^*$  就可以产生像 Dijkstra 算法的最优解。优先队列最糟糕的情况就是和 Dijkstra 一样的效率,但是,我们如果提供一个良好的启发函数  $h[v]$ , $A^*$  算法的平均效率就会好得多了。

实现  $A^*$  算法最重要的考虑因素就是内存的使用和计算的时间,直接说就是优先队列如何最大限度地降低时间复杂度和空间复杂度。我们用  $N$  表示单元数目,整个算法的复杂度为  $O(N^2)$ 。考虑空间复杂度,算法开始时,如果每一点顶点需要保存  $g[v]$  和  $h[v]$ ,就要花费  $O(N)$ 。而且,优先队列的每一个成员也需要占据空间,队列的成员最大数目为  $N$ ,因此总的空间复杂度有个上限  $O(N)$ 。实际应用中,优先队列的大小要比  $N$  小得多。

#### 4 矢量地图下算法的实现

我们定义的矢量地图包括地图矢量库和地图数据库两部分,在最短路径问题中,我们主要关心的是地图矢量库。地图矢量库中定义了矢量地图中几何结构(点、线、面)的位置形状信息。

下面主要介绍一下道路的有关定义,因为这些涉及后面的算法实现。

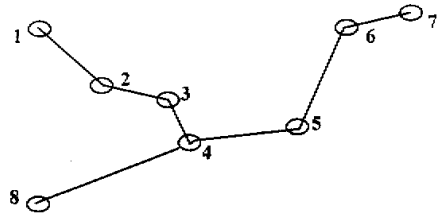
定义 1 矢量边:一些点的坐标的集合,表征着一条连续的折线。

定义 2 大节点、小节点、节点:大节点是矢量边的端点;小节点是指矢量边除端点之外的内部点;大节点和小节点统称节点。

定义 3 弧:一条弧就是一条矢量边,是若干个节点的集合。组成弧的节点是顺序排列的。

定义 4 路:路是若干条弧的集合。

由图 1 看出,弧是节点的集合,路是弧的集合。可见节点、弧、路之间的关系非常密切,也正是这种各元素的“紧密”联系为下一步基于地理空间信息的决策提供了基础。



小节点 2、3、5、6      弧 [ 1 2 3 4 ][ 4 5 6 7 ][ 4 8 ]  
大节点 1、4、7、8      路 以上的弧的任意组合均可能构成路

图 1 几种拓朴结构的定义

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
BEGIN
    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 为从起始桥到当前桥的当前路径长度;Preseat 为从起始桥到当前桥的当前路径的上一段桥保存在 CloseList 中的位置。

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,下面我们就算法的收敛性和正确性加以说明:

(a) OpenList 中结点的 Length 总保持增长的趋势,而出发点为目标点的路径长度有穷,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(图 2 给出优化后算法的工作流程图)。

注意到最短路径的走向总是朝着终止桥的方向,我们可以对算法加以优化:将寻优的估价函数定为  $f(x) = g(x) + h(x)$  其中  $g(x)$  表示从起始桥到当前桥实际走过的长度,  $h(x)$  取当前桥到终止桥的直线距离。这样,在原来的 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。这种优势在稠密网络中体现得非常明显。

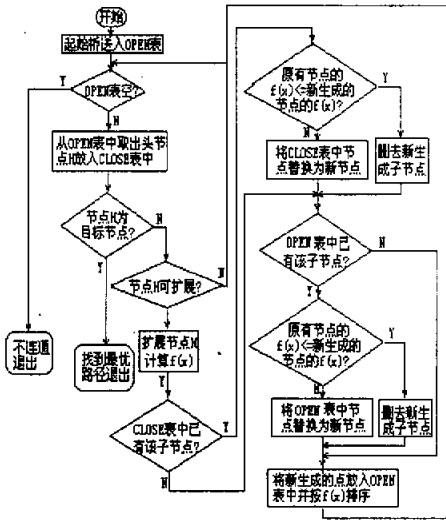


图 2 寻优算法流程图

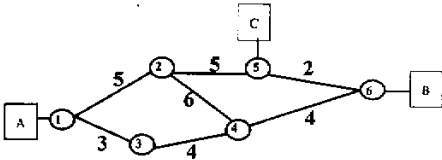


图 3 简单拓扑结构

4	④	⑤⑥②	⑤⑥② (找出路径 :2⑤)
5	⑤	(⑥②⑥)→⑥② (找出路径 :3④⑥)	

下面我们用工图 3 作一示例,对于由顶点 A、B、C 及交叉点 1~6 构成的简单拓扑结构,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,见表 1。

### 5 结果和讨论

笔者利用上述改进的 A\* 算法结合矢量地图的结构实现了最短路径的寻优(如图 4,图中较粗的路线即为选出的最短路径),而且寻优的速度很快,只要 2-3 秒。我们可以看出,对于这样一个比较稠密的道路网络,该算法还是很快捷和有效的。对于启发式的选择,还可以根据不同的寻优策略做相应的修改。另外,探索更有效的数据存储结构,这对算法的进一步完善和提高也是有益的。



图 4 最短路径寻优结果

### [参考文献]

- [1] 郭耀煌,等编著. 筹学与工程系统分析[M]. 中国建筑工业出版社,1986.12.
- [2] 严蔚敏,吴伟民,等编著. 数据结构[M]. 清华大学出版社,1992.6.
- [3] 孙德敏编著. 工程最优化[M]. 中国科学技术大学出版社,1997.11.
- [4] 段凌宇. 城市车辆联网监控系统的设计与实现[D]. 1995.5.
- [5] 姚振旺. GIS 环境矢量电子地图生成校正平台的设计和实现[D]. 1998.5.

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	①	③②	③②
2	③	②④	②④
3	②	(④⑤④)→④⑤	(④⑤④)→④⑤