

改进 Dijkstra 算法在嵌入式 GIS 系统中的应用研究

梁 骥, 张雪燕

(宁波广播电视大学 文法系, 浙江 宁波 315000)

摘要: 针对嵌入式 GIS 系统的特点对经典 Dijkstra 算法进行优化处理. 分两步以提高算法效率, 第一步采用椭圆限制区域算法来缩小算法的搜索范围; 第二步为每个结点添加属性值、增加前趋表, 以辅助算法快速找到一条最短路径. 最后将此算法应用到嵌入式 GIS 系统中, 测试结果表明: 在城市道路网络中, 改进的算法大大提高了嵌入式 GIS 系统的效率.

关键词: Dijkstra 算法; 椭圆限制搜索区域算法; 嵌入式地理信息系统; 前趋表

中图分类号: TP301.6

文献标志码: A

文章编号: 1001-5132 (2013) 01-0053-04

Dijkstra 算法是荷兰计算机科学家 Dijkstra E W 于 1959 年提出的最短路径算法, 它是图论学中求最短路径问题的经典算法, 适用于计算两点间的最短路径问题^[1]. 它采用贪心启发策略, 是目前已知的理论上最完善、实践上运用最广泛的算法, 具有极强的抗差性和稳定性.

但是经典 Dijkstra 算法在搜索过程中由于缺乏方向性, 即使在找到终点时结束算法, 仍然有大量的冗余计算, 在实际应用中也有一定的难度. 因此自 Dijkstra 算法提出以来, 众多科研人员就最短路径问题提出了很多改进方法. 过去的研究主要集中在基于数据存储结构的优化(采用邻接表、十字链表等)^[2-4]限制搜索区域(陆峰在 1999 年提出的椭圆限制搜索区域)^[2]以及优化优先级队列结构(基于堆结构和桶结构)^[5]方面. 笔者提出了针对单源单汇点的最短路径算法, 先限制搜索区域来减少计算量, 再根据城市道路的特点修改存储结构, 并将它应用到 GIS 决策系统中以检验其实际效果.

1 Dijkstra 的 SSSP 算法

1.1 经典 Dijkstra 算法的主要思想

Dijkstra 算法是一种比较-添加模型, 它将结点分成三部分——未标记、永久标记和临时结点.

设图 $G(V, E)$, w 为权值. 算法将 V 分成 S 和 T 2 个子集, S 存放永久标记结点, $T = V - S$, 表示目

前还没有计算的结点. $d[N]$ 是长度为 N 的一维数组, 用来存放从源点到其他结点的最短路径长度.

求解从源点 s 到 i 的最短路径的过程如下:

(1) 初始化, $S = \{ \}$, $T = V$;

(2) 构建图的邻接矩阵为 $Neighbour[N][M]$, 其中, N 为结点数, M 为最大邻接结点数. 并根据 $Neighbour[N][M]$ 来初始化 $d[N]$;

(3) 从 $d[N]$ 中选择最小的元素, 并假设此结点编号为 i ;

(4) 将结点 i 放到 S 中, 并从 T 中删除; 利用结点 i 来更新 $d[N]$ 数组:

If ($d[j] > d[i] + judge[i][j]$),

$d[j] = d[i] + judge[i][j]$,

其中, $judge[i][j]$ 为判断矩阵, 它的值可参照矩阵 $Neighbour[N][M]$ 来获得.

(5) 重复(3)和(4), 循环结束条件为 $T = \{ \}$.

1.2 Dijkstra 算法分析

从上述 Dijkstra 算法流程中可见, 直接影响算法的速度是顶点集 V 的大小. 由于步骤(3)和步骤(4)是个循环比较过程, 此时如果不经任何处理就选择 1 个权值最小的结点, 那么算法就必须扫描顶点集 V 中所有的结点. 此时, 整个算法的时间复杂度为 $O(N^2)$, 对于嵌入式系统来说, 资源耗费过多.

其次, 由于传统 Dijkstra 算法获得的最短路径上的结点是无序的, 而对于嵌入式 GIS 系统中的用

户来说,要求看到的是能在图上显示的路径,也就是说必须是有序的^[6].

而在实际应用中,由于城市的道路网络异常复杂,用户的指令也实时产生,并且往往随机和不确定地定义起点和终点.因此,此时经典 Dijkstra 算法就无法满足这些需求,但单源单汇点算法就较为适合这些要求^[6].

2 改进的 Dijkstra 算法

2.1 应用椭圆限制搜索区域法处理搜索范围

由 Jonson 算法可知,在计算两点间的最短路径时,无需遍历搜索整个道路网络,而只需搜索局部的道路网络就可得到它们间的最短路径.因此,此时可通过预处理结点集合,来进一步缩小结点集合的范围以提高算法效率.而椭圆模型是控制路网规模算法中最常用的一种^[7-8].

定理 假设图 $G(V, E)$, S 为源点, D 为终点($S, D \in V$). 对于 G 中结点 i 到 S 和 D 的欧式距离分别为 $|SI|$ 和 $|DI|$, 则 $|SI| + |DI| \leq M$, 其中, M 为最短路径对应的大致极限距离(图 1).

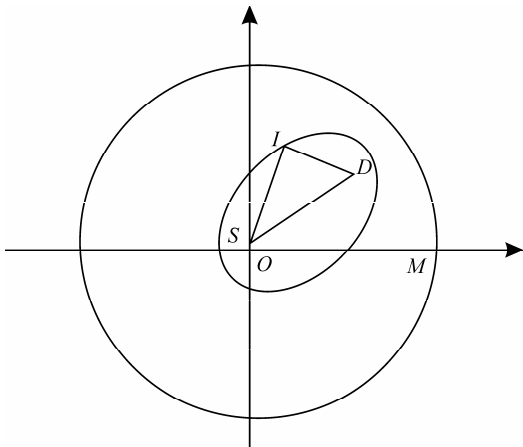


图1 椭圆限制搜索区域

显然, I 的轨迹正好形成了一个以 S 和 D 为焦点, M 为长轴的椭圆, 因此搜索区域只需考虑椭圆范围之内即可. 而本文中的 M 是针对杭州市城市道路网络情况, 其获取步骤如下:

(1) 选取典型的城市道路区域网络, 构造 2 个结点集合 T_1, T_2 .

$$C = T_1 \times T_2 = \{(t_1, t_2) | t_1 \in T_1, t_2 \in T_2\},$$

其中, C 中的元素 (t_1, t_2) 是待求最短路径的起始点和目的地对.

(2) 计算 C 中每个元素结点对间的欧氏距离 E_{ij} 以及最短路径长度 D_{ij} , 此时的比值系数集合 R :

$$R_{ij} = E_{ij} / D_{ij}.$$

(3) 对 R 进行分析后求得 δ , 并且使得 R 的总数满足置信水平阈值 φ , $\varphi=0.95$ 即表明在椭圆内找到全局最短路径的可能性 $\geq 95\%$. 则椭圆的长轴 M 为 $M = \delta \cdot E_{ij}, R_{ij} < \delta$.

(4) 对于 $(x_i, y_i), (x_j, y_j)$ 两点, 得到限制椭圆方程为:

$$\frac{[\cos \phi(x-a) + \sin \phi(y-b)]^2}{A^2} + \frac{[-\sin \phi(x-a) + \cos \phi(y-b)]^2}{B^2} = 1,$$

其中, a, b 分别为椭圆中心位置坐标; ϕ 为 x 轴正方向与椭圆长轴的夹角; A, B 分别代表椭圆的长半轴和短半轴.

2.2 生成前趋表排序最短路径上的结点

由于 Dijkstra 算法不对求出的结点进行排序, 此时可引入前趋表的概念. 前趋表(hpr table)主要由头结点和它的 n 个链表组成, 其中的 n 为路网结点数, 其结构如图 2 所示.

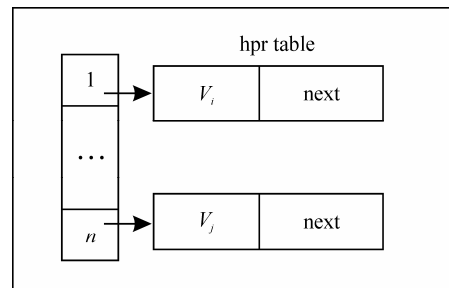


图2 hpr 表结构

图 2 中的 $1, \dots, n$ 为结点编号, V_i, V_j 为直接前趋结点($0 \leq i, j < n$, n 为结点总数), $next$ 为指向下个直接相连的结点指针. 此时的前趋表中存储了目前路径上结点的所有直接前趋, 因此只需沿着终点就能找到它的直接前趋结点, 最后直至源点. 此过程就能顺利地解决一条按顺序进行排序的最短路径上的结点集.

而图 3(a)为道路节点网络, 图 3(b)则为其对应的前驱图. 这样, 利用前趋表可以很方便地求出 V_1 至 V_6 的最短路径(V_1, V_3, V_5, V_6).

2.3 算法基本思路

设前趋表为 hpr , 源点为 V_s , 终点为 V_d , 改进

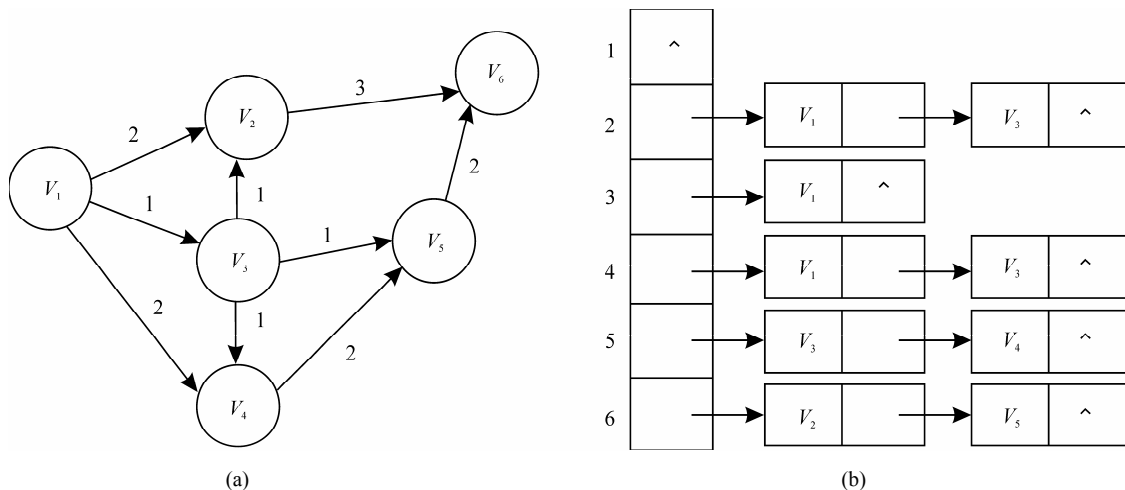


图 3 结点网络和 hpr 表

表 1 原始 Dijkstra 算法和改进的 Dijkstra 算法比较

起点	终点	N_{sd} /个	N_{sd}' /个	T_{sd} /s	T_{sd}' /s
(79.26, 82.18)	(81.33, 80.24)	1950	312	1.564	0.580
(80.76, 82.68)	(82.01, 81.65)	3731	3731	3.060	3.582
(83.45, 80.83)	(82.25, 79.60)	1604	331	1.328	0.654
(79.58, 80.31)	(81.15, 79.22)	1853	258	1.492	0.565
(79.16, 82.60)	(79.16, 82.58)	566	107	0.518	0.243

的算法步骤如下:

(1) 利用 2.1 划定源点至终点的椭圆搜索区域, 判断条件可放宽至椭圆的外接矩形.

横坐标的取值范围为:

$$(a - \sqrt{A^2 \cos^2 \phi + B^2 \sin^2 \phi}, a + \sqrt{A^2 \cos^2 \phi + B^2 \sin^2 \phi});$$

纵坐标的取值范围为:

$$(a - \sqrt{A^2 \sin^2 \phi + B^2 \cos^2 \phi}, a + \sqrt{A^2 \sin^2 \phi + B^2 \cos^2 \phi});$$

(2) 将源点 V_s 放入永久标记集合 S 中, 计算出所有结点的 $distance$ 属性值,

$$distance = cost[S][k] + D[k][d],$$

$$dist[k] = cost[S][k] + D[k][d].$$

如果 $V_k distance \neq \infty$, 那么 $hpr[k]$ 此时仅有 1 个直接前趋值 V_0 , 否则 $pr[k] = \wedge (0 < k \leq n-1)$;

(3) 如果当前处理的结点即为终点, 或者 S 中结点的范围超出步骤(1)的取值范围, 则算法退出;

(4) 选择剩余结点中 $distance$ 最小的结点 k , 并将其标记为永久结点, 放入 S 中;

(5) 如 $dist[i] > dist[k] + cost[i][k]$, 则

$$dist[i] = dist[k] + cost[i][k],$$

$$V_i distance = dist[i] + D[i][d].$$

删除 $pr[i]$ 中所有的直接前趋, 建立新的直接前趋结点 k ; 如果 $dist[i] = dist[k] + cost[i][k]$, 则直接把结点 k 插入到 $pr[i]$ 中.

综上所述, 较之经典的 Dijkstra 算法, 改进算法能大大减少搜索范围, 减少执行的时间.

3 实验结果

在嵌入式 GIS 系统中应用上述算法, 运行环境如下: Windows CE 操作系统, Pentium 4 CPU 3.60 GHz, 2 G 内存. 地图选用杭州市市区图, 相关坐标为北京 54 坐标系, 1:10000 的地籍数据采集, 其中总共包含 3982 个道路交点.

表 1 中的原采集坐标数据精确到小数点后 13 位, 而文中取值精确到小数点后 2 位. N_{sd} 为经典 Dijkstra 算法实际标记的结点数, N_{sd}' 为改进算法实际标记的结点数, T_{sd} 为 Dijkstra 算法所需时间, T_{sd}' 为改进算法所需时间. 而表 1 中的 (80.76, 82.68) 到 (82.01, 81.65) 由于算法没有在一定范围内找到

通路, 因此造成改进算法运算时间反而增加. 对于落在椭圆范围内的结点对而言, 搜索范围基本上小于原来算法的 $1/4$ (平均可达到 0.235 左右). 但是由于道路网络实际分布上的差异, 势必会造成某些样本的实际搜索范围可能 $\geq 1/4$.

4 结语

对道路地图的实验证明, 改进算法卓有成效, 大大提高了系统的运行效率. 但选取具体哪一种算法来进行对道路地图处理, 最后还是要根据对实际的道路网络情况进行分析; 并且对于连通性较差的道路而言, 文中的改进 Dijkstra 算法也不适用, 且只会增加额外计算时间和相关存储量^[8].

参考文献:

- [1] Zhan F B. Three fastest shortest path algorithms on real road networks: Data structures and procedures[J]. Journal of Geographic Information and Decision Analysis, 2001, 1(1):69-82.
- [2] 陆锋, 卢冬梅, 崔伟宏. 交通网络限制搜索区域时间最短路径算法[J]. 中国图象图形学报, 1999, 4(10):849-853.
- [3] 王开义, 赵春江, 青桂仙, 等. GIS 领域最短路径搜索问题的一种高效实现[J]. 中国图象图形学报, 2003, 8(8):951-956.
- [4] 王海梅, 周献中. 网络系统中的最短路径分析及其应用研究[J]. 兵工学报, 2006, 27(3):515-518.
- [5] 陆锋. 最短路径算法: 分类体系与研究进展[J]. 测绘学报, 2001, 30(3):269-275.
- [6] Wayne Wolf. 嵌入式计算机系统设计原理[M]. 北京: 机械工业出版社, 2002.
- [7] Stig N, Bengt R. Computer cartography shortest route programs[M]. Sweden: The Royal University of Land, 1969.
- [8] 崔伟宏. 空间数据结构研究[M]. 北京: 中国科学技术出版社, 1995.

An Improved Dijkstra Algorithm in Embedded-GIS System

LIANG Ji, ZHANG Xue-yan

(Department of Literature and Law, Ningbo Radio & TV University, Ningbo 315000, China)

Abstract: For the embedded system, a more efficient implementation of Dijkstra algorithm for embedded geographic information system is presented. The algorithm's efficiency is improved by two steps: first, reduce the scanning scale by applying elliptic restricted area-searching algorithm; second, add property value to each node and a priority table, which helps the algorithm to find a shortest path quickly. The improved algorithm is applied to the given embedded-GIS system, which suggests the improvement of the efficiency of the embedded-GIS system greatly in urban road networks.

Key words: Dijkstra; elliptic restricted area-searching algorithm; embedded-GIS; priority table

(责任编辑 章践立)