第 37 卷 第 17 期
Vol.37 No.17

计 算 机 工 程
Computer Engineering

2011 年 9 月
September 2011

·开发研究与设计技术· 文章编号：1000—3428(2011)17—0279—03 文献标识码：A 中图分类号：TP301.6

港区导航系统中最短路径搜索算法

陆 檩，李世杰，王贵甫，闵新力，张 余，高 珊

(上海申腾信息技术有限公司，上海 200040)

摘 要：分析 Dijikstra 算法、限制区域搜索算法以及 A*算法的时间复杂度和空间复杂度，提出一种最短路径搜索算法。将静态存储和动态搜索相结合，以限定区域搜索算法为主、A*算法为辅，并根据港区路况实现该算法。实验结果表明，在区域路网结构相对比较规则的情况下，该算法能够提高路径搜索的效率。

关键词：最短路径搜索算法；静态存储；动态搜索；限定区域搜索算法；A*算法

Shortest Route Search Algorithm in Harbor Guided System

LU Lin, LI Shi-jie, WANG Gui-fu, MIN Xin-li, ZHANG Yu, GAO Shan

(Shanghai Shenteng Information Technology Co., Ltd., Shanghai 200040, China)

【Abstract】Analysing the time complexity and space complexity of Dijkstra Algorithm, restricted area search algorithm and A* algorithm, on the bases this, the paper proposes a shortest route search algorithm. The algorithm makes the static memory and dynamic search combined, uses the restricted area research algorithm as the main algorithm, is supplemented by A* algorithm, according to the port road condition to realize the algorithm. Experimental results show that the algorithm can enhance the route searching efficiency in the occasion of regular road network structure.

【Key words】shortest route search algorithm; static store; dynamic search; limited area search algorithm; A* algorithm

DOI: 10.3969/j.issn.1000-3428.2011.17.094

1 概述

目前，多数系统最短路径有 3 种搜索算法，分别是 Dijikstra 算法、限制区域搜索算法和 A*算法，且对于不同路径，以上 3 种算法的运算时间各有优劣[1]。

为使港区最短路径搜索算法的搜索效率达到最高，本文以前面提到的 3 种算法为基础，提出一种以限定区域搜索算法为主、A*算法为补充的新算法。

2 常用的路径搜索算法

2.1 Dijkstra 算法

文献[2-3]提出的 Dijkstra 算法是电子地图解决最短路径问题的理论基础，其基本定义如下：给定一个带权有向图 $G=(V, E)$，其中，每条边的 E 权是一个非负实数，V 是 G 中所有顶点的集合，另外给定 V 中的一个顶点，称为源。

设置一个集合 S 并不断扩充。一个顶点，当且仅当从源到该顶点的最短路径长度已知时该顶点属于集合 S。初始时，S 仅含有源。设 v 是 G 的某一个顶点，将从源到 v 且中间只经过 S 的路径称为从源到 v 的特殊路径，并用数组 D 记录源到当前每个顶点的最短特殊路径长度。

Dijkstra 算法的实现过程为：每次从 V 减 S 中取出具有最短特殊路径长度的顶点 v，加到 S 中，同时，对数组 D 进行修改。当 S 包含所有 V 中顶点时，则 D 记录从源到所有其他各顶点的长度。

2.2 限定区域搜索算法及搜索区域比较

道路网是一种大规模的、分散的且又互连的图，其结构相对简单，在用 Dijkstra 算法寻找最短路径时，可从起点开始搜索，展开一个半径到达目标点的圆，则对指向目标点方向相反区域的大多数搜索没有用。文献[4]通过去除不可能包含最短路径区域的方法，提出一种专为车辆导航系统寻找最短路径的最优方法。

Dijkstra 算法的搜索方式是按顶点距源点的距离递增的顺序产生最短路径，搜索方向随时都准备向其四面八方展开，具有很大的盲目性，而限定区域搜索方法的搜索方法却不同，具体 2 种算法的搜索区域对比结果如图 1 所示。

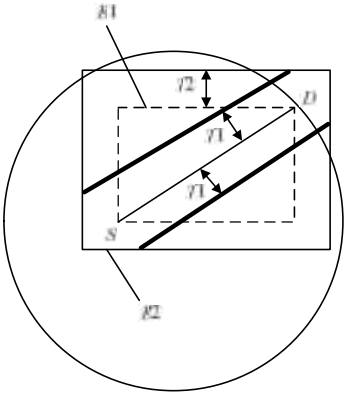

图 1 Dijkstra 和限定区域搜索算法搜索区域

由图 1 可知，Dijkstra 算法最终扫过的搜索区域是以起始点为原点，起始点与目标点的欧氏距离为半径的一个圆。

基金项目：上海市科学技术委员会重点攻关专项基金资助项目(08DZ1500900)

作者简介：陆 檩(1974－)，男，高级工程师，主研方向：无线定位算法；李世杰，硕士；王贵甫、闵新力，高级工程师、博士；张 余，工程师；高 珊，硕士

收稿日期：2011-04-07 **E-mail：**ericlu74@yahoo.com.cn

限定区域搜索方法只在矩形 $R2$ 被 2 条粗线切开剩下的一小部分区域中进行搜索，而不是搜索整个圆区域。矩形 $R1$ 内有对角线 SD，而矩形 $R2$ 是以 $T2$ 为门限从矩形 $R1$ 扩展的矩形。这 2 条粗线与 SD 相距 $T1$ 且平行于 SD。为保证最优路径包含在限定区域内，需要决定 $T1$ 和 $T2$ 为 2 个变量。

2.3 A*搜索算法

A*算法在搜索过程中引入启发式方法[5]，节点的选择基于开始节点的耗费和接近目标点的估计(启发式估计)，而不单是基于选择最低耗费的下一个节点(从起始点测量得到)。文献[4]为解决最优路径搜索描述了这一方法。

A*算法由指向父亲结点的指针、OPEN 表、CLOSE 表和相当于图结构中权值的每个结点的 F 值组成。假设启发式算法为 $F=H+G$，其中，H 值为从当前结点移动到目标结点的预估移动耗费。

3 时间复杂度比较

3.1 Dijkstra 算法时间复杂度

假设第一次循环的运算时间是 $O(V)$，在每一轮循环中，数组 Extract_Min 运算时间是 $\mathrm{lb}V$，循环内反复检查与当前节点相邻的节点，总运算时间是 $O(E)$，则算法时间复杂度为：

$$O\left((V+E)\times \mathrm{lb}(V)\right)=O\left(E\times \mathrm{lb}(V)\right)$$

随着节点数量增加，本算法的运算时间会变得越来越长。

3.2 限定区域搜索算法时间复杂度

若路网的节点为平均分布，则假设如下：

(1)密度为 C。

(2)V^*：使用 Dijkstra 算法检查的顶点数目。

(3)V：使用限制区域搜索算法检查的顶点数目。

(4)$Sdij$：使用 Dijkstra 算法搜索的区域。

(5)$Sres$：使用限制区域搜索算法搜索的区域。

(6)R：起始点与目标点的欧氏距离。

(7)K_1, K_2：门限因数。

则 P 可以用来描述效率提高的比率，即：

$$P=V/\Delta V=$$
$$C\times Sre_s/C\times Sd_{ij}\leqslant$$
$$2\times T_1\left(R+T_2\times\sqrt{2}\right)/\pi R^2=$$
$$2\times K_1\times R\left(R+K_2\times R\sqrt{2}\right)/\pi R^2=$$
$$2\times K_1\times\left(1+\sqrt{2}\times K_2\right)/\pi$$

其中，K_1 和 K_2 是非常小的数字，则效率得到很大提高。一般假设 K_2=0.4，则 $P\approx K_1$。

3.3 A*算法时间复杂度及 3 种算法比较

该算法不会改善最差情况下的时间复杂度，但可改善平均时间的复杂度，并不像 Dijkstra 算法那样盲目，而具有方向性。由于最短路径从起始点开始，扩展到从起始点指向目标点的节点，因此，运算时间比 Dijkstra 算法的运算时间要短许多。

将上述 3 种算法随着路径长的增加，平均运算时间的变化作对比，且对比 3 种算法的运算时间和精度，其中，距离表示路径长度，此算法搜寻路径的长度与 Dijkstra 算法搜寻路径的长度的比值，定义为对于某一特定路径的某一算法的精度。

通过 3 种算法的比较可得到如下结论，通过使用欧几里德启发式方法，限定指向目标点的搜索区域，将运算时间减少大约一半，故限定区域搜索算法比 Dijkstra 算法更加有效。比较结果如图 2 所示。

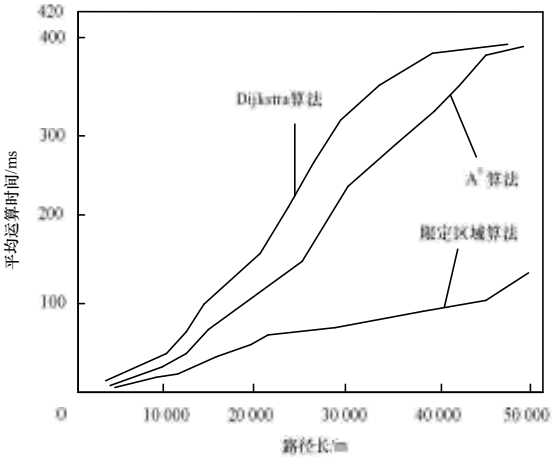

图 2 3 种算法路径长与平均运算时间的对比

由图 2 可知，Dijkstra 和 A*算法的运算时间在路径长达到 30 000 m 之前增加得快，然而，由于城市地图是有限的，因此，当路径长超过 30 000 m 时，在搜索范围之外没有更多的未搜寻点，故搜索空间不会随着路径长增长而增加地太大，对于运行时间缩短的速率则会降低。

4 本文算法思想

本文根据某区路网和车载导航系统 GPS 的实际情况，采用车载 GPS 终端机静态存储路径，结合按需的动态路径搜索算法实现“动静”结合的区域导航算法。

4.1 静态存储与动态搜索

由于某区路网分布具有初始结点(卡口)和目标结点(泊位区)位置固定且数量有限的特点，因此可在车载的 GPS 终端机中，静态存储每个卡口到各个泊位区的最短路径。当车辆到达卡口并获得卡口坐标和泊位坐标后，从所存路径中选出对应的最短路径，则以后就不用调用算法进行计算，省去等待 GPS 接收机启动稳定的时间，以上即为路径静态存储。

司机一般都会按照 GPS 终端机所指示的路径驾驶，故使用静态存储的方法可较大提高导航的实时性，提高卡口进港作业的效率，但静态存储也有其不完善的地方，需要动态路径搜索算法对其补充。

当车辆偏离 GPS 终端机所指示道路，或由于人为或自然原因导致所选道路不通畅时，需要 GPS 终端设备调用最短路径搜索算法动态搜索出一条从当前位置到达目的地的最短路径，以上即为动态搜索。

4.2 带阻塞的 A*算法与限制区域搜索算法的结合

本文采用将带阻塞的 A*算法与限制区域搜索算法相结合的动态搜索算法。

矩形限制区域的设定方法为如下步骤：

Step1 以起始点为原点，以水平方向为 X 轴，以垂直方向为 Y 轴建立直角坐标系。

Step2 构造一个以起始点和目标点之间的连线为对角线的矩形搜索区域。

Step3 为该搜索区域设定限制距离 $T1$ 和扩充值 $T2$。

在算法实现过程中，将每条路定义为一个 $Link$，一条 $Link$ 包含如下信息：

(1)起点坐标：($X0$, $Y0$)；

(2)终点坐标：($X1$, $Y1$)；

(3)方向：单向($X0$, $Y0$)－>($X1$, $Y1$)为正向，($X1$, $Y1$)－>($X0$, $Y0$)为反向，($X0$, $Y0$)<－>($X1$, $Y1$)为双向；

(4)道路长度；

(5)*curlinkidx*：记录当前 Link 在 Link 集合中的索引值；

(6)*prelinkidx*：记录当前 *Link* 的前驱。当其值取 0 时，表示当前 *Link* 未被选中；当其值取 *curlinkidx* 时，表示前驱为 *Start*；其他情况下，取 *Link* 的索引值。将后面 2 种情况作为选中的 *Link*。

路径(Path)计算中 *Link* 集合的结构定义如下：

```
//Link table struct
typedef struct_Path_Link_Table
{
unsigned short curlinkidx;    //Current link's index
unsigned short prelinkidx;    //Previous link's index
unsigned long distcount;      //Unit: m, Distance of links
}PATHLINKTABLE;
```

区域设定后，寻路算法流程如图 3 所示。

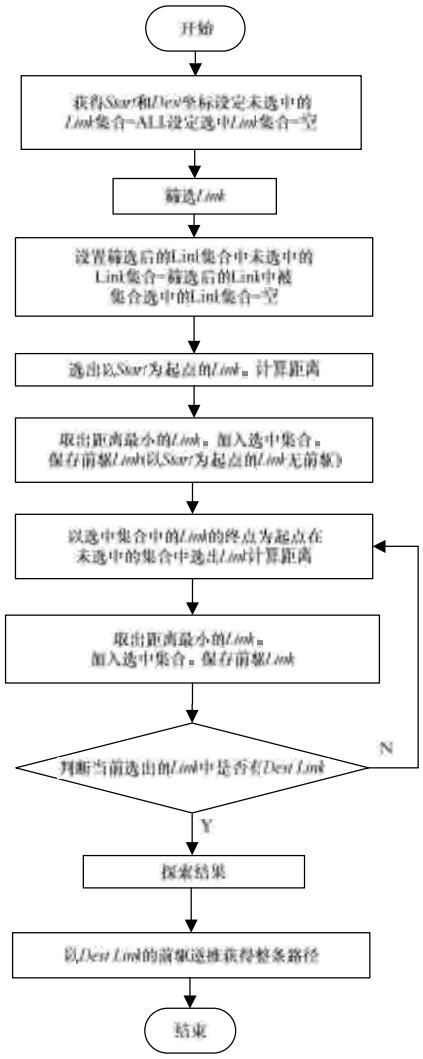

图 3　限定区域搜索算法流程

限制区域搜索算法的基础是 Dijkstra 算法，但限定区域搜索算法是通过检查这些节点是否在范围之内，将限定区域外的节点滤除，而不是在每一次循环中检查所有相邻节点，此种检查程序操作方式是通过检查每个 *Link* 的起始坐标是否在限定区域内来完成的。此种方法的举例说明如图 4 所示。由图 4 可知，只有当起点坐标和终点坐标都在限定区域内部时，节点才在此区域中，即 *L*1、*L*2、*L*3、*L*4，其中，只有 *L*4 在限制区域内。其中，*T* 是一个扩充值，当第一次搜索失败时，算法会先扩大搜索区域，然后在扩大后的搜索区域中再次搜索，如果还是搜索不到，则用 A*算法进行搜索。在城市中的道路中，*T* 是 1 500 m，本文考虑的车辆行驶在城郊区域，故可将 *T* 的范围缩小至 800 m。

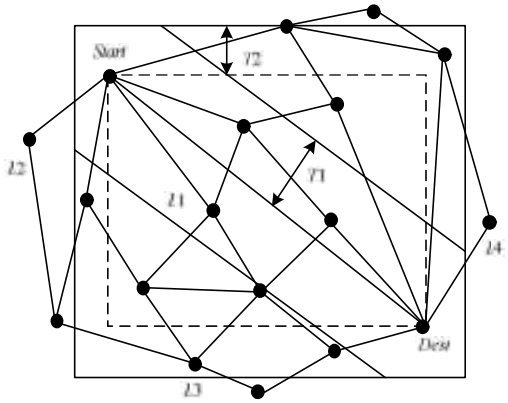

图 4　*Link* 筛选方法

限制区域搜索算法存在不完整性搜索的问题，故该算法可能搜索不到路径。例如在起点和终点之间存在着一条必经的道路，而存在地理和工程方面的因素，故这条道路恰好不在搜索范围以内，虽然这种情况出现的概率很小，但在大样本空间下多次独立实验中出现的可能性是不能被忽略的，这时限制区域搜索算法将无法找出所求路径，此时将引入 A* 算法作为补充。

本文所用的 A*算法先对地图上的节点数据做排序处理，用堆排序的方法对每条路的路径长度进行排序，再运用 A* 算法搜索路径。定义 *H* 值为当前结点与目标节点之间的直线距离，此距离可通过 2 点的坐标计算得出。当前结点与父亲结点的距离定义为 *G* 值，可以直接从这 2 个结点所定义的 *Link* 的长度计算得出。根据上文分析，给出 *Link* 判断结果如表 1 所示。

表 1　*Link* 的判断结果

Link 号	(*X*0, *Y*0)位置判断	(*X*1, *Y*1)位置判断	结果
*L*1	区域内	区域内	保留
*L*2	区域外	区域外	舍弃
*L*3	区域内	区域外	舍弃
*L*4	区域外	区域内	舍弃

如表 1 所示，*L*1 的起点坐标和终点坐标都在所选区域以内，故 *L*1 被保留。*L*2 的起点坐标和终点坐标都在所选区域以外，故舍弃。*L*3 的起点坐标在所选区域内，但终点坐标不在所选区域内，这样的 *Link* 也被舍弃，同理，*L*4 也被舍弃。

5　实验及结果

对于实际城市道路网络结构相对比较规则的最短路径的规划，限制搜索区域最短路径规划算法无须进行大量的乘积与开方计算，相对于 Dijkstra 算法，能够极大降低算法的搜索规模。结合上文分析分析，限制搜索区域的最短路径规划算法理论而言可提高最短路径规划的效率。

为验证限制搜索区域算法的效率，选取某市 8 个行政区共 6 616 个 *Link*，包含 4 525 个节点的地图数据作为实验数据。工具采用 Visual C++ 6.0，数据库采用 Oracle 10 g，数据库连接技术采用 OO4O(Oracle Object for OLE)，数据结构采用邻接表，硬件采用酷睿双核 CPU 2.00 GHz，2 GB 内存。

对于采用不同因数的情况，分别计算只采用限定区域算法的搜索时间和只采用 Dijkstra 算法的搜索时间。并对数据

(下转第 285 页)