[经营·管理]

DOI:10.3969/j.issn.1005-2895.2019.03.021

取档机器人路径规划的改进 Dijkstra 算法

罗晓冬，张秋菊

（江南大学 机械工程学院，江苏 无锡 214122）

摘 要：档案库取档机器人在使用传统 Dijkstra 算法进行路径规划时，存在无法筛选出拐弯数最少、经过节点数最少的最短路径等缺点，提出了一种改进型的 Dijkstra 算法。首先针对档案库平面布局建立基于拓扑法的电子地图；然后根据任务需求，建立最短路径搜索数学模型；采用 Dijkstra 算法并结合深度优先遍历算法筛选出任意 2 个节点间的所有最短路径，并找出花费代价最小的路径。最后对改进的算法进行仿真实验，结果表明，改进后的 Dijkstra 算法可以有效地提高取档机器人的运行效率，可以用最小的行驶代价到达目标点。

关 键 词：取档机器人；路径规划；Dijkstra 算法；深度优先遍历算法；最小行驶代价

中图分类号：TP242.6　　**文献标志码**：A　　**文章编号**：1005-2895(2019)03-0101-04

Dijkstra Algorithm in Path Planning of Archives Accessing Robot

LUO Xiaodong，ZHANG Qiuju

（School of Mechanical Engineering，Jiangnan University，Wuxi，Jiangsu 214122，China）

Abstract：An improved Dijkstra algorithm was proposed to overcome the shortcomings of path planning for archives accessing robot，such as the inability to select the shortest path with the least number of turns and the least number of nodes when using the general Dijkstra algorithm. First，the electronic map based on topological method was established according to the layout of archives. second，the shortest path search mathematical model was established according to the mission requirements. Dijkstra algorithm combined with depth-first traversal algorithm was used to select all the shortest paths between two locations and find the path with the least cost. Finally，the improved algorithm was simulated. The results show that the improved Dijkstra algorithm can effectively improve the efficiency of the robot and can reach the target point with the lowest cost.

Keywords：archives accessing robot；path planning；Dijkstra algorithm；depth-first traversal algorithm； minimum driving cost

随着社会的发展，各单位档案数量日益增加，增加了人工存取档案的工作量。机器人调档系统的实现，能够减轻工作人员的工作负担，提高工作效率。选取行程最短、代价花费最小的路径是取档机器人的一个重要问题。对于路径规划，目前主要的方法有 Dijkstra 算法、A * 算法和遗传算法等，Dijkstra 算法因其简洁易懂得到了普遍的应用。目前对 Dijkstra 算法的研究主要集中在对算法本身进行改进和算法应用两方面。金婷[1]在传统 Dijkstra 算法基础上对每个顶点增加前置邻结点属性，改进后的算法能够求解多条路径问题。王树西[2]分析了多邻接点问题与多条最短路径问题的成因，改进后的算法可以有效解决这两类问题。孙强[3]在 Dijkstra 算法的基础上使用了一些独特的数据结构，改进后的算法能高效率地求出图中一个顶点到其它各顶点的所有最短路径。施剑烽[4]在 Dijkstra 算法基础上添加了多个代价因素评价指标，使得 AGV 在运行时所消耗代价更少。机器人在行驶的过程中过多的转弯会增加机器人的运行时间与控制难度，同时也容易使机器人偏离运行轨道。基于此课题组提出了一种改进的 Dijkstra 算法，可筛选出所有最短路径，并在所有最短路径中找出转弯最少，经过节点最少的路径，可以有效地降低机器人在行驶时的运行消耗。

收稿日期：2018-10-18；**修回日期**：2019-03-13

第一作者简介：罗晓冬(1991)，男，江苏扬州人，硕士研究生，主要研究方向为机电一体化。E-mail：530082581@qq.com

1 取档机器人运行环境的分析与表达

运行环境的表达是机器人路径规划的基础，一般通过电子地图进行描述。目前对电子地图描述的方式的多种多样，主要有拓扑法、栅格法和可视图法等。其中拓扑地图法能直观地模拟环境地图，同时具有高效性和精确性[5-6]，因此笔者选择拓扑地图作为环境建模方法。图 1 所示为档案库拓扑地图。图中的节点对应记录相应位置的 RFID 标签，当取档机器人接收到指令后会先行驶到对应档案架的标签处；在此位置上，当升降抓取机构就位后开始进行档案的查找与抓取。拓扑图中的引导路径对应实际的行驶路径，笔者将各节点及其之间存在的节点连线简化为无向的连通图。在计算机中，对于稠密图适宜采用邻接矩阵来表示，而对于稀疏图则适合采用邻接表来表示[7]。由于笔者研究的拓扑地图属于稠密图，采用邻接矩阵 $\boldsymbol{A}$ 来表示比较方便。

$$\boldsymbol{A}=\begin{pmatrix}
0 & 400 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \cdots & \infty & \infty & \infty & \infty \\
400 & 0 & 1\,400 & \infty & \infty & \infty & \infty & \infty & \infty & \cdots & \infty & \infty & \infty & \infty \\
\infty & 1\,400 & 0 & 1\,400 & \infty & \infty & \infty & \infty & \infty & \cdots & \infty & \infty & \infty & \infty \\
\infty & \infty & 1\,400 & 0 & 1400 & \infty & \infty & \infty & \infty & \cdots & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & 1\,400 & 0 & 1\,400 & \infty & \infty & \infty & \cdots & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & 1\,400 & 0 & 1\,400 & \infty & \infty & \cdots & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & 1\,400 & 0 & 1\,400 & \infty & \cdots & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & 1\,400 & 0 & 400 & \cdots & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 400 & 0 & \cdots & \infty & \infty & \infty & \infty \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \cdots & 0 & 1\,400 & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \cdots & 1\,400 & 0 & 1\,400 & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \cdots & \infty & 1\,400 & 0 & 400 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \cdots & \infty & \infty & 400 & 0
\end{pmatrix}。$$

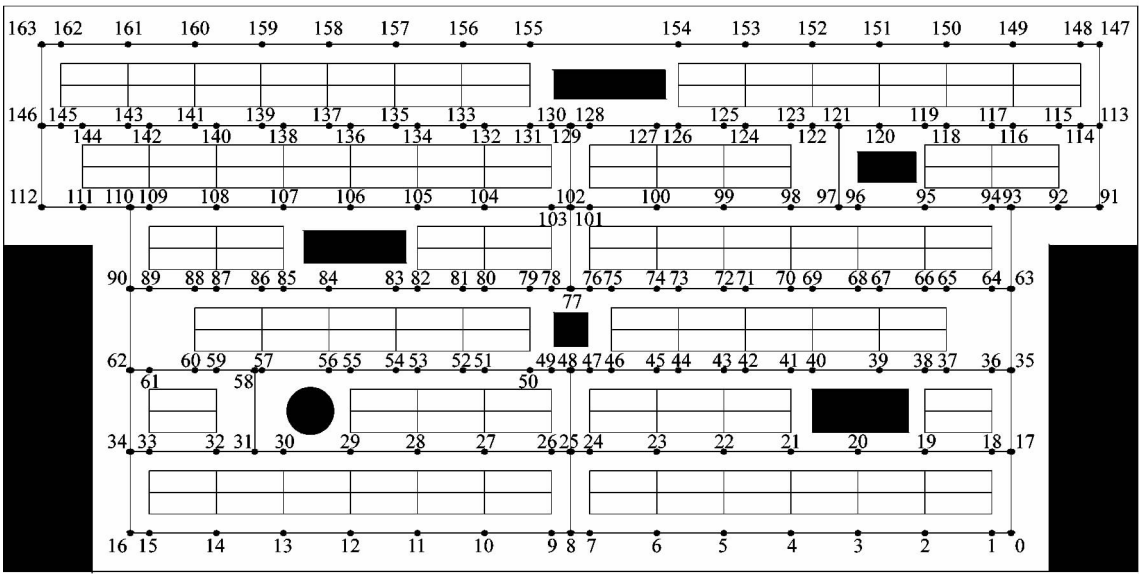

图 1 档案库拓扑地图

Figure 1 Topological map of archival repository

2 最短路径搜索数学模型的建立

高效率的路径搜索表现为两点之间的总路径最短，同时机器人在行驶的过程中转弯次数最少，再者为了减少计算机处理负担，机器人经过的节点数也要最少。建立数学模型前假设以下条件成立：

1）将机器人抽象成一个质点，不计其长、宽、高。

2）最优路径判别指标的优先级为：经过节点数最少 < 转弯次数最少 < 总路径最短。

3）机器人匀速运行且运行时不发生碰撞等情况。

规定把档案库中的每个节点看成该无向图中的顶点 v,节点与节点之间的路径看成是图中的边 e。由此可用图论的形式把整个地图表示出来:

$$G=(V,E)。$$

式中:V 是图 G 中顶点的集合,记为 $V=\{v_1,v_2,v_3,v_4,\cdots,v_n\}$;$E$ 是图 G 中边的集合,记为 $E=\{e_1,e_2,e_3,e_4,\cdots,e_n\}$,$e$ 为点的无序对(v_i,v_j),且 $i\neq j$。

由上述条件,建立数学模型:

$$\min D=\min\sum_{m,n}L(e_i); \tag{1}$$

$$\min T_{\text{turn}}=\sum_{m,n}t_{\text{turn}}(v_i); \tag{2}$$

$$\min N_{\text{node}}=\sum_{m,n}n_{\text{node}}(v_i)。 \tag{3}$$

式中:m,n 为节点,其中 $m,n=1,2,3,\cdots,161$,且 $m\neq n$;D 为总路径长度;$L(e_i)$表示一条路径中相邻两编号节点之间的距离;T_{turn}为拐弯次数;$t_{\text{turn}}(v_i)$表示一次拐弯;N_{node}为节点数;$n_{\text{node}}(v_i)$ 表示经过一个节点,其中 $i\neq j$。

3 取档机器人路径规划算法设计

3.1 传统 Dijkstra 算法与深度优先遍历算法

Dijkstra 算法是图论 $G=(V,E)$ 中用来求解初始节点到其余节点最短路径的算法。它的搜索方式是从初始节点开始逐层向外扩展,直到所有节点都被搜索到为止[8-9]。

深度优先遍历算法(DFS)是沿着树的深度遍历树的节点,尽可能深的搜索树的分支。它的搜索方式是首先以一个未被访问过的顶点作为起始顶点,沿当前顶点的边走到未访问过的顶点;当没有未访问过的顶点时,则回到上一个顶点,继续试探别的顶点,直到所有的顶点都被访问过[10]。

3.2 基于 Dijkstra 与深度优先遍历的改进算法

改进的算法是在传统 Dijkstra 算法结合深度优先遍历算法基础上实现的,通过 C#编写,其算法过程如下:

1）创建集合 S 和 U。初始状态时,S 中只存储开始节点 P_0,U 存储除 P_0之外的其他所有节点。声明 2 个数组 d_{dist}和 p_{prev},其中,d_{dist}用来保存开始节点到各个顶点的最短距离,p_{prev}数组用于存储前驱顶点的下标号。

2）从 U 中选取与 P_0相连接的节点 P_i,满足 P_0到 P_i的路径长度小于任何点到 P_0的长度,把 P_i添加到集合 S 中,并从集合 U 中删去 P_i,此时最短路径就是 P_0—P_i。

3）以 P_i为新的起始节点,查找集合 U 中与 P_i连接的节点。若 P_0到 P_i的距离再加上 P_i经过集合 U 中节点 P_{i+1}的距离比从源点 P_0不经过 P_i再到集合 U 中节点 P_{i+1}的距离短,则路径中的 P_i的下一节点为 P_{i+1},把 P_{i+1}加入集合 S 并从 U 中去除。

4）重复以上过程直至所有节点都加入集合 S。最终由这些节点所连成的路径即为最短路径。此时的 d_{dist}数组保存了开始节点到各个顶点的最短距离,p_{prev}数组存储了各顶点的前驱点的下标号。

5）声明栈类型变量 AllVertex,List < ArrayList > 类型变量 AllRoutines 和布尔类型的 Color 数组。利用深度优先遍历算法从源点开始遍历图中每个节点,每遍历到一个节点时,将该节点在 Color 数组中进行标记,表示该点已经访问过,同时将该节点在变量 AllVertex 中进行压栈。通过不断的递归调用,每遍历一个点都进行压栈和标记处理。当遍历到目标节点时,此时栈中保存了一条两点之间的路径,将此路径保存到 AllRoutines 中,然后对当前节点进行出栈处理,从上一个节点继续遍历其他分支,一旦搜索到其他路径就将路径保存在 AllRoutines 中。最后 AllRoutines 中保存了两点之间所有的路径,但这些路径不是最短路径,再通过 4)中 d_{dist}找出最短路径的值跟这些路径的值进行比对,最终找出所有最短路径。

6）筛选出所有最短路径后,需要找出这些路径中哪条路径转弯次数最少。这里主要通过对节点编号的运算进行判别的。假设一条路径上有连续编号 i,j 和 k。对于 j,若 j 满足$[(j\pm1\neq k)\&\&(j\pm1=i)]\|[(j\pm1=k)\&\&(j\pm1\neq i)]$,则记为一次拐弯。

7）通过 count 函数计算路径中的节点数。至此一条路径最短,拐弯次数最少,经过节点数最少的路径被确定,算法结束。

4 实验结果与仿真分析

表 1 为使用传统 Dijkstra 算法在图 1 所示电子地图中选取 8 组路径的仿真结果。

表 2 为使用改进型 Dijkstra 算法在图 1 所示电子地图中选取 8 组路径的仿真结果。

由实验组 2,3,7 和 8 对比可得:改进型的 Dijkstra 算法拐弯次数更少;除组 2 外,改进型的 Dijkstra 算法机器人通过的节点数也是最少的。综上可知,传统 Dijkstra 算法规划出来的路径虽然是最短路径,但是在经过节点数和拐弯次数这 2 个指标上并不是最优的。改进的 Dijkstra 算法搜索的路径不仅最短,而且在转弯次数和经过的节点数(除组 2 外)2 个指标都明显小于传统Dijkstra仿真出来的结果,有效降低了运行损

表 1 传统 Dijkstra 算法路径规划结果

Table 1 Path planning results of traditional Dijkstra algorithm

序号	源点 - 目标点	节点编号	路径总长/mm	节点数量	转弯次数
1	0 - 34	0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 14 - 15 - 16 - 34	20 100	18	1
2	0 - 58	0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 25 - 26 - 27 - 28 - 29 - 30 - 31 - 58	19 300	17	3
3	0 - 104	0 - 17 - 35 - 63 - 64 - 65 - 66 - 67 - 68 - 69 - 70 - 71 - 72 - 73 - 74 - 75 - 76 - 77 - 102 - 103 - 104	17 800	21	3
4	0 - 76	0 - 17 - 35 - 63 - 64 - 65 - 66 - 67 - 68 - 69 - 70 - 71 - 72 - 73 - 74 - 75 - 76	13 900	17	1
5	0 - 129	0 - 17 - 35 - 63 - 64 - 65 - 66 - 67 - 68 - 69 - 70 - 71 - 72 - 73 - 74 - 75 - 76 - 77 - 102 - 128 - 129	18 100	21	3
6	0 - 163	0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 14 - 15 - 16 - 34 - 62 - 90 - 110 - 111 - 112 - 146 - 163	30 600	25	3
7	36 - 103	36 - 35 - 63 - 64 - 65 - 66 - 67 - 68 - 69 - 70 - 71 - 72 - 73 - 74 - 75 - 76 - 77 - 102 - 103	13 400	19	4
8	58 - 20	58 - 57 - 56 - 55 - 54 - 53 - 52 - 51 - 50 - 49 - 48 - 25 - 24 - 23 - 22 - 21 - 20	14 400	17	2

表 2 改进型 Dijkstra 算法路径规划结果

Table 2 Path planning results of improved Dijkstra algorithm

序号	源点 - 目标点	节点编号	路径总长/mm	节点数量	转弯次数
1	0 - 34	0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 14 - 15 - 16 - 34	20 100	18	1
2	0 - 58	0 - 17 - 35 - 36 - 37 - 38 - 39 - 40 - 41 - 42 - 43 - 44 - 45 - 46 - 47 - 48 - 49 - 50 - 51 - 52 - 53 - 54 - 55 - 56 - 57 - 58	19 300	26	1
3	0 - 104	0 - 17 - 35 - 63 - 93 - 94 - 95 - 96 - 97 - 98 - 99 - 100 - 101 - 102 - 103 - 104	17 800	16	1
4	0 - 76	0 - 17 - 35 - 63 - 64 - 65 - 66 - 67 - 68 - 69 - 70 - 71 - 72 - 73 - 74 - 75 - 76	13 900	17	1
5	0 - 129	0 - 17 - 35 - 63 - 93 - 94 - 95 - 96 - 97 - 98 - 99 - 100 - 101 - 102 - 128 - 129	18 100	16	3
6	0 - 163	0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 14 - 15 - 16 - 34 - 62 - 90 - 110 - 111 - 112 - 146 - 163	30 600	25	3
7	36 - 103	36 - 35 - 63 - 93 - 94 - 95 - 96 - 97 - 98 - 99 - 100 - 101 - 102 - 103	13 400	14	2
8	58 - 20	58 - 31 - 30 - 29 - 28 - 27 - 26 - 25 - 24 - 23 - 22 - 21 - 20	14 400	13	1

耗,提高了运行效率。

5 结语

课题组通过对档案库平面结构进行分析,建立了基于拓扑法的电子地图,并建立了以最短路径、最少拐弯次数及最少节点数为目标函数的数学模型。在传统 Dijkstra 算法的基础上,将 Dijkstra 和深度优先遍历算法相结合,对该搜索算法进行了研究设计,可以搜索出一条花费代价最小的路径,对档案室取档机器人路径规划具有一定的参考意义。

参考文献:

[1] 金婷,方欢,方贤文. 改进型 Dijkstra 算法的最短路径求解[J]. 软件导刊,2016,15(2):129 - 131.

[2] 王树西,李安渝. Dijkstra 算法中的多邻接点与多条最短路径问题[J]. 计算机学,2014,41(6):217 - 224.

[3] 孙强,沈建华,顾君忠. Dijkstra 的一种改进算法[J]. 计算机工程与应用,2002,38(3):99 - 101.

[4] 施剑烽,杨勇生. 基于改进的 Dijkstra 算法 AGV 路径规划研究[J]. 科技视界,2016(20):111 - 112.

[5] 刘维民. AGV 路径规划与调度系统研究[D]. 广州:华南理工大学,2016:7.

[6] 毛洋洋,赵欢,韩世博,等. 面向复杂曲面的机器人砂带磨抛路径规划及后处理研究[J]. 机电工程,2017,34(8):829 - 834.

[7] 郭丽晓. 基于拓扑地图的 AGV 智能路径规划技术研究[D]. 杭州:浙江大学,2013:18.

[8] 王玉林,魏国亮,鲍海锋. 基于 Dijkstra 算法的磁带导引 AGV 路径规划[J]. 农业装备与车辆工程,2018,56(3):51 - 54.

[9] 张默. Dijkstra 最短路径算法的研究[J]. 数学学习与研究,2018(16):152.

[10] 严蔚敏,吴伟民. 数据结构[M]. 2 版. 北京:清华大学出版社,1992:167 - 168.