计算机工程与应用
Computer Engineering and Applications
ISSN 1002-8331,CN 11-2127/TP

《计算机工程与应用》网络首发论文

题目：双向搜索机制的改进 A*算法研究
作者：孔继利，张鹏坤，刘晓平
网络首发日期：2020-10-16
引用格式：孔继利，张鹏坤，刘晓平．双向搜索机制的改进 A*算法研究．计算机工程与应用. https://kns.cnki.net/kcms/detail/11.2127.TP.20201016.1327.008.html

双向搜索机制的改进 A*算法研究

孔继利，张鹏坤，刘晓平

北京邮电大学 现代邮政学院，北京 100876

摘　要：针对大规模环境下传统 A* 算法路径寻优存在的内存占有率高、计算效率低下的问题，提出了一种改进 A*算法。首先引入了双向搜索机制，以原始起点、终点和对向搜索所处的当前节点作为目标点进行搜索操作，使 AGV 的路径寻优具备更加合理的方向性；优化评价函数，改进了评价函数的传统计算方式，通过测试为评价函数选择了合适的权重系数，减少路径寻优过程中的冗余点，提升路径寻优的计算效率，节约内存占有率。为了验证改进 A*算法的有效性，本文在 Matlab 平台中进行编程，在不同尺寸的含障碍栅格地图中进行了仿真。仿真结果表明：改进 A*算法在路径寻优过程中所遍历的节点数量较少，搜索过程中的计算效率更高，并且可获得到达目标点的最短路径。

关键词：双向搜索；改进 A* 算法；路径寻优

文献标志码:A　**中图分类号:**TP242.6　doi：10.3778/j.issn.1002-8331.2002-0016

孔继利，张鹏坤，刘晓平. 双向搜索机制的改进 A*算法研究.计算机工程与应用

KONG Jili, ZHANG Pengkun, LIU Xiaoping. Research on improved A* algorithm of bidirectional search mechanism. Computer Engineering and Applications

Research on improved A* algorithm of bidirectional search mechanism

KONG Jili, ZHANG Pengkun, LIU Xiaoping

School of Modern Post, Beijing University of Post and Telecommunications, Beijing 100876, China

Absrtact: Aiming at the problems of high memory occupancy and low computing efficiency of A* algorithm in large-scale environment, an improved A* algorithm is proposed. Firstly, a bidirectional search mechanism is introduced. Search with the original starting point, the end point and the opposite current point as the target point, so that the AGV path optimization has directionality. Secondly, the evaluation function is optimized. Select the appropriate weight coefficient for the evaluation function, so as to improve the calculation efficiency of path optimization. In order to verify the effectiveness of the improved A* algorithm, it is programmed in Matlab platform and simulated in different sizes of grid map with obstacles. The simulation results show that the number of nodes traversed by the improved A* algorithm is less, the calculation efficiency is higher and the shortest path can be obtained.

Key words: bidirectional search; improved A* algorithm; path optimization

1　引言

随着经济日益发展以及生产管理水平的进步，高度自动化、智能化的 AGV 应用愈加广泛，柔性制造系统[1]、智慧仓储[2]、自动化码头[3]、智能停车场[4]都是 AGV 常见的应用场景。对 AGV 调度而言，最重要的内容之

基金项目：国家重点研发计划资助项目(No.2018YFB1403100)；教育部人文社会科学研究青年基金项目(No. 20YJC630054)。

作者简介：孔继利(1982—)，男,博士,副教授,研究领域为智能物流系统规划与设计,智能调度与路径规划;张鹏坤(1996—)，通信作者,男,硕士研究生,研究领域为物流系统分析与仿真,E-mail:zpk707@126.com;刘晓平(1965—)，男,博士,教授，研究领域为工业机器人及智能物流技术装备,物流大数据。

一就是路径规划。AGV 的路径规划是指选取从任务起始点到目标点的一条路线，使一定目标（时间、距离、能耗等）达到最优化，并且避免与已知障碍物的碰撞。为 AGV 选择有效的路径，可以提高物流效率，降低运输成本。因此，对 AGV 路径规划算法进行研究具有重要意义。

在点点间运输问题的路径规划中，常见的算法有 Dijkstra 算法[5]、Floyd 算法[6]、人工势场法等。A*算法同样作为常见的点点间路径规划算法[7]，与 Dijkstra 算法和 Floyd 算法相比具有更强的启发式信息，在最短路径的搜索效率上存在一定优势。A*算法是一种基于全局的最短路径搜索算法，能够较好地避免人工势场法频繁出现的局部最优问题。A*算法已经广泛应用于多种场景，包括室内机器人的路径规划、无人船的路径规划和电子游戏中的无碰撞检测等。但 A*算法在实际应用中存在着遍历节点多、搜索过程中计算量庞大等问题，在大规模环境下不断调用 A*算法会占据大量内存资源。因此，传统 A*算法以及改进 A*算法在计算效率上依旧有提升空间。

国内外已有不少学者针对 A*算法的局限性，对 A*算法进行改进。Harabor 和 Grastien[8]提出了跳点算法，其 OPEN 列表中只储存有代表性的跳点，通过对跳点的连接可以实现长距离的跳跃，提高计算效率；但当地图中障碍物较多时，会找到过多的跳点，降低算法求解效率。王小红和叶涛[9]提出了一种多邻域搜索的改进 A*算法，在小地图中效果较为显著，但当地图规模较大时计算量过大，导致搜索效率降低。Wang 和 Xiang[10]提出了一种改进 A*算法，生成的路径比传统 A*算法更优，但计算效率并未得到较高的提升。Lin 等[11]分析了当前节点的父节点对启发式函数的影响，并利用最优权值来优化路径，提出了一种提高计算效率的改进 A*算法，但以牺牲最优路径为代价。吴鹏等[12]提出了一种双向搜索 A*算法，正向、反向搜索过程分别以对方所扩展的最优节点作为自己的目标节点，该算法可大幅缩短路径寻优时间，但同样会以牺牲最优路径为代价。王中玉等[13]对评价函数进行了改进，并且提出了减少路径冗余点的方法，通过仿真证明了改进算法的平滑性更强，但该方法的计算效率仍旧有提升空间。

本文为了进一步提升路径寻优的计算效率与保证路径的最优化，设计了一种基于双向搜索机制的改进 A*算法，从搜索方式与评价函数两方面对传统 A*算法进行优化。该算法可应用于仓储系统等已知障碍物的场景。

2 环境模型的建立

在 AGV 的路径规划中，首先需要建立合适的地图模型。常用的地图表示方法有栅格法[14]、拓扑地图法[15]等。栅格法构建地图模型可以直接有效地进行环境信息的描述，并且在编码过程中易于实现。因此，栅格法在 AGV 的路径规划中得到了广泛应用。

本文利用栅格法进行地图模型的构建，并且根据实际情况将栅格分成两种：白色和黑色；白色代表可通行状态，黑色代表障碍状态，如图 1 所示。

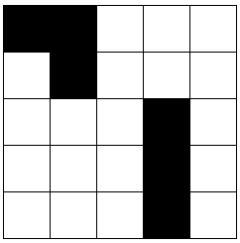


图 1　栅格地图

Fig.1　Grid map

根据应用场景的实际需求，本文在 AGV 运行过程中做出如下假设：

(1) 假设 AGV 在水平方向与垂直方向的正常直线行走中不会与旁边栅格中的障碍物相撞，且只能在可通行状态下的栅格中运行；

(2) 在不存在边界和障碍物的情况下，通常 AGV 在栅格中的行进为一步行走。由于本文考虑的是以室内

仓储空间为应用场景，在实际的应用中 AGV 小车在某些方向（如栅格中心连线与横轴呈 45°的方向）经过障碍物时可能会与障碍物发生碰撞，造成损失，如图 2 所示。故本文假设 AGV 只能在当前节点周围的 4 个方向上遍历并依据选择标准选择节点，并且不考虑 AGV 自身高度的影响。AGV 可能的运行方向如图 3 所示。

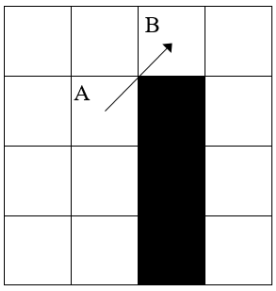


图 2　AGV 在某些方向可能发生碰撞示意图

Fig.2　Schematic diagram of AGV collision in some directions

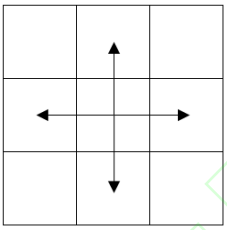


图 3　AGV 可能的运行方向

Fig.3　The possible direction of the AGV

(3) 在 AGV 运动过程中，周围环境保持不变。

3　基于双向搜索机制的改进 A*算法设计

3.1　改进 A*算法的描述

传统 A*算法为单向搜索，遍历的节点较多，计算效率较低。本文提出的改进 A*算法为双向搜索机制，通过正反两个方向的交替搜索减少计算过程中的冗余节点，并通过设置合理的评估函数进一步提升搜索效率。

文献[12]所提出的改进 A*算法同样为双向搜索机制，故选择文献[12]中的算法作为对比算法。本文分别设置正向搜索的 $OPEN_1$ 列表、$CLOSE_1$ 列表、评估函数，反向搜索的 $OPEN_2$ 列表、$CLOSE_2$ 列表、评估函数。n_1、n_2 分别作为正向搜索和反向搜索过程中的当前节点。正向搜索过程在搜索任务终点 e 的同时考虑反向搜索的当前点 n_2，在启发式函数中对 n_1 相距 e 的距离和 n_1 相距 n_2 的距离进行加权处理，共同引导正向搜索方向；反向搜索过程在搜索任务起点 s 的同时考虑正向搜索的当前点 n_1，在启发式函数中对 n_2 相距 s 的距离和 n_2 相距 n_1 的距离进行加权处理，共同引导反向搜索方向。这样的双向搜索方式可以较好的避免启发式信息过强导致牺牲路径长度作为代价的问题，并且能够保证在路径寻优过程中尽可能少地遍历无关节点，提高算法的计算效率。

3.2　改进 A*算法的处理流程

当改进 A*算法开始运行，首先需要初始化 $OPEN_1$ 列表、$CLOSE_1$ 列表和 $OPEN_2$ 列表、$CLOSE_2$ 列表。搜索过程中正向搜索与反向搜索交替进行。

改进 A*算法处理流程如下：

(1) 将起始点 s 设置为正向当前节点 n_1，并加入 $OPEN_1$ 列表。

(2) 搜索 n_1 点周围可到达的节点，把它们加入 $OPEN_1$ 列表。将起始点 s 设置为新加入节点的父节点。

(3) 将起始点 s 从 $OPEN_1$ 列表中删除，加入 $CLOSE_1$ 列表中。

(4) 将目标点 e 设置为反向当前节点 n_2，并加入 $OPEN_2$ 列表。

(5) 搜索 n_2 点周围可到达的节点，把它们加入 $OPEN_2$ 列表。将目标点 e 设置为新加入节点的父节点。

(6) 将目标点 e 从 $OPEN_2$ 列表中删除，加入 $CLOSE_2$ 列表中。

(7) 重复如下步骤：

①遍历 $OPEN_1$ 列表，找到其中评估函数 $f_1(x)$ 值最小的节点，将其设置为正向当前节点 n_1。

②将 n_1 点从 $OPEN_1$ 列表中删除，加入 $CLOSE_1$ 列表中。

③检查 n_1 点周围可到达的节点，并忽略其中已在 $CLOSE_1$ 列表的节点或障碍物。如果可到达的节点不在 $OPEN_1$ 列表中，则将其加入到 $OPEN_1$ 列表，并且将 n_1 点设定为它们的父节点。如果可到达的节点存在于 $OPEN_1$ 列表中，称该节点为 x 点，计算经过 n_1 点到达 x 点的 $g_1(x)$ 值，如果该 $g_1(x)$ 值小于原 $g_1(x)$ 值，则将 n_1 点作为 x 点的父节点，更新 $OPEN_1$ 列表中的 $f_1(x)$ 和 $g_1(x)$。

④遍历 $OPEN_2$ 列表，找到其中评估函数 $f_2(y)$ 值最小的节点，将其设置为反向当前节点 n_2。

⑤将 n_2 点从 $OPEN_2$ 列表中删除，加入 $CLOSE_2$ 列表中。

⑥检查 n_2 点周围可到达的节点，并忽略其中已在 $CLOSE_2$ 列表的节点或障碍物。如果可到达的节点不在 $OPEN_2$ 列表中，则将其加入到 $OPEN_2$ 列表。并且将 n_2 点设定为它们的父节点。如果可到达的节点存在于 $OPEN_2$ 列表中，称该节点为 y 点，计算经过 n_2 点到达 y 点的 $g_2(y)$ 值，如果该 $g_2(y)$ 值小于原 $g_2(y)$ 值，则将 n_2 点作为 y 点的父节点，更新 $OPEN_2$ 列表中的 $f_2(y)$ 和 $g_2(y)$。

(8) 终止条件为：

①当 $OPEN_1$ 列表中包含 $CLOSE_2$ 列表中的点或 $OPEN_2$ 列表中包含 $CLOSE_1$ 列表中的点，此时路径已找到。

②当 $OPEN_1$ 列表或 $OPEN_2$ 列表任何一个为空时，此时寻路失败，输出寻路失败提示。

改进 A*算法的具体流程图如图 4 所示。

改进 A*算法的伪代码如下：

```
insert start node in OPEN1
insert target node in OPEN2
//将两个 OPEN 列表进行初始化操作。
While OPEN1≠null & OPEN2≠null
    computen1 child nodex
    update OPEN1
    //遍历 n1 的子节点并更新 OPEN1 列表。
    if x is in the CLOSE2
        break
    //当 n1 的子节点在 CLOSE2 列表时，则寻找到了一条最短路径，跳出循环。
    endif
    find min f1(x) in OPEN1
    n1= x
    insert n1 inCLOSE1
    //将 OPEN1 列表中 f1(x)最小值的点设置为 n1 并更新 CLOSE1 列表
    computen2 child nodey
    update OPEN2
    //遍历 n2 的子节点并更新 OPEN2 列表。
    if y is in the CLOSE1
        break
    //当 n2 的子节点在 CLOSE1 列表时，则寻找到了一条最短路径，跳出循环。
    endif
    find min f2(y) in OPEN2
    n2= y
    insert n2inCLOSE2
    //将 OPEN2 列表中 f2(y)最小值的点设置为 n2 并更新 CLOSE2 列表
Get path/Fail to get path
```

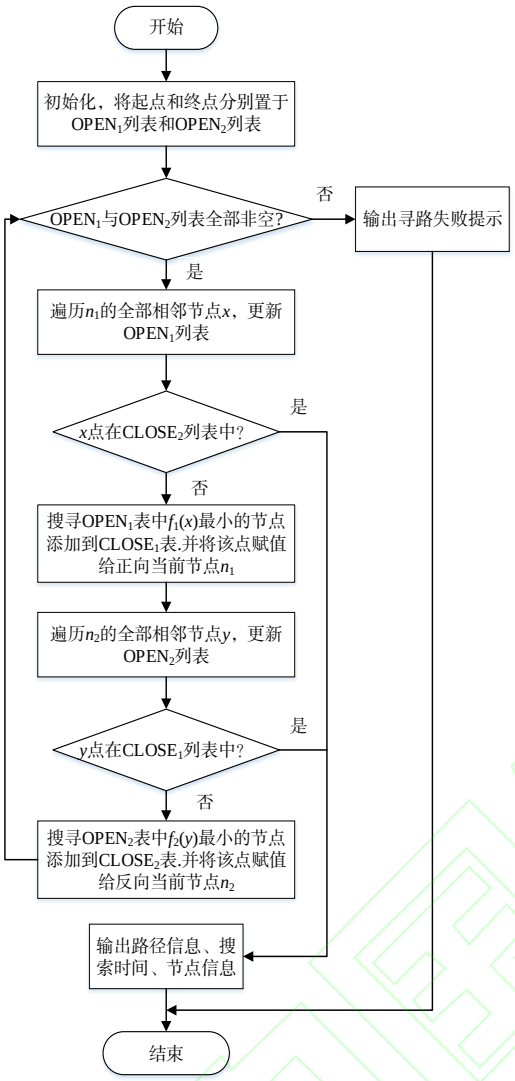


图 4　改进 A*算法具体流程图

Fig.4　Specific flow chart of the improved A* algorithm

3.3　改进 A*算法的评估函数

3.3.1　评估函数的设计

本文从寻找合适的启发式函数和加权处理两方面对传统 A*算法的评估函数进行改进，平衡启发式信息的强度，提高算法搜索效率。

(1) 正向搜索

正向搜索的评估函数如式 1。

$$f_1(n_1) = (1-b) \times \left(\frac{h_1(n_1)}{1+a} + \frac{a \times h_1'(n_1)}{1+a} \right) + b \times g_1(n_1) \quad (1)$$

其中：n_1 为正向搜索中的当前点。

$f_1(n_1)$ 为正向搜索的评估函数。

$h_1(n_1)$ 为正向搜索中以任务终点 e 为目标点的启发式函数，采用欧式距离。

$h_1'(n_1)$ 为正向搜索中以反向搜索当前点 n_2 为目标点的启发式函数，采用欧式距离。

a 为启发式函数中 $h_1(n_1)$ 与 $h_1'(n_1)$ 之间的权重系数，取值范围$[0,+\infty)$。

b 为历史代价函数的权重系数，取值范围为(0,1)。

$(1-b)$ 为启发式函数的权重系数。

$g_1(n_1)$ 表示正向搜索历史代价函数，其计算方式如下：

$$g_1(n_1) = g_1(n_1\text{-}1) + g_1'(n_1) \quad (2)$$

$g_1(n_1\text{-}1)$ 为 n_1 的父节点的历史代价。

$g_1'(n_1)$ 为父节点到 n_1 的成本代价，采用欧式距离。

(2) 反向搜索

反向搜索的评估函数如式 3。

$$f_2(n_2) = (1\text{-}b) \times \left(\frac{h_2(n_2)}{1+a} + \frac{a \times h_2'(n_2)}{1+a} \right) + b \times g_2(n_2) \quad (3)$$

其中：n_2 为反向搜索中的当前点。

$f_2(n_2)$ 为反向搜索的评估函数。

$h_2(n_2)$ 为反向搜索中以任务起点 s 为目标点的启发式函数，采用欧式距离。

$h_2'(n_2)$ 为反向搜索中以正向搜索当前点 n_1 为目标点的启发式函数，采用欧式距离。

$g_2(n_2)$ 表示反向搜索历史代价函数,其计算方式如下:

$$g_2(n_2) = g_2(n_2\text{-}1) + g_2'(n_2) \quad (4)$$

$g_2(n_2\text{-}1)$ 为 n_2 的父节点的历史代价。

$g_2'(n_2)$ 为父节点到 n_2 的成本代价，采用欧式距离。

3.3.2 评估函数参数值的确定

在参数 a 和参数 b 的确定过程中，本文选择尺寸为 50×50 的栅格地图进行测试。每组参数测试 5 次，记录该组参数的路径长度、遍历节点数量与平均路径寻优时间。实验发现：当参数 b 超过 0.5 时，算法遍历的节点数量与路径寻优时间将会明显上升，这主要是由于历史代价函数 $g_1(n_1)$ 与 $g_2(n_2)$ 的权重过高，使算法更倾向于于 Dijkstra 算法，而启发式信息较弱，所以路径寻优时间较长。遍历节点数量与路径寻优时间如图 5、图 6 所示(当参数 b 取值为 0.1、0.2、0.3 时，遍历点的数量相同，路径寻优时间高度相似，故在图 4、图 5 中的折线重叠，原本的 7 个类型在折线图中显示为 5 个)。

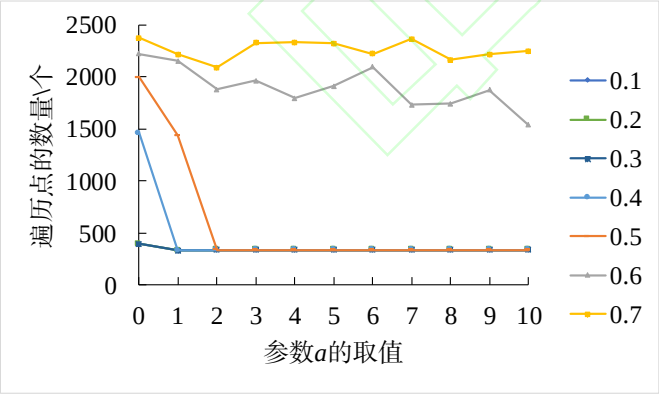


图 5　不同参数下遍历点的数量

Fig.5　The number of the traversal nodes with different parameters

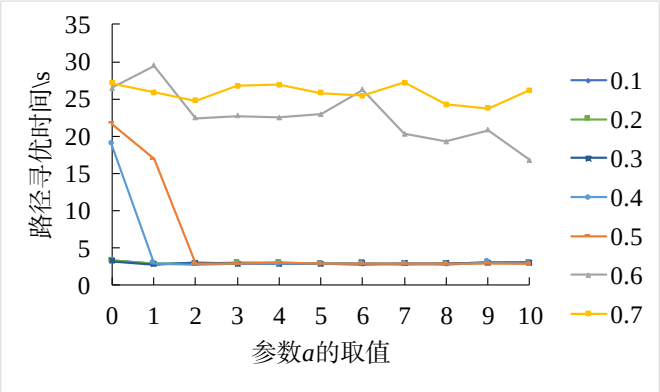


图 6　不同参数下的路径寻优时间

Fig.6　The time of the path optimization with different parameters

剔除掉初次实验中不好的参数组合后，为防止出现启发式函数权重过高而导致寻找到的路径不是最短路径的问题，更换测试的起点和终点进行二次测试，路径寻优的路径长度如图 7 所示。

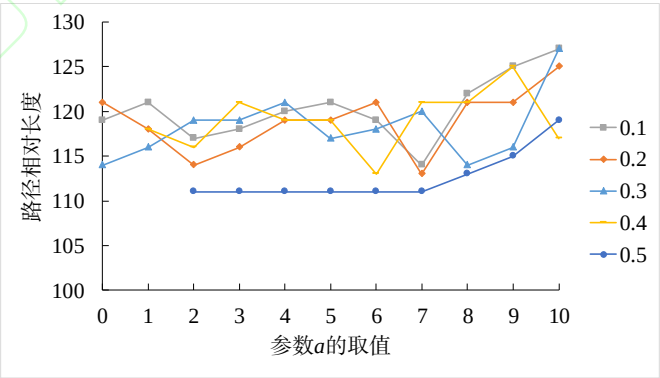


图 7　不同参数下的路径长度

Fig.7　The length of the path with different parameters

由图 6 可知：当参数 b 取 0.5 且参数 a 在[2,7]区间内时，可以保证找到最短路径，在 $a \in [2,7]$ 区间时的路径寻优时间与遍历节点数量如表 1 所示。

表 1　不同 a 值对应路径寻优时间和遍历节点数量

Table 1　The path optimization time and traversal nodes number corresponding to different a values

参数 a	2	3	4	5	6	7
路径寻优时间/s	21.09	20.86	19.31	20.89	21.22	20.83
遍历节点数量/个	1891	1926	1792	1930	1917	1809

由表 1 可知：当参数 a 取 4 时，遍历节点数量为 1792，寻路时间为 19.31s，为最优结果。经过测试，当参数 a 取 4，参数 b 取 0.5 时在其他地图中的路径寻优结果同样较为优异，故本文最终确定参数 a 取 4，参数 b 取 0.5。正向搜索的最终评估函数如式 5 所示，反向搜索的最终评估函数如式 6 所示。

$$f_1(n_1) = 0.5 \times \left(\frac{h_1(n_1)}{5} + \frac{4 \times h_1'(n_1)}{5} \right) + 0.5 \times g_1(n_1) \quad (5)$$

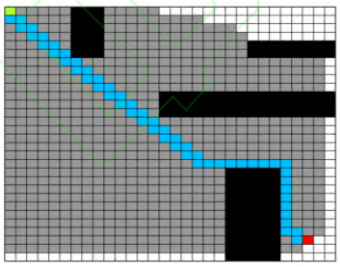
$$f_2(n_2) = 0.5 \times \left(\frac{h_2(n_2)}{5} + \frac{4 \times h_2'(n_2)}{5} \right) + 0.5 \times g_2(n_2) \quad (6)$$

4　仿真分析

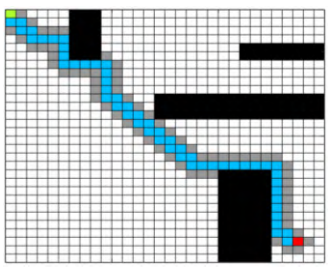
为了验证本文提出的基于双向搜索机制的改进 A*算法的求解效率和效果，现将其在 MATLAB2018b 实验平台下进行编程，在不同的栅格地图下进行仿真实验，并与传统 A*算法、文献[12]中的改进 A*算法进行比较。计算机配置为：Windows 10 操作系统，处理器为 i7-8565U，主频 1.8GHz，运行内存为 16G。

本文构建了不同尺寸的包含已知障碍物的栅格地图。在地图中，黑色栅格代表障碍物，白色栅格为无障碍的可行区间，蓝色栅格为搜索到的可行路径，灰色栅格为已经参与搜索计算的栅格，绿色栅格代表任务起点，红色栅格代表任务终点。

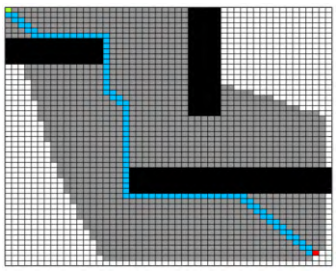
本文算法与传统 A*算法的部分仿真结果如图 8 所示。



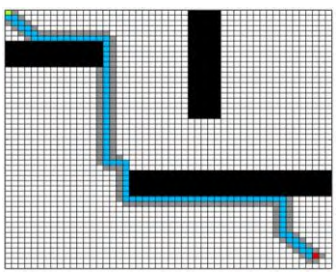
(a)30×30 栅格地图下传统 A*算法求解结果



(b)30×30 栅格地图下本文改进 A*算法求解结果



(c)50×50 栅格地图下传统 A*算法求解结果



(d)50×50 栅格地图下本文改进 A*算法求解结果

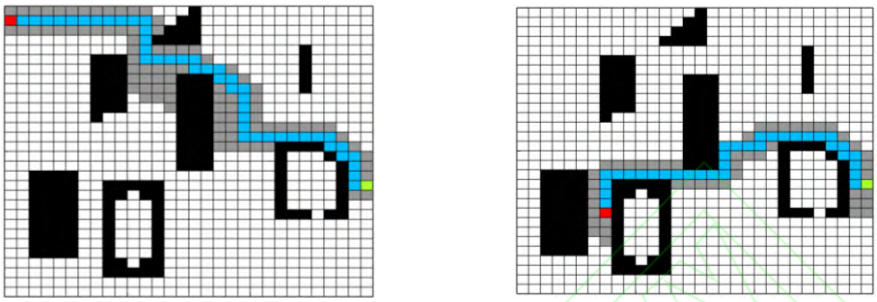
图 8　路径规划部分仿真结果(1)

Fig.8　Partial simulation results of path planning(1)

由图 8 可知：本文提出的改进 A*算法在不同尺寸的含已知障碍物的栅格地图中所遍历节点明显减少，搜

索效率明显提高，可节约大量的计算资源。

对文献[12]中的算法与地图进行编码仿真，设置与文献[12]相同的起始点与目标点,测试结果如图 9 所示。



(a)目标点 1 搜索到的路径　　(b) 目标点 2 搜索到的路径

图 9　编码文献[12]算法的仿真结果

Fig.9　The simulation results of the coding literature [12] algorithm

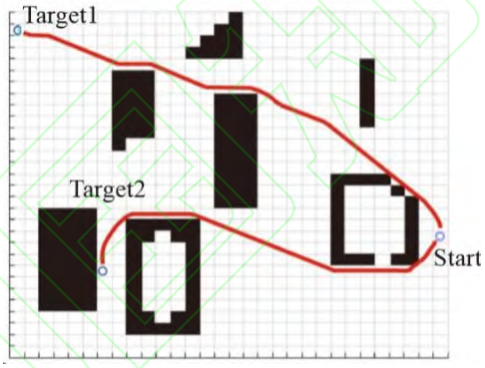


图 10　文献[12]的仿真结果

Fig.10　The simulation results of the literature [12] algorithm

图 10 为文献[12]搜索到的路径，与图 9 中所搜到的路径相似。两个算法求解结果的主要区别为搜索目标点 2 时的路径绕行障碍物的方向不同,但均可搜索到最短路径。这主要是由于文献[12]中采取的节点扩展方向为 8 方向，而本文在编码时采取的节点扩展方向为 4 方向。

在不同地图环境下对传统 A*算法、文献[12]中的改进 A*算法、Dijkstra 算法、蚁群算法、本文的改进 A*算法进行对比。表 1 与表 2 给出了三种算法在不同栅格地图中的搜索时间、扩展节点数量和路径长度的对比；其中，搜索时间之比、扩展节点之比是表格中较优异的求解结果与本文改进 A*算法的相应求解结果的比值。

因为两条路径有重叠部分，为防止混淆，将两条路径放在两张图中，图 9(a)为图 10 中目标点 1 的情况，图 9(b)为图 10 中目标点 2 的情况。

表 2　不同算法的仿真结果对比(1)

Table 2　Comparison of simulation results of different algorithms (1)

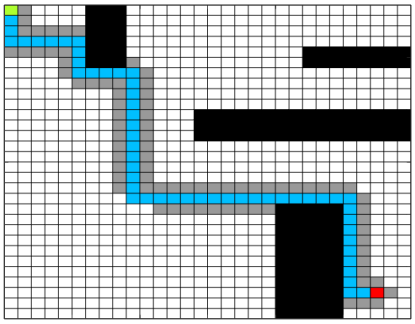
地图大小	搜索时间/s			扩展节点数量/个			相对路径长度			搜索时间之比	扩展节点之比
	A*算法	文献[12]中的改进A*算法	本文的改进A*算法	A*算法	文献[12]中的改进A*算法	本文的改进A*算法	A*算法	文献[12]中的改进A*算法	本文的改进A*算法		
30×30	8.41	2.38	2.12	404	134	118	54	54	54	1.12	1.14
50×50	39.03	2.83	2.72	1475	224	216	94	94	94	1.04	1.04
75×75	76.55	3.95	3.85	2918	355	330	142	142	142	1.03	1.08
100×100	179.01	6.07	5.89	6711	477	432	194	194	194	1.03	1.10

表 3　不同算法的仿真结果对比(2)

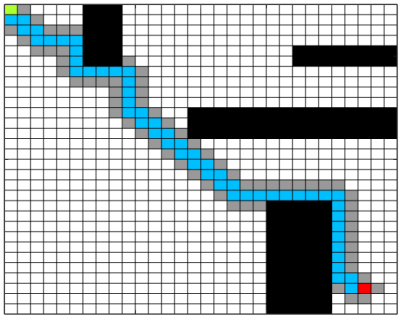
Table 3　Comparison of simulation results of different algorithms (2)

地图大小	搜索时间/s			扩展节点数量/个			相对路径长度			搜索时间之比	扩展节点之比
	Dijkstra算法	蚁群算法	本文的改进A*算法	Dijkstra算法	蚁群算法	本文的改进A*算法	Dijkstra算法	蚁群算法	本文的改进A*算法		
30×30	19.71	9.23	2.12	754	414	118	54	62	54	4.35	3.51
50×50	70.04	41.54	2.72	2348	1211	216	94	108	94	15.27	5.61
75×75	151.22	70.15	3.85	5372	2752	330	142	170	142	18.22	8.34
100×100	307.84	168.46	5.89	9247	6452	432	194	241	194	2836	14.93

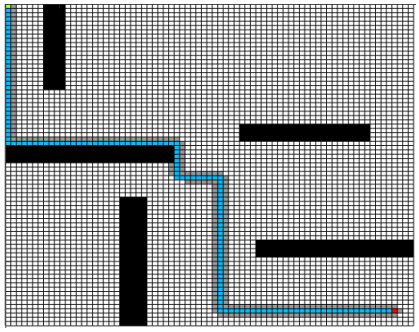
由表 2 和表 3 的计算结果可知：本文的改进 A*算法与文献[12]中的改进A*算法相较于传统的Dijkstra算法、蚁群算法和 A*算法在搜索时间与扩展节点上具有较大优势，并且随着栅格地图的尺寸规模增大，搜索效率的提升效果更加明显；与文献[12]中的改进 A*算法求解结果相比，本文设计的改进 A*算法在四种规模的栅格地图中可获得相同的路径长度，但搜索时间更短、扩展节点数量更少，在效率上具有明显优势。部分仿真结果如图 11 所示。



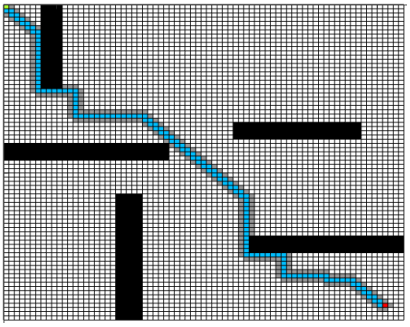
(a)30×30 栅格地图下文献[12]中的算法求解结果



(b)30×30 栅格地图下本文改进 A*算法求解结果



(c)75×75 栅格地图下文献[12]中的算法求解结果



(d)75×75 栅格地图下本文改进 A*算法求解结果

图 11　路径规划部分仿真结果(2)

Fig.11　Partial simulation results of path planning(2)

更换起始点与目标点，重新通过不同算法进行路径规划，表 4 和表 5 给出了不同算法在不同栅格地图中的搜索时间、扩展节点数量和路径长度的对比。

表 4　不同算法的仿真结果对比(3)

Table 4　Comparison of simulation results of different algorithms (3)

地图大小	搜索时间/s			扩展节点数量/个			相对路径长度			搜索时间之比	扩展节点之比
	A*算法	文献[12]中的改进A*算法	本文的改进A*算法	A*算法	文献[12]中的改进A*算法	本文的改进A*算法	A*算法	文献[12]中的改进A*算法	本文的改进A*算法		
30×30	12.01	7.31	7.02	520	353	308	61	67	61	1.05	1.15
50×50	40.01	24.13	23.23	1494	972	938	111	111	111	1.04	1.04
75×75	71.27	28.23	43.21	2843	1444	1899	132	140	132	0.65	0.76
100×100	3.15	3.15	3.15	199	199	199	100	100	100	1.00	1.00

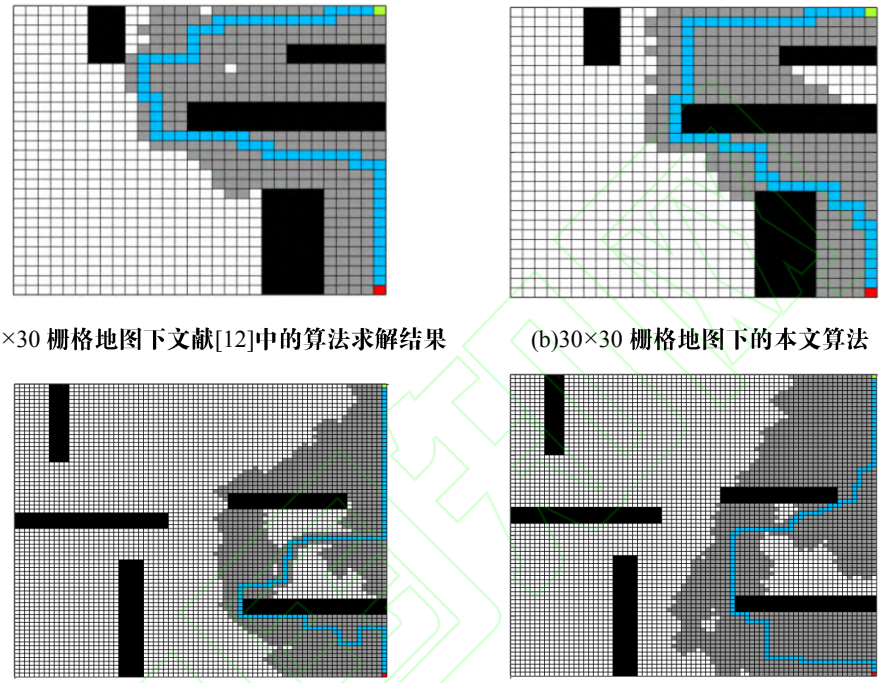
表 5　不同算法的仿真结果对比(4)

Table 5　Comparison of simulation results of different algorithms (4)

地图大小	搜索时间/s			扩展节点数量/个			相对路径长度			搜索时间之比	扩展节点之比
	Dijkstra算法	蚁群算法	本文的改进A*算法	Dijkstra算法	蚁群算法	本文的改进A*算法	Dijkstra算法	蚁群算法	本文的改进A*算法		
30×30	15.20	14.31	7.02	738	582	308	61	72	61	2.04	1.89
50×50	64.08	45.27	23.23	2024	1693	938	111	136	111	1.95	1.80
75×75	112.17	82.40	43.21	3746	3042	1899	132	181	132	1.91	1.60
100×100	5.48	4.11	3.15	422	252	199	100	100	100	1.30	1.27

由表 4 可以看出在 100×100 栅格地图下，文献[12]中的改进 A*算法与本文的改进 A*算法的搜索时间与扩展节点数量相当，这主要是因为在节点搜索的过程中没有接触到障碍物，形成了简易的直达路径。在 30×30 与 50×50 栅格地图下，本文的改进 A*算法在搜索时间与扩展节点数量上具有一定的优势，搜索效率较高，且

在 30×30 栅格地图下本文的改进 A*算法搜索的路径长度相较于文献[12]中的改进 A*算法具有一定优势。在 75×75 栅格地图下，虽然本文的改进 A*算法相较于文献[12]中的改进 A*算法的搜索时间与扩展节点数量存在一定劣势，但可求得最短路径，这主要是因为本文中的改进 A*算法的启发式信息弱于文献[12]中的改进 A*算法，导致寻路时间较长，但保证了最短路径。部分仿真结果如图 12 所示。



(a)30×30 栅格地图下文献[12]中的算法求解结果　(b)30×30 栅格地图下的本文算法

(c)75×75 栅格地图下文献[12]中的算法求解结果　(d)75×75 栅格地图下的本文算法

图 12　路径规划部分仿真结果(3)

Fig.12　Partial simulation results of path planning(3)

5　结束语

由于传统 A*算法在路径规划过程中所遍历的点过多，在其搜索过程中存在计算量庞大、内存占用严重等缺点，无法满足 AGV 小车在规模较大的仓储系统内路径规划的实时性要求。为了提高路径规划的效率，本文提出了一种双向搜索机制的改进 A*算法。经过在 Matlab 平台中的不同尺寸栅格地图进行仿真实验，并与传统 A*算法和文献[12]中的改进 A*算法进行对比，证明了本文提出的改进 A*算法可生成最短路径的同时可以明显提升寻路速度，尤其是在规模较大的场景下效果更加明显。

虽然本文所提出的改进 A*算法相较于传统 A*算法与文献[12]中的改进 A*算法在效率上有了较大提升，但并未考虑场景的动态变化和由于 AGV 发生转向所花费的时间成本、电量成本等。在未来的研究中可将这些因素作为改进算法的依据，提高 AGV 路径规划的应用价值。

参考文献:

[1] Shi Y, Wang X, Sun X , et al. A two-phase strategy with micro genetic algorithm for scheduling Multiple AGVs[C]//2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Budapest: IEEE, 2016:3101-3106.

[2] Draganjac I, Miklic D, Kovacic Z, et al. Decentralized Control of Multi-AGV Systems in Autonomous Warehousing Applications[J]. IEEE Transactions on Automation Science and Engineering, 2016,13(4):1433-1447.

[3] 张素云, 杨勇生, 梁承姬, 等. 自动化码头多 AGV 路径冲突的优化控制研究[J].交通运输系统工程与信息，2017, 17(2): 83-89.
S. Zhang, Y. Yang, C. Liang, et al. Optimal Control of Multiple AGV Path Conflict in Automated Terminals[J]. Journal of Transportation Systems Engineering and Information Technology, 2017, 17(2):83-89.

[4] X. Sun, Y. Zhao, S. Shen, et al. Scheduling Multiple AGVs with Dynamic Time-windows for Smart Indoor Parking Lot[C]//2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD), Nanjing: IEEE, 2018:864-868.

[5] Dijkstra EW.A note on two problems in connexion with graphs [J]. Numerische Mathematik, 1959, 1(1): 269-271.

[6] Floyd,Robert W. Algorithm 97: Shortest path[J]. Communi-cations of the ACM,1962, 5(6):344-348.

[7] Hart P E, Nilsson N J ,Raphael B. A formal basis for the heuristic determination of minimum cost paths[J]. IEEE Transactions on Systems Science and Cybernetics, 1968, 4(2): 100-107.

[8] HaraborD, GrastienA. The JPS path finding system[C]//5th Annual Symposium on Combinatorial Search. Menlo Park, USA: AAAI, 2012: 207-208.

[9] 王小红, 叶涛. 基于改进 A*算法机器人路径规划研究[J]. 计算机测量与控制, 2018,26(7):282-286.
X. Wang, T. Ye. Researchon Robot Path Planning Based on Improved A* Algorithm[J].Computer Measurement & Control, 2018, 26(7): 282-286.

[10] Wang Z, Xiang X. Improved Astar algorithm for path planning of marine robot[C]//37th Chinese Control Conference (CCC), Wuhan: IEEE, 2018:5410-5414.

[11] Lin M, Yuan K, Shi C, et al. Path planning of mobile robot based on improved A* algorithm[C]//29th Chinese Control And Decision Conference (CCDC), Chongqing: IEEE, 2017: 3570-3576.

[12] 吴鹏,桑成军,陆忠华,等.基于改进 A*算法的移动机器人路径规划研究[J].计算机工程与应用,2019,55(21)：227-233.
P. Wu, C. Sang, Z. Lu, et al. Research on Mobile Robot Path Planning Based on Improved A* Algorithm[J]. Computer Engineering and Application,2019,55(21):227-233.

[13] 王中玉, 曾国辉, 黄勃, 等. 改进 A*算法的机器人全局最优路径规划[J].计算机应用, 2019,39(9)：2517-2522.
Z. Wang, G. Zeng, B. Huang, et al.Global optimal path planning for robots with improved A* algorithm[J]. Journal of Computer Application, 2019,39(9):2517-2522.

[14] 刘一松, 魏宁, 孙亚民. 基于栅格法的虚拟人快速路径规划[J]. 计算机工程与设计,2008, 29(5):1229-1230.
Y. Liu, N. Wei, Y. Sun. Path Planning Algorithm Based on Grid Method for Virtual Human[J]. Computer Engineering and Design,2008, 29(5):1229-1230.

[15] 郝树运. AGV 智能停车库路径规划与布局优化研究[D]. 北京: 北京交通大学,2019.
S. Hao. ResearchonPath Planning and Layout Optimization of AGV Intelligent Garage[D]. Beijing: Beijing Jiaotong University, 2019.