计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190700138

基于改进 Dijkstra 算法的 AGVs 无碰撞路径规划

姜辰凯[1]　李　智[1,2]　盘书宝[2]　王勇军[2]
1 桂林电子科技大学电子工程与自动化学院　广西 桂林 541004
2 桂林航天工业学院电子信息与自动化学院　广西 桂林 541004
(1084710601@qq.com)

摘　要　针对多自动导引车(Automatic Guided Vehicle,AGV)在柔性制造系统中出现的路径规划与冲突问题,提出了一种基于时间窗的改进 Dijkstra 算法,实现多 AGV 的动态路径规划。首先,利用传统 Dijkstra 算法为执行调度任务的多 AGV 规划路径,并统计被规划路径的使用程度,计算加权系数,然后将加权后的路径长度更新到数据库中;其次,计算 AGV 通过每个工位节点的时间,通过时间窗的排布避免碰撞冲突;最后,当产生冲突时,通过计算并设置 AGV 的优先级,对优先级较低的 AGV 重新进行路径规划。仿真实验结果表明,该算法能够在最优路径下有效避免冲突与死锁,不仅提高了系统效率,而且使系统具有较好的鲁棒性。

关键词:自动导引车;路径规划;改进 Dijkstra 算法;时间窗;无碰撞冲突
中图法分类号　TP242

Collision-free Path Planning of AGVs Based on Improved Dijkstra Algorithm

JIANG Chen-kai[1],LI Zhi[1,2],PAN Shu-bao[2] and WANG Yong-jun[2]
1 School of Electronic Engineering and Automation,Guilin University of Electronic Technology,Guilin,Guangxi 541004,China
2 School of Electronics and Automation,Guilin University of Aerospace Technology,Guilin,Guangxi 541004,China

Abstract　Aiming at the path planning and conflict problems of automatic guided vehicle (AGV) in flexible manufacturing systems,an improved Dijkstra algorithm based on time window is proposed to realize dynamic path planning of multiple AGVs. Firstly,the traditional Dijkstra algorithm is used to calculate the path of the multi-AGV for the scheduled task,and the degree of use of the planned path is calculated and the weighting coefficient is calculated,and then the weighted path length is updated to the database. Secondly,calculate the time for the AGV to pass through each station node, and avoid collision conflicts through the arrangement of time windows. In the end,when the conflict occurs,the path of the lower priority AGV is re-planned by calculating and setting the priority of the AGV. The simulation results show that the proposed algorithm can effectively avoid conflicts and deadlocks under the optimal path,which not only improves the system efficiency,but also makes the system more robust.

Keywords　Automatic guided vehicle,Path planning,Improved dijkstra algorithm,Time window,Collision-free conflict

1　引言

随着工业 4.0 时代的到来,许多行业掀起了一股智能搬运的升级浪潮。AGV 由于具有自动化程度高、灵活性高等优点,被广泛应用于柔性制造物流系统。而多 AGV 的调度和路径规划,对降低物流运输成本,提升生产物流效率起着至关重要的作用。在多 AGV 路径规划[1]中,有 3 个核心问题需要考虑:1)规划的路径是否存在冲突或死锁等问题;2)验证从调度任务的起点到终点之间的路径是否可行;3)系统的运行时间在规划的路径中能否达到最优。

针对上述问题,国内外学者做了大量的研究。Gao 等[2]提出了基于双层模糊逻辑的路径规划算法,通过改变多机器人的速度来降低碰撞几率,虽然系统的效率与鲁棒性得到了提高,但该算法不能保证规划的路径为全局最优。He 等[3]提出了将 Dijkstra 算法与时间窗模型相结合的方法,实现了 AGV 调度系统的顺利运行,避免了碰撞冲突,但是在多 AGV 的情况下,该方法的可行性有待验证。Zhang 等[4]提出结合 AGV 路径安全距离的控制模型,采用 Dijkstra 算法进行多任务路径规划,通过控制速度与调整路径节点的位置来减少路径冲突。Yuan 等[5]结合交通管制规则对 A* 算法进行改进,在同一路径上设置两条方向相同的单行通道,虽然该方法改善了交通拥堵与死锁问题,但其是以牺牲仓储空间、降低空间利用率的方式来解决拥堵问题。Zhu 等[6]将 Dijkstra 算法和时间窗法进行有效结合,提出了一种基于动态时间窗的 AGV 路径规划方法,通过设定 AGV 优先级解决了 AGV 柔性差、易出现死锁和冲突等问题,有效提高了 AGV 的智能化水平

到稿日期:2019-07-19　返修日期:2019-10-21　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
通信作者:盘书宝(panshubao@guat.edu.cn)

和整体运行效率。Jeon 等[7]提出了一种基于 Q 学习的新型路径规划算法，通过提前估计 AGV 在行驶中因冲突而产生的等待时间，为 AGV 规划一条行驶时间最短的路径，但是该算法求解耗时长，且无法避免死锁的产生。

针对上述问题，本文提出了一种基于时间窗的改进 Dijkstra 算法，用于路径规划。改进的 Dijkstra 算法是建立在传统 Dijkstra 算法上的一种有机路径规划算法，加入了占有度值和加权值对路径使用程度的评估，使规划的路径更加合理。该方法在任务调度环节，用 Dijkstra 算法顺序地为多辆 AGV 规划最优行驶路径，并加入时间窗来解决多 AGV 的碰撞冲突问题；当存在冲突时，通过计算并设置 AGV 的优先级，将优先级低的车辆重新进行动态路径规划，避免了冲突发生，最小化 AGV 调度对系统效率的影响。

2 多 AGV 路径规划问题的描述

2.1 应用场景描述

本文基于制造车间的柔性物流系统的应用背景进行研究。在制造车间内，每个工位点所需的物料不同，需要把对应的物料运输到指定的工位节点，因此为指明 AGV 的路径规划中的上料点与下料点的位置，需要在该环境中设置位置信息相互关联的节点[8]；同时，因为每两个工位节点之间的路程有所区别，运输时间有所不同，存在物流成本，需要计算路径代价。针对以上情况，本文在构建地图时采用拓扑建模法[9]。

图 1 为对环境进行的地图建模。在有向连接网络 $G=(V,E)$ 中，V 表示工位节点集合，E 表示路径的集合，每条路径可以表示为每两个工位节点的连线[10]。

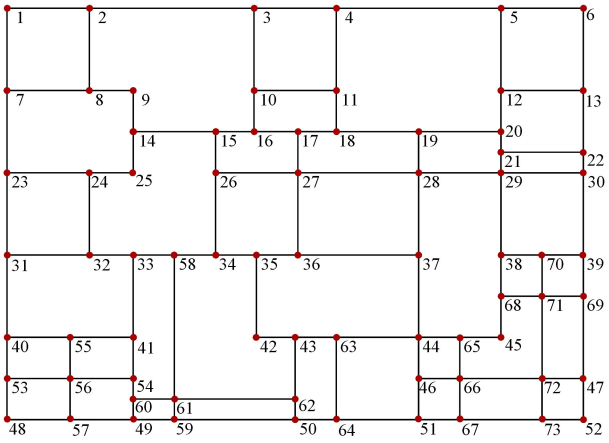

图 1 拓扑建模法构建的地图

Fig. 1 Map constructed by topological modeling

本文设置每一条边的权值为路径的实际长度，代表路径的代价。工位节点的顺序表示 AGV 的运行方向。基于柔性物流系统的应用场景，对系统作如下设定：

(1)每条边均可双向行驶，但同一时刻每条边只允许一辆 AGV 运行；

(2)AGV 在接收到调度命令后以恒定速度行驶，本实例中每辆 AGV 车身长度为 1.6m，路径长度已存储到数据库中，速度为 0.6m/s；

(3)每辆 AGV 只有在完成当前任务，到达待命点后，才可以接收下一次调度任务；

(4)为防止多辆 AGV 之间发生意外碰撞，将 AGV 的车身长度加安全距离形成的扇形区域定义为车辆间的安全区域，在此区域内不允许出现其他 AGV。

2.2 冲突描述

多辆 AGV 在运行过程中，主要存在 3 种冲突类型：节点冲突、赶超冲突和相向冲突[11-12]。由于本文假设 AGV 在进行物料运输过程中均为匀速行驶，因此对于因速度不同产生的赶超冲突，本文不予讨论。

2.2.1 相向冲突

如图 2 所示，两辆 AGV 在同一路径上相向而行时，由于在同一时刻下，每条路径只允许一辆 AGV 通行，此时两辆 AGV 将发生碰撞，造成死锁。

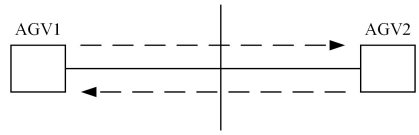

图 2 相向冲突

Fig. 2 Contradictory conflict

2.2.2 节点冲突

如图 3 所示，当 AGV1 与 AGV2 在同一时刻到达同一节点时，在该工位点处将发生节点冲突，造成碰撞，产生死锁。

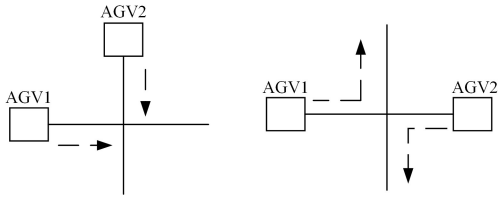

图 3 节点冲突

Fig. 3 Node conflict

2.3 多 AGV 路径规划指标

本文基于制造车间的柔性物流系统的应用场景，拟采用 AGV 的利用率 u_i 作为本文的评价指标[13]，u_i 的计算方法如下：

$$u_i=\frac{t_r}{t_w+t_r} \tag{1}$$

其中，t_w 表示 AGV 在运行过程中的等待时间，t_r 表示 AGV 的实际行驶时间，t_s 表示总的行驶时间，则总的行驶时间 $t_s=t_r+t_w$。由路径规划的评价指标函数可知，在进行多 AGV 的路径规划时，路径冲突次数越短，AGV 的等待时间就越少，AGV 的利用率就越高。

3 单源最短路径 Dijkstra 算法

3.1 Dijkstra 算法的介绍

Dijkstra 算法的核心思想是以起点为中心向外层扩展，求出该节点到其他节点的最短路径，直到扩展到终点[14]。该算法适用于单源最短路径规划问题，即从一个点开始到其他所有点的路径。

3.2 改进的 Dijkstra 算法

改进的 Dijkstra 算法是建立在传统 Dijkstra 算法上的一

种有机路径规划算法，加入了占有度值和加权值对路径使用程度的评估，使规划的路径更加合理。

改进的 Dijkstra 算法的步骤如下。

步骤 1 对所需完成调度任务的 AGV 首先采用传统的 Dijkstra 算法进行路径规划。

步骤 2 在此次路径规划中，对被规划路径中的每两个工位节点之间的路径进行规划次数的统计。

步骤 3 对路径的使用程度进行评估，本文用 o 表示路径占有度，其计算方式如下：

$$o_i=1+\frac{k_i}{k_1+k_2+\cdots+k_i+\cdots} \tag{2}$$

步骤 1 中被使用的路径用集合 $P=\{p_1,p_2,\cdots,p_i\}$ 表示，k_i 表示路径 p_i 在此次多 AGV 路径规划中的使用次数，k_i 的值在步骤 2 中进行统计计算；o_i 表示路径 p_i 的占有度，AGV 的路径占有度的集合设为 $O=\{o_1,o_2,\cdots,o_i\}$，o_i 的值越大，表示该路径在此次多 AGV 路径规划中的使用次数越多，此条路径产生碰撞的几率越大。

步骤 4 统计占有度 o 并存储到数据库中。

步骤 5 基于路径的使用程度，本文规定路径 p_i 的加权系数 w_i 的计算方式如下：

$$w_i=1+\frac{n_i(o_i-1)}{k_i} \tag{3}$$

其中，n_i 为系统在运行时路径 p_i 上所规划的 AGV 数量，AGV 通过路径 p_i 后，此段路径对应的 n_i 减 1；o_i 表示路径 p_i 的占有度；k_i 表示路径 p_i 在此次多 AGV 路径规划中的使用次数；路径长度的加权系数用集合 $W=\{w_1,w_2,\cdots,w_i\}$ 表示。

步骤 6 本文用 l_i 表示路径 p_i 的长度，l_i 通过计算得到，加权之后路径 p_i 的长度 $l_i=l_i*w_i$，将加权后的路径长度整理并记录到数据库中。

步骤 7 将加权后的路径长度更新到路径长度列表中，并重新对此次所需完成调度的 AGV 运用 Dijkstra 算法进行路径规划。

4 多 AGV 路径规划

4.1 变量的定义

本文对制造车间的柔性物流系统的应用背景进行研究，在为多 AGV 规划行驶路径时，需要计算 AGV 在某条单一路径上的行驶时间，计算方法如下：

$$t=\frac{P+L}{v} \tag{4}$$

其中，P 表示相邻两工位节点间的路径长度，L 表示车身长度，v 代表车辆的行驶速度。本文规定车辆的等待时间 $t_w=t$，以提高车辆在行驶过程中的容错能力，防止车辆在行驶过程中因打滑而发生意外碰撞。

在有向拓扑图中，假设调度系统发布 n 个调度任务，且每个调度任务均不相同，起始点的集合记为 $S_p\{s_{p1},s_{p2},\cdots,s_{pi}\}$，对应的终点集合为 $E_p\{e_{p1},e_{p2},\cdots,e_{pi}\}$，执行调度任务的 AGV 的集合为 $C\{c_1,c_2,\cdots,c_i\}$。假设自动导引车 c_n 从起始节点到终点所经过的节点数为 m，其所对应的节点集合为 ${}^nd=\{{}^nd_1,{}^nd_2,\cdots,{}^nd_m\}$，其所经过的路径集合为 $P=\{p_1,p_2,\cdots,p_{m-1}\}$，其所对应的时间窗为 ${}^nT=\{{}^nt_1,{}^nt_2,\cdots,{}^nt_m\}$。那么可将一个调度任务定义为：

$$A_n=\{s_{pn},e_{pn},{}^nd,t_s\} \tag{5}$$

其中，t_s 为任务的开始时间。

4.2 多 AGV 无碰撞路径规划算法

本算法中，通过时间窗对需要完成调度任务的 AGV 进行合理的路径分配，避免碰撞冲突的发生。车辆进入路径 p_i 的时间可通过以下方式计算获得。

(1)当 i 为起点时：${}^{in}t_{pi}=t_s$。

(2)当 i 不是起点时：${}^{in}t_{pi}={}^{out}t_{i-1}$。

AGV 离开节点 i 的时间窗为：

$${}^{out}t_i={}^{in}t_{ei}+t_e+t_w \tag{6}$$

在路径规划完之后，需要判断车辆间的冲突类型。判断方法如下：如果 ${}^xd_i={}^yd_i(x,y\in C)$，且自动导引车 c_x 与 c_y 访问节点 i 的时间窗存在交集，则节点冲突成立。如果 ${}^xd_i={}^yd_{i-1}$，${}^xd_{i-1}={}^yd_i(x,y\in C)$，且此时自动导引车 c_x 与 c_y 访问节点 i 的时间窗存在交集，则表明相向冲突成立。

AGV 在执行调度任务时会产生碰撞冲突，此时能否选择合适的 AGV 优先通过，将会对系统的运行效率产生较大的影响。本文考虑了路径代价与冲突次数，规定经过工位节点数较少的 AGV 优先通过，这样有利于减少冲突次数，提高 AGV 的运输效率；当两辆 AGV 经过的节点数相同时，路径长度较短的优先通过，这样有利于降低 AGV 的等待时间。

通过以上介绍，本文规定了改进 Dijkstra 算法的运算流程，明确了冲突类型的判别方法和时间窗的计算方法，以及当发生冲突时，AGV 通过顺序的优先级。则基于改进的 Dijkstra 算法的多 AGV 路径规划的具体实现步骤如下。

步骤 1 从 AGV 调度系统中获得调度任务的起始工位点与终点的信息，以完成任务 $A_i=\{s_{pi},e_{pi},{}^id,t_s\}$ 的初始化。

步骤 2 采用传统 Dijkstra 算法依次求解当前调度任务的最优路径，直到完成所有任务。

步骤 3 第一阶段计算每段路径的占有度，并重新计算路径的加权长度；第二阶段采用改进的 Dijkstra 算法重新规划当前调度任务的运输路径。

步骤 4 在各自的节点处，统计对应 AGV 的出入时间。

步骤 5 判断多 AGV 在调度过程中是否存在冲突，并统计正在执行运输任务的 AGV 路径中含有的工位节点个数，计算正在执行调度任务的每个 AGV 的优先级。

步骤 6 对执行调度任务的 AGV 中产生的冲突类型进行判断，当冲突类型为相向冲突时，则在产生冲突的两辆 AGV 中选择优先级较低的一辆重新进行路径规划，求解次优路径，然后回到步骤 4；如果冲突类型为节点冲突，则在产生冲突的两辆 AGV 中选择优先级较低的一辆设定为等待状态，并利用上述计算方法更新等待 AGV 的后续节点时间窗。

步骤 7 再次对执行调度任务的 AGV 判断是否存在冲突，如果任意两个 AGV 之间都不存在冲突，则判断停止。

5 仿真结果分析

5.1 仿真平台

本文采用 C# 开发的仿真平台对多 AGV 调度系统进行仿真研究，该仿真平台用于实时显示多 AGV 路径规划下的运行结果，本文规定 AGV 为匀速行驶，速度为 0.6 m/s。仿真平台的界面如图 4 所示。

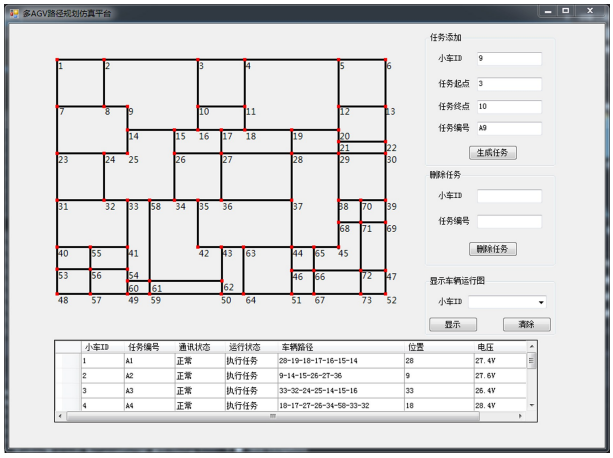

图 4 仿真平台界面

Fig. 4 Simulation platform interface

5.2 对比分析

为了对运行结果有更直观的观察，本文从以下 3 个方面进行对比：1）传统 Dijkstra 算法与改进 Dijkstra 算法的路径规划结果；2）改进 Dijkstra 算法与 A * 算法的路径规划结果；3）有冲突预防与无冲突预防下的路径规划结果。

针对传统 Dijkstra 算法与改进 Dijkstra 算法的路径规划仿真模型，为了验证改进 Dijkstra 算法能够让 AGV 有效规避占有度较高的路径，随机生成 9 个任务请求，分别为：

$A_1=\{sp_1,ep_1,{}^1d,0\}$，其中 $sp_1=28,ep_1=14$；

$A_2=\{sp_2,ep_2,{}^2d,0\}$，其中 $sp_2=9,ep_2=36$；

$A_3=\{sp_3,ep_3,{}^3d,0\}$，其中 $sp_3=31,ep_3=16$；

$A_4=\{sp_4,ep_4,{}^4d,0\}$，其中 $sp_4=18,ep_4=32$；

$A_5=\{sp_5,ep_5,{}^5d,0\}$，其中 $sp_5=11,ep_5=33$；

$A_6=\{sp_6,ep_6,{}^6d,0\}$，其中 $sp_6=4,ep_6=24$；

$A_7=\{sp_7,ep_7,{}^7d,0\}$，其中 $sp_7=34,ep_7=20$；

$A_8=\{sp_8,ep_8,{}^8d,0\}$，其中 $sp_8=11,ep_8=37$；

$A_9=\{sp_9,ep_9,{}^9d,0\}$，其中 $sp_9=3,ep_9=35$。

（1）对于 AGV 运输系统任务，采用传统 Dijkstra 算法规划的路径，如表 1 所列。

表 1 传统 Dijkstra 算法规划的路径表

Table 1 Path table planned by traditional Dijkstra algorithm

任务编号	起点	终点	传统 Dijkstra 算法规划的路径
A_1	28	14	28－19－18－17－16－15－14
A_2	9	36	9－14－15－26－27－36
A_3	31	16	31－32－24－25－14－15－16
A_4	18	32	18－17－27－26－34－58－33－32
A_5	11	33	11－18－17－27－26－34－58－33
A_6	4	24	4－11－10－16－15－14－25－24
A_7	34	20	34－35－36－27－17－18－19－20
A_8	11	37	11－18－19－28－37
A_9	3	35	3－10－16－15－26－34－35

（2）对路径进行分解，各路径出现的次数以及对应的占有度值如表 2 所列。

表 2 占有度值表

Table 2 Occupancy value table

序号	路径	规划次数	占有次数
1	28－19	2	1.03
2	19－18	3	1.05
3	18－17	4	1.07
4	17－16	1	1.01
5	16－15	4	1.07
6	15－14	4	1.07
7	9－14	1	1.01
8	15－26	2	1.03
9	26－27	3	1.05
10	27－36	2	1.03
11	33－32	1	1.03
12	32－24	1	1.01
13	24－25	2	1.03
14	25－14	2	1.03
15	17－27	3	1.05
16	26－34	3	1.05
17	34－33	2	1.03
18	11－18	2	1.03
19	34－35	2	1.01
20	4－11	1	1.01
21	11－10	1	1.01
22	10－16	1	1.01
23	35－36	1	1.01
24	19－20	1	1.01
25	3－10	1	1.01
26	28－37	1	1.01
27	31－32	1	1.01

（3）根据表 2，重新计算对应路径的加权长度，将加权后的路径长度更新到数据库中，并对传统 Dijkstra 算法的路径规划结果与改进 Dijkstra 算法的路径规划结果进行对比，则改进后发生变化的路径如表 3 所列。

表 3 已规划路径表

Table 3 Planned path table

任务编号	起点	终点	传统 Dijkstra 算法规划路径	改进 Dijkstra 算法规划路径	改进后规避占有度较高路段
A_1	28	14	28－19－18－17－16－15－14	28－27－26－15－14	19－18，18－17，14－15
A_3	31	16	31－32－24－25－14－15－16	31－32－33－58－34－26－15－16	14－15
A_4	18	32	18－17－27－26－34－58－33－32	18－17－16－15－26－34－58－33－32	26－27
A_7	34	20	34－35－36－27－17－18－19－20	34－35－36－37－28－19－20	17－18
A_9	3	35	3－10－16－15－26－34－35	3－10－16－17－27－36－35	26－34

从表 3 可以得出，采用改进 Dijkstra 算法进行路径规划时，能够有效规避占有度较高的路段，减少碰撞次数的发生，降低调度系统的难度。

针对改进 Dijkstra 算法与 A* 算法的路径规划仿真模型，为了验证改进 Dijkstra 算法能够让 AGV 具有更高的运输效率，随机生成 10 个任务请求，分别为：

$A_1=\{sp_1,ep_1,{}^1d,0\}$，其中 $sp_1=1,ep_1=18$；

$A_2=\{sp_2,ep_2,{}^2d,0\}$，其中 $sp_2=1,ep_2=38$；

$A_3=\{sp_3,ep_3,{}^3d,0\}$，其中 $sp_3=3,ep_3=43$；

$A_4=\{sp_4,ep_4,{}^4d,0\}$，其中 $sp_4=5,ep_4=51$；

$A_5=\{sp_5,ep_5,{}^5d,0\}$，其中 $sp_5=7,ep_5=34$；

$A_6=\{sp_6,ep_6,{}^6d,0\}$，其中 $sp_6=15,ep_6=51$；

$A_7=\{sp_7,ep_7,{}^7d,0\}$，其中 $sp_7=24,ep_7=41$；

$A_8=\{sp_8,ep_8,{}^8d,0\}$，其中 $sp_8=25,ep_8=52$；

$A_9=\{sp_9,ep_9,{}^9d,0\}$，其中 $sp_9=1,ep_9=46$；

$A_{10}=\{sp_{10},ep_{10},{}^{10}d,0\}$，其中 $sp_{10}=1,ep_{10}=52$。

表 4 改进 Dijkstra 算法的路径规划表

Table 4 Improved Dijkstra algorithm path planning table

任务编号	起点	终点	改进 Dijkstra 算法规划路径	路径长度
A_1	1	18	1—2—3—4—11—18	198
A_2	1	38	1—2—3—4—11—18—19—28—29—38	324
A_3	3	43	3—10—16—15—26—34—35—42—43	198
A_4	5	51	5—12—20—21—29—28—37—44—46—51	216
A_5	7	34	7—8—9—14—15—26—34	162
A_6	15	51	15—16—17—18—19—28—37—44—46—51	216
A_7	24	47	24—32—33—58—34—35—36—37—44—46—66—72—47	306
A_8	25	52	25—14—15—16—17—18—19—20—21—22—30—39—69—47—52	342
A_9	1	46	1—2—3—4—11—18—19—28—37—44—46	360
A_{10}	1	52	1—2—3—4—11—18—19—20—21—22—30—39—69—47—52	432

表 5 A* 算法的路径规划表

Table 5 A* algorithm path planning table

任务编号	起点	终点	A* 算法规划路径	路径长度
A_1	1	18	1—2—3—10—11—18	198
A_2	1	38	1—2—3—4—11—18—19—28—29—38	324
A_3	3	43	3—10—16—17—27—36—35—42—43	198
A_4	5	51	5—12—20—21—29—38—68—45—65—44—46—51	216
A_5	7	34	7—23—24—32—33—58—34	162
A_6	15	51	15—16—17—18—19—28—37—44—46—51	216
A_7	24	47	24—32—33—58—34—35—36—37—44—46—66—72—47	306
A_8	25	52	25—14—15—16—17—18—19—20—21—22—30—39—69—47—52	342
A_9	1	46	1—2—8—9—14—15—16—17—18—19—28—37—44—46	360
A_{10}	1	52	1—2—3—10—16—17—18—19—20—21—22—30—39—69—47—52	432

从表 4 和表 5 可以得出，改进的 Dijkstra 算法与 A* 算法所规划的路径长度是相同的，但是两种算法所规划的路径中有 6 次不相同，这 6 次不同的路径分别如表 6 所列。

表 6 路径节点、拐弯次数对比表

Table 6 Path node, corner comparison table

任务编号	改进 Dijkstra 算法规划路径经过节点数	改进 Dijkstra 算法规划路径拐弯次数	A* 算法规划路径经过节点数	A* 算法规划路径拐弯次数
A_1	6	1	6	3
A_3	9	5	9	5
A_4	10	1	12	2
A_5	7	3	7	3
A_9	11	3	14	5
A_{10}	15	5	16	5

从表 6 可以得出，对于某些起点到终点的路径规划，虽然 A* 算法与改进 Dijkstra 算法所规划的路径长度是一致的，但是改进 Dijkstra 算法所规划的路径的节点数和拐弯次数少于 A* 算法，节点数的减少可以缓解多 AGV 节点冲突问题，拐弯次数的减少可以缩短 AGV 的运输时间，因此改进 Dijkstra 算法具有更高的运输效率。

针对有冲突预防与无冲突预防的仿真模型，为了验证系统的防撞冲突能力和规避死锁能力，随机生成 3 个调度任务，分别为：

$A_1=\{sp_1,ep_1,{}^1d,0\}$，其中 $sp_1=14,ep_1=36$；

$A_2=\{sp_2,ep_2,{}^2d,0\}$，其中 $sp_2=19,ep_2=3$；

$A_3=\{sp_3,ep_3,{}^3d,0\}$，其中 $sp_3=7,ep_3=37$。

图 5 给出了多 AGV 路径规划在无碰撞预防下的情况。可以看出，只对 AGV 进行路径规划时，AGV1 与 AGV3 在路径 16-17 处将发生相向冲突，AGV2 与 AGV1 在路径 27-36 处将发生相向冲突，造成调度系统阻塞；而从图 6 可以看出，由于采用了基于时间窗的改进 Dijkstra 算法对多 AGV 路径规划中优先级较低的 AGV2 与 AGV3 重新选择了次优路径，因此避免了冲突与碰撞的发生。

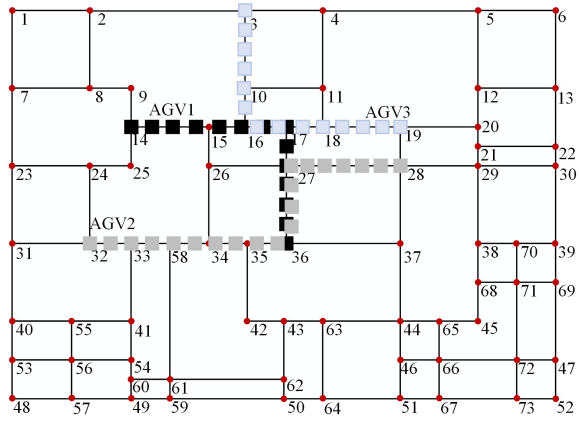

图 5 有冲突路径规划图

Fig. 5 Conflicting path planning

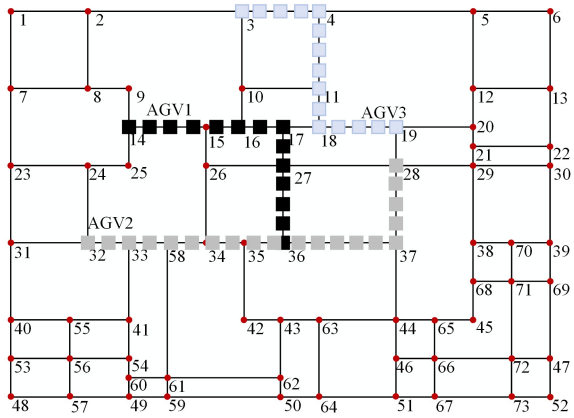

图 6 基于时间窗的多 AGV 路径规划

Fig. 6 Time window based multi-AGV path planning

同时，此仿真情况下的路径规划结果及效率分析如表 7 与表 8 所列。

表 7 无碰撞预防下 AGV 的路径规划结果

Table 7 Path planning results of AGV without collision prevention

车辆编号	AGV1	AGV2	AGV3
优先级	1	2	3
路径	14－15－16－17－27－36	32－33－58－34－35－36－27－28	19－18－17－16－10－3
里程/m	126	180	126
行驶时间/s	210	300	210
等待时间/s	0	60	30
完成调度时间/s	210	360	240
车辆利用率/%	100	83.3	87.5
总利用率/%	90.3		

表 8 防碰撞冲突下 AGV 的路径规划结果

Table 8 Path planning results of AGV under collision prevention

车辆编号	AGV1	AGV2	AGV3
优先级	1	2	3
路径	14－15－16－17－27－36	32－33－58－34－35－36－37－28	19－18－11－4－3
里程/m	126	180	126
行驶时间/s	210	300	210
等待时间/s	0	0	0
完成调度时间/s	210	300	210
车辆利用率/%	100	100	100
总利用率/%	100		

通过对比表 7 和表 8 中的数据可以看出，在运行里程相同的条件下，当系统未加入防撞冲突时，AGV 的等待时间和完成调度的时间均较长，AGV 车辆的利用率有待提升；当系统加入防撞冲突时，AGV 车辆的等待时间减少，完成调度的时间缩短，AGV 车辆的利用率得到大幅提升。

结束语 本文针对柔性制造系统中的碰撞冲突问题，提出了一种基于时间窗的改进 Dijkstra 算法，对多 AGV 进行动态路径规划。该算法不仅解决了路径规划中的多 AGV 的碰撞冲突问题，而且在路径规划过程中使得 AGV 规避了使用度较高的路径，有效降低了 AGV 发生冲突的概率；同时在有防撞冲突的预防情况下，减少了 AGV 的等待时间，缩短了 AGV 完成调度的时间，有效提高了系统的运输效率，并增强了 AGV 运输系统的鲁棒性。本文通过仿真实验验证了该方法切实可行。

参 考 文 献

[1] QIU L, HSU W J, HUANG S Y, et al. Scheduling and routing algorithms for AGVs: a survey[J]. International Journal of Production Research, 2002, 40(3): 745-760.

[2] GAO X, SU Q. Multi-robot path planning and collision avoidance based on double fuzzy logic[J]. Computer Technology and Development, 2014, 24(11): 79-82.

[3] HE L N, LOU P H, QIAN X M, et al. Conflict-free automated guided vehicles routing based on time window[J]. Computer Integrated Manufacturing Systems, 2010, 16(12): 2630-2634.

[4] ZHANG S Y, YANG Y S, LIANG C J, et al. Optimal control of multiple AGV path conflict in automated terminals [J]. Journal of Transportation System Engineering and Information Technology, 2017, 17(2): 83-89.

[5] YUAN R, DONG T, LI J. Research on the collision-free path planning of multi-AGVs system based on improved A* algorithm[J]. American Journal of Operations Research, 2016, 6(6): 442-449.

[6] ZHU L B, WANG H, WANG J L, et al. Research on path planning of parking system based on dynamic time window[J]. Chinese Journal of Engineering Design, 2017, 24(4): 440-448.

[7] JEON S M, KIM K H, KOPFER H. Routing automated guided vehicles in container terminals through the Q-learning technique [J]. Logistics Research, 2011, 3(1): 19-27.

[8] SMOLIC R N, BOGDAN S, KOVACIC Z, et al. Time windows based dynamic routing in multi-AGV systems[J]. IEEE Transactions on Automation Science and Engineering, 2010, 7(1): 151-155.

[9] WEI S, WANG L, WANG B R, et al. Improvement of A* algorithm and its application in AGV path planning[J]. Process automation instrumentation, 2017, 38(11): 51-54.

[10] ZHONG M S, YANG Y S, ZHUO Y M. Free conflict AGV path planning in automated terminals based on speed control[J]. Computer Science, 2019, 46(7): 308-314.

[11] TAI Y P, XING K X, LIN Y G, et al. Research of path planning in multi-AGV system[J]. Computer Science, 2017, 44(S2): 84-87.

[12] XU F. Research on robot obstacle avoidance and path planning based on improved artificial potential field method[J]. Computer Science, 2016, 43(12): 293-296.

[13] GHASEMZADEH H, BEHRANGI E, ABDOLLAHI A M. Conflict-free scheduling and routing of automated guided vehicles in mesh topologies[J]. Robotics and Autonomous Systems, 2009, 57(6): 738-748.

[14] WANG H, ZHU L B, WANG J L, et al. Research on path planning of parking system based on Dijstra-ant colony hybrid algorithm[J]. Chinese Journal of Engineering Design, 2016, 23(5): 489-496.

JIANG Chen-kai, born in 1993, graduate student. His main research interests include AGV dispatching system and path planning research.

PAN Shu-bao, born in 1985, B. Eng, M. Eng, lecturer. His main research interests include precision measurement and intelligent control.