doi:10.19920/j.cnki.jmsc.2026.04.006

自动化码头 AGV 无死锁在线路径规划算法①

周琛淏，焦俊玲，车阿大*
（西北工业大学管理学院，西安 710072）

摘要： 随着无人运输从单车智能化向网联集群智能化的快速发展，在封闭与半封闭式工作场景中运营大规模无人车队将面临诸如道路拥堵、车辆冲突甚至死锁等挑战. 本研究针对自动化码头 AGV 无死锁在线路径规划问题，提出了一种两阶段在线算法，将路径规划问题拆解为线路规划与轨迹规划两个紧密关联的子问题. 在第一阶段，考虑运输网络中的动态负载以及潜在的冲突影响，提出了改进 A* 算法用于规划车辆从起点至终点的最优线路，以平衡网络负载，降低道路拥堵及死锁风险；在第二阶段，提出了动态集群划分算法与无死锁轨迹规划算法，基于车辆实时状态将车辆动态地划分为多个相互独立的集群，并在集群内规划车辆预留路径从而确保车辆的无冲突无死锁通行. 仿真实验结果表明，与静态集中式路径规划算法相比，所提出的两阶段在线算法在小规模场景下能够提升约 20% 的运营效率，减少近 80% 的运算时间，并实现了在大规模场景中实时规划 100 辆到 500 辆车无死锁的行驶路径. 本研究对无人网联车辆在复杂工业场景中的应用具有重要的现实意义.

关键词： 自动化码头；AGV；车辆路径规划；无死锁；在线算法
中图分类号： U279.2　　**文献标识码：** A　　**文章编号：** 1007－9807(2026)04－0104－14

0 引　言

随着我国进入以创新驱动为主导的工业化后期阶段，无人运输车辆的发展突飞猛进并得以在不同工业场景中进行测试，例如用于运输集装箱的自动引导车 AGV、无人集装箱卡车，用于运输矿石、建筑垃圾的重载运输车，用于搬运危险品、医疗废品的无人叉车等. “十四五”时期，智能网联技术推动了无人运输车辆由单车自主运行向大集群无人化运行模式转变.

作为典型的半封闭式工业场景，自动化集装箱码头广泛部署了各类无人运输车辆，用于在岸边和堆场之间搬运集装箱. 车辆作业的高效性与安全性对提升码头的吞吐量与竞争力起着至关重要的作用. 同时，随着港口吞吐量的持续增长和单体超大型码头的投产（例如上海洋山四期码头、宁波舟山梅山港区码头），码头道路网络愈发庞大，因此需要在码头内部署更多车辆以保证运输效率. 然而现有的车辆路径规划算法无法有效解决大规模车队在行驶过程中频繁产生的车道拥堵、冲突与死锁等问题，更难以满足大规模无人运输车辆管控系统的实时决策需求.

针对上述问题，现有研究主要从规划车辆行驶线路与控制车辆高效运输两个角度开展研究. 车辆路径规划问题根据研究目标可分为两类：一类在不考虑车辆冲突、死锁等运行约束下，寻找从起点到终点的最优线路；另一类以寻找从起点到终点的最优运行轨迹为目标，并确保运行过程车辆无冲突且无死锁. 为了便于区分，分别采用线路

① 收稿日期：2023－01－05；修订日期：2024－02－08.
基金项目：国家自然科学基金资助项目（72101203；71871183）；陕西省重点研发计划资助项目（2022KW－02）；陕西省高校青年创新团队资助项目（2022－105）.
通讯作者：车阿大（1972— ），男，浙江宁波人，博士，教授，博士生导师. Email：ache@nwpu.edu.cn

规划和轨迹规划来命名这两类问题. 针对线路规划问题,可进一步分为单车规划与多车规划两种. 单车规划的常见算法包括 Dijkstra 算法、A * 算法以及相关变体算法,如动态 A * 算法等. 在有障碍物的场景下,A * 算法求解效率较高,但是在实际应用中会产生不必要的转弯等问题,导致系统效率降低. 为此,Li 和 Su[1] 考虑了车辆转弯带来的影响,张新艳等[2] 加入了电量、路径代价以及系统效率等因素,为车辆规划出更平滑的线路. 对于多车规划,多数研究聚焦于车辆路径规划 VRP 这一类问题[3-7],用于为已知任务点安排适当的行车路线以满足相关约束. 此外,Fransen 等[8] 和 Zhou 等[9] 提出的动态线路规划算法考虑了受车流影响而动态变化的路径负载或预计车辆等待时间. Cao 等[10] 在动态线路规划的基础上,增加了车辆指派决策. 随着机器学习方法的普及,Zhou 等[9] 和 Hu 等[11] 分别提出了改进的 Q 学习方法以及多智能体深度学习方法. 实验发现,与 Dijkstra 算法相比,机器学习方法在可接受的求解时间内能够得到更优的运算结果. 然而相关研究将车辆行驶道路简化为网格,未考虑车辆实际尺寸、安全距离以及车辆转弯、变道等操作,无法在实际运作过程中避免冲突与死锁.

为了保障车辆在运行过程中完全不会发生冲突与死锁,相关研究引入了时间维度,使其成为考虑车辆调度与路径的车辆行驶轨迹规划问题,求解方法包括考虑时间窗的数学规划与启发式算法[12,13]、Petri 网控制算法[14-16] 等. 以上研究将时间离散化,且需要确保系统能够精确控制车辆运行使其能够在指定时间抵达指定位置. 然而当车辆因拥堵、临时避让等原因导致与原调度计划不一致时,原始的轨迹规划将不再有效,车辆仍有可能陷入冲突死锁状况.

针对上述问题,一种解决思路是在车辆运行之前,根据车辆物理尺寸与实际行驶轨迹求解无冲突与死锁的预留通行区域,并在运行过程中通过排他性占用实现无冲突的运输[17-19]. 其中,Gawrilow[19] 提出了一种两个阶段算法,并分别设计了考虑负载平衡的最短线路算法以及基于预留通行区域的死锁预防算法. 这些方法虽然对于保障车辆行驶的安全性具有更佳的表现,但其计算时间随着车辆数与网络复杂度的增加呈指数型增长,只能用于求解较小规模的场景,例如 Kim 等[17] 与 Gawrilow 等[18] 分别求解最多 16 辆和 72 辆车. 考虑到车辆运行的动态性与求解规模的局限性,另一种解决思路是将车辆物理道路划分为多个区域,每当有车辆进入相应的区域时重新求解区域内车辆的运行方案,例如 Zhao[20] 在每个区域内采用考虑时间窗的混合整数规划模型求解.

以自动化码头为背景的文献有限,而 AGV 线路与轨迹规划问题也常以车间及仓库为研究对象. Drótos 等[21] 通过路径规划与通行调度的配合确保车辆运行时不存在死锁. Zhao 等[22] 在静态线路规划时预留足够的额外空间,以确保出现死锁时可以及时将死锁车辆移走. Brownlee 等[23] 在 Dijkstra 算法基础上,动态地给有向图的边设置时间窗以实现排他性占用. Kim 等[24] 将动态线路预留方法与 Dijkstra 算法相结合,动态求解最优运行轨迹以避免死锁状态发生. 不过,由于车间与仓库使用的 AGV 尺寸较小,以上研究同样将车辆路径简化为网络,所研究的问题与码头 AGV 有显著区别.

综上所述,现有成果在车辆线路规划和轨迹规划方面奠定了良好基础,但仍存在如下研究不足:1)现有文献对实际交通网络进行了简化,且不考虑车辆的物理尺寸与车辆行驶过程中的不确定因素,难以落地应用;2)部分文献依赖车辆在时间与空间上协同调度用以解决车辆间的冲突与死锁,但是复杂动态的作业环境使车辆很难按照原有计划完成作业;3)已有算法的求解时间随着车辆数量与场景规模呈指数增长,难以满足车辆运行过程中的实时决策要求. 基于上述情况,本研究结合上海港与新加坡港自动化集装箱码头 AGV 车队调度系统的构架,提出了一种两阶段的无死锁在线路径规划算法,用于同时解决大规模 AGV 系统的线路规划和轨迹规划问题:第一阶段基于车辆的实时状态在线更新网络负载,并通过改进的 A * 算法规划线路以减少车辆在局部区域聚集的情况发生;基于第一阶段的线路设计与车辆实时位置,第二阶段动态地为车辆划分集群,并通过并行运算为每个集群内的车辆规划无死锁、无冲突的行驶轨迹. 仿真实验表明,该算法能够在大型复杂交通网络中科学高效地规划车辆路径. 通过算法的触发机制与实时运行及决策过程

紧密融合,本研究为解决大规模无死锁车辆路径规划问题提供了新思路.

1　问题与算法框架

1.1　问题描述

为了实现车辆的精准控制,可将码头的道路网络抽象为赋权有向连通图 $\boldsymbol{G}=(\boldsymbol{V},\boldsymbol{E})$,其中点 $v\in\boldsymbol{V}$ 代表道路网络中用于控制车辆动作的逻辑点(逻辑点以阵列的方式覆盖道路网络,点与点的水平与垂直间距通常为 4 m);边 $e\in\boldsymbol{E}$ 代表连通点与点间的车辆行驶轨迹(简称"连通边"),考虑到车辆的实际行驶轨迹,连通边可以是直线、弧线、S 弯等多种形式且长度已知.

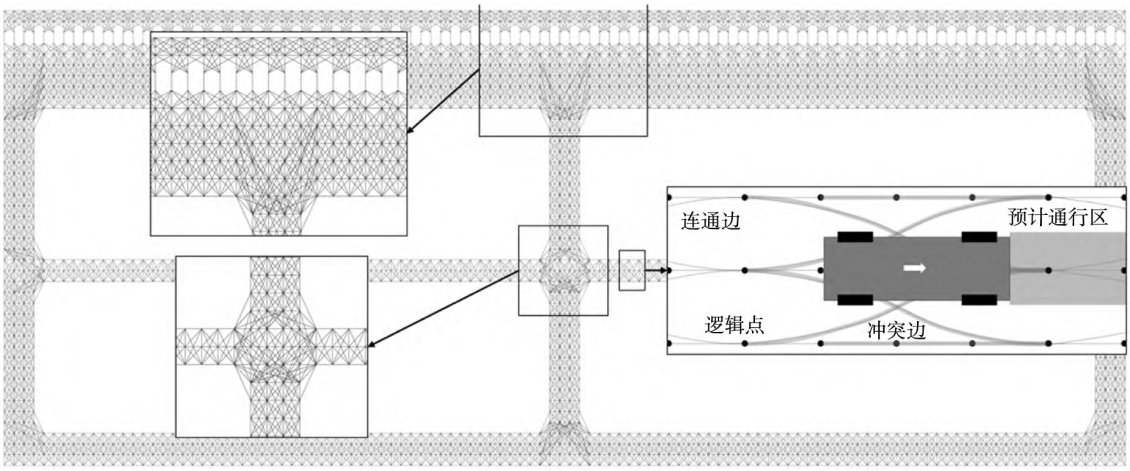

图 1　基于码头道路网络的赋权有向连通图与逻辑点、连通边、冲突边示意图

Fig. 1 Illustration of directed connected graph and corresponding logic point, arc and conflict arc

由于物理尺寸较大,码头车辆在特定连通边转弯时,会占用其他连通边的空间. 因此,需要为每条连通边定义相应的冲突边集合 Conflicts(e). 一辆车在通过边 e 时,需要同时占用其对应的冲突边,以避免其他车辆驶入该区域,如图 1 所示. 图中所示的道路网络以一个岸线长度约 1 km,纵深约 400 m 的真实码头为蓝图,建立了包含2 250个逻辑点和10 064条连通边的连通图. 图中将弧线、S 弯等较为复杂的连通边画为直线,但长度数据、冲突边集合等信息与实际轨迹保持一致.

冲突集合的存在使车辆在行驶过程中更容易与其他车辆发生冲突或死锁,影响水平运输作业效率. 常见冲突有相遇冲突(如图 2)、赶超冲突(如图 3)、占用冲突(如图 4)三种形式. 这三种形式的冲突可以通过车辆之间的简单避让或停车等待很快解决,而循环等待的死锁情况(如图 5)则需要在规划车辆行驶轨迹的过程中提前处理,以避免死锁情况发生.

一系列行驶任务 $(r_1,r_2,\cdots,r_n)$ 随着时间的推移以在线的方式陆续产生,并由调度系统分配给指定车辆. 每个任务包含任务类型(空驶或满载)、起点、终点和任务产生时间等信息. 车辆接收任务后将根据本研究提出的算法计算行驶线路,并在车辆实时运行过程中计算行驶轨迹,直到抵达终点,视为任务结束. 在码头作业的高峰期,将会有非常多的车辆同时运行并相互争夺道路资源. 如果无法有效规划行驶线路与运行轨迹,车辆将频繁陷入冲突与死锁状态,严重降低系统运行效率与安全性.

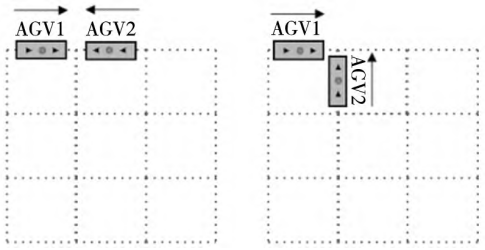

图 2　相遇冲突

Fig. 2 Heading conflict

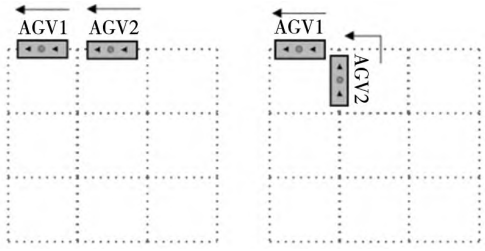

图 3　赶超冲突

Fig. 3 Catching up conflict

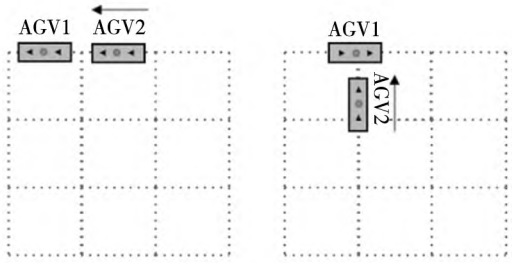

图 4　占用冲突

Fig. 4 Occupation conflict

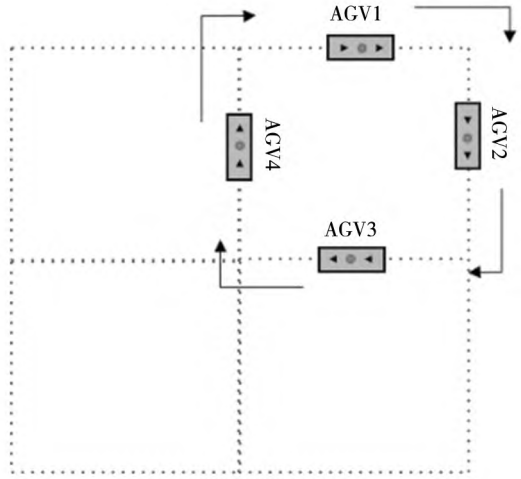

图 5　多车循环死锁

Fig. 5 Cyclic deadlock

1.2　算法框架

随着码头规模扩大，将车流均衡分配到多条线路上能够使车辆分散在更广的区域内，能够有效减少多个车辆在局部区域相互争抢道路资源状况的发生. 任意两辆车相距较远时，不会发生冲突，且发生死锁的可能性微乎其微. 车辆在运行过程中存在多种不确定性，如实时变化的道路负载与车辆位置. 为此，决策系统从全局角度一次性规划车辆从起点至终点的完整无死锁行驶轨迹既不必要，也不实际. 相反，通过线路规划将车流动态分散到多条线路上以减少车辆在局部区域的拥堵，再根据每辆车的实时位置与待行驶线路，为一定范围内有潜在冲突的车辆规划无死锁的行驶轨迹，确保车辆行进过程中无碰撞无死锁. 该思路在不牺牲作业效率与安全性的前提下，能够降低算法复杂度，并满足大规模水平运输系统的实时决策需求，相较全局规划更加具有研究与应用价值.

本研究提出的算法框架包含两个阶段共三个耦合紧密的子算法：线路规划阶段的动态线路规划算法与轨迹规划阶段的动态集群划分算法及无死锁轨迹规划算法. 为了使车辆能够连续稳定运行，三个子算法的触发机制②需要与实时运行及决策过程结合，其中线路规划用于车辆出发前的整体线路设计，而轨迹规划用于车辆运行时不断调整行驶轨迹.

1）线路规划阶段的动态线路规划算法

当任何车辆即将从任务起点出发时，或当调度系统需要对相关车辆更新行驶线路时，触发该算法. 该算法基于网络各条连通边上的负载水平，以车辆的起点与终点作为输入，计算最优线路并输出连接起点与终点的连通边的集合，如图 6 所示. 该算法不考虑车辆行驶途中的冲突及避让等运行约束，其目标是为车辆寻找一条可以减少道路拥挤、变道转弯以及行驶时间的运行线路. 由于网络中的每条边都具有方向性，且车辆行驶轨迹具有连贯性，算法规划出来的线路并非欧氏距离上的最短路径.

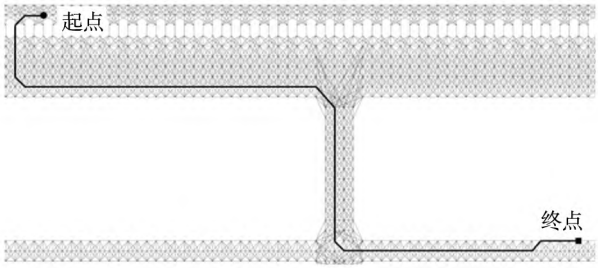

图 6　动态线路规划算法示意图

Fig. 6 Illustration of dynamic route planning algorithm

2）轨迹规划阶段的动态集群划分算法

由于道路网络对应的有向连通图 $\boldsymbol{G}=(\boldsymbol{V},\boldsymbol{E})$ 的规模很大，求解无死锁的轨迹规划不仅运算难度大，且车辆在行驶过程中存在不确定性，往往容易偏离原有计划. 为此，本研究提出了动态集群的概念，即基于网络中所有车辆的实时位置与线路规划算法所得的运行线路，按照时间或距离阈值确定车辆预计行驶的区域，将阈值范围内存在行驶区域重叠的车辆划分为一个集群，并随着车辆位置动态更新集群.

② 限于篇幅，触发机制设计未在本文列出，留存备索.

如图 7 所示，当车辆接收到调度系统为其分配的行驶线路后，将依次沿着指定的连通边行驶. 在行驶过程中，车辆每次通过一个节点将触发该算法. 每辆车需要基于自身当前位置，判断在未来一定距离内（例如 100 m、150 m、200 m 等），待行驶的连通边与冲突边集构成的预计通行区与其他车辆的待行驶边集是否存在重叠. 如果车辆与其他车辆的预计通行区不存在重叠，车辆自己将构成一个集群（如 AGV1）；如果存在重叠（如 AGV2 和 AGV3），则表示这些车辆在未来短时间内存在划到一个集群中的可能，因此需要避免死锁的发生. 之后，针对各个集群独立地计算无死锁的车辆运行轨迹，可以在局部区域内实现更加精确的决策.

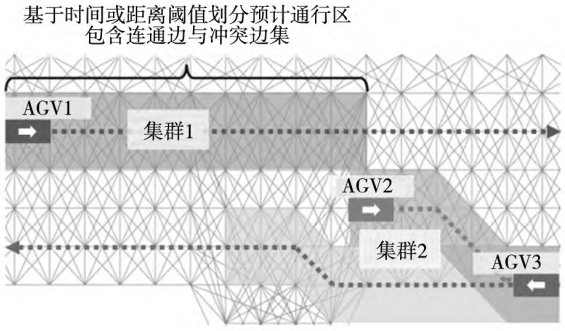

图 7　动态集群划分算法示意图

Fig. 7 Illustration of dynamic vehicle grouping algorithm

3）轨迹规划阶段的无死锁轨迹规划算法

当所有车辆完成集群划分后，将产生多个相互独立的集群. 在集群 j 内为每辆车规划无死锁的行驶轨迹只需要考虑集群范围内的边集，即形成了一个子图 $\boldsymbol{G}_j = (\boldsymbol{V}_j, \boldsymbol{E}_j)$. 该算法在车辆即将抵达预留路径的终点前，或者集群受其他车辆影响发生变动时触发，输出集群内车辆行驶轨迹预留路径表（如图 8 所示），确保行驶过程中不会发生冲突与死锁. 由于集群之间相互独立，因此每个集群的运算任务可以用并行计算的方式处理.

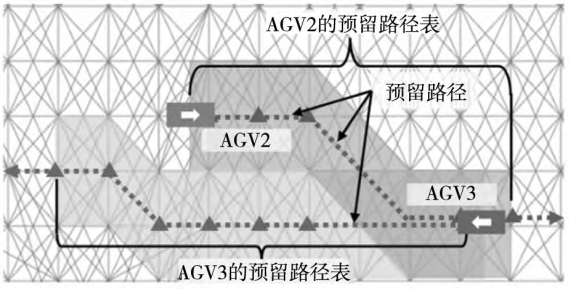

图 8　无死锁轨迹规划算法示意图

Fig. 8 Illustration of deadlock-free path planning algorithm

由以上算法框架可知，与现有多数两阶段问题[25]不同，本算法的第一阶段算法需要随着新任务的开始而反复执行，而第二阶段算法则随着车辆实时运行而反复执行.

算法为车辆行驶提供的决策仅仅是可连续行驶的连通边集，并不会输出具体调度细节，即“何时到哪”. 当车辆相遇时，可按照“先占用先通行”原则依次通行. 通行判断由车辆运行控制机制决定，不包含在算法框架之中. 这样设计的优势是可以处理各种车辆运行中的不确定因素. 例如，无论车辆行驶快慢，算法始终保障所有车辆的预留路径不存在死锁状态；即使某车辆因故障临时停车或者更改目的地，算法可实时计算新的预留路径或新的线路规划，确保相关车辆正常通行.

2　算法设计

2.1　动态线路规划算法

标准 A * 算法使用当前所选的待评估点与终点的预估距离作为评价函数，能够以更快的速度找到一条相对较优的线路，从而提高搜索效率. 然而，仅以距离最短为优化目标会导致车辆转弯次数过多以及大量任务分配相同线路等问题[2]. 因此，本研究提出的改进 A * 算法以连通边为决策单位，引入边实时负载的概念来表示每条边上已经规划且尚未通过的车辆数量，并随着车辆前行不断更新边上的负载权值. 对边 e 的评价函数 $F(e)$ 由四个部分组成，即 $F(e) = T^r(e) + T^l(e) + T^c(e) + T^n(e)$，具体定义如下.

1）相邻两边的转弯时间成本

$$T^r(e) = \sum_{e_i = e^o, \cdots e} t^r(e_{i-1}, e_i) = \sum_{e_i = e^o, \cdots e} \{B_1 \times \mathrm{Abs}[\theta(e_i) - \theta(e_{i-1})]\} \quad (1)$$

$T^r(e)$ 表示从车辆当前任务的起始边 e^o 到检测边 e 的所有边转弯时间成本总和. 由于车辆转弯时间与转弯角度有关，相邻两边 e_{i-1} 与 e_i 之间的转弯时间成本 $t^r(e_{i-1}, e_i)$ 可由两条边所构成夹角的弧度绝对值 $\mathrm{Abs}[\theta(e_i) - \theta(e_{i-1})]$ 计算，B_1 表示转弯时间成本系数.

2）负载平衡时间成本

$$T^l(e)=\sum_{e_i=e^o\cdots e}t^l(e_i)=\sum_{e_i=e^o\cdots e}[B_2\times\sigma(e_i)\times t^v(e_i)] \tag{2}$$

$T^l(e)$ 表示从车辆当前任务的起始边 e^o 到检测边 e 之间的所有边负载成本总和. 每条边 e_i 的负载成本 $t^l(e_i)$ 由该条边上的实时负载 $\sigma(e_i)$ 和正常通行时长 $t^v(e_i)$ 决定. 其中，$\sigma(e_i)$ 表示边 e_i 上当前被其他车辆已申请且尚未经过的线路数量，$t^v(e_i)$ 表示车辆在单条边 e_i 上的通行时长. B_2 表示拥堵时间成本系数. 当一条边有多条线路需要通行时，更容易发生拥堵，其通行时间将会明显上升.

3）冲突死锁时间成本

$$T^c(e)=\sum_{e_i=e^o\cdots e}t^c(e_i)=\sum_{e_i=e^o\cdots e}\left\{B_3\times\sum_{\forall e_k\in Conflicts(e_i)}\left[\sigma(e_k)\times t^v(e_k)\right]\right\} \tag{3}$$

$T^c(e)$ 表示从当前任务的起始边 e^o 到检测边 e 之间的所有边对应的冲突边负载成本总和. 每条边 e_i 的冲突死锁成本 $t^c(e_i)$ 由该条边对应的冲突边 $e_k\in Conflicts(e_i)$ 的实时负载与正常通行时长决定，B_3 表示冲突时间成本系数.

4）距离任务终点的预计通行时间成本

$$T^n(e)=\frac{\sqrt{(x_E(t)-x(e))^2+(y_E(t)-y(e))^2}}{\delta} \tag{4}$$

$T^n(e)$ 依据欧氏距离计算检测边 e 到任务结束边 t 之间的预计通行时间. 其中，检测边 e 起点的横纵坐标分别为 $x(e)$ 和 $y(e)$，任务终点的横纵坐标分别为 $x_E(t)$ 和 $y_E(t)$，δ 表示车辆匀速行驶速度.

2.2　动态集群划分算法

在线路已知的情况下，车辆在任意时刻根据其当前位置，基于距离阈值可以得到包含连通边与相应冲突边的预计通行区. 假设所有车辆的集合为 A，触发算法的车辆为 a_1，已划分集群的车辆集合为 $A'=\varnothing$. 具体算法步骤如下.

步骤 1　初始化时，更新 a_1 的预计通行区 $\boldsymbol{E}^{a_1}$，创建集群 A_1 并将 a_1 加入其中. 更新当前集群对应的子图 $\boldsymbol{G}_1=(\boldsymbol{V}_1,\boldsymbol{E}_1)$，$\boldsymbol{E}_1=\bigcup_{a\in A_1}\boldsymbol{E}^a$. 将 a_1 加入集群 A'.

步骤 2　从 A 中取一辆尚未划分集群的车辆 $a_2\in A-A'$，并更新其预计通行区 $\boldsymbol{E}^{a_2}$.

步骤 3　判断 $\boldsymbol{E}^{a_2}$ 与已有集群的边集 $\boldsymbol{E}_k$ 的交集是否为空集. 若与所有集群的边集交集皆为空集，则车辆 a_2 的路线与这些集群中车辆的路线在阈值范围内不存在空间交叉，转步骤 4；若与一个或多个集合 $\boldsymbol{E}_k$ 不为空集，转步骤 5.

步骤 4　保留现有集群，新建集群 A_j 并将 a_2 加入其中，更新当前集群对应的子图 $\boldsymbol{G}_j$，将 a_2 加入集群 A'，转步骤 6.

步骤 5　将多个集群 A_k 合并为新的集合 A_j，并将 a_2 加入其中，更新当前集群对应的子图 $\boldsymbol{G}_j$，将 a_2 加入集群 A'，转步骤 6.

步骤 6　如果 $A'=A$，算法停止，否则转步骤 2.

以下以单位长度 4 为距离阈值，举例说明动态集群划分的过程：如图 9 所示，假设当前时间为 t_1 时，根据上述算法可得 AGV1 与 AGV2 构成集群 1，AGV3 构成集群 2；如图 10 所示，在 t_2 时，AGV2 前移并同时与 AGV1 和 AGV3 存在交集，因此将 AGV3 加入集群 1，并删除集群 2.

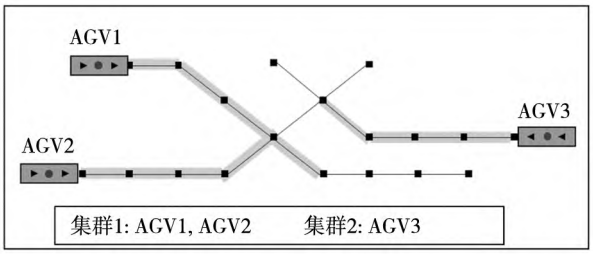

图 9　t_1 时的集群划分

Fig. 9 Group dividing at t_1

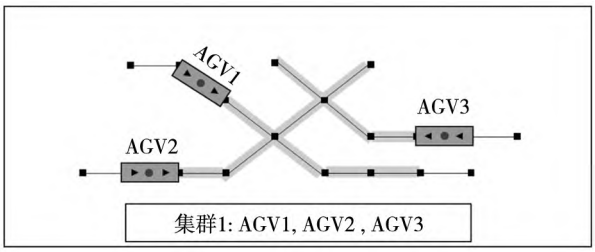

图 10　t_2 时的集群划分

Fig. 10 Group dividing at t_2

2.3　无死锁轨迹规划算法

在每次集群更新后，需要更新集群内车辆的预留路径表，使车辆实现局部区域内的连续行驶

与绝对安全性. 为此,本研究在 Gawrilow 等[19]静态死锁预防算法的基础上,提出了基于集群与子图的无死锁轨迹规划算法. 该算法的主要思路是针对集群中每辆车的待行驶线路,依次执行预留路径划分算法,并将每一组已确定的车辆预留路径信息记录在死锁检测图上. 在后续的计算过程中,通过死锁检测算法判断新的预留请求是否在死锁检测图上存在死锁. 如果构成死锁则更改预留请求,直至无死锁时,确认该组预留请求,并将预留路径信息记录在死锁检测图上. 该算法的核心要点是将每辆车待行驶线路的边集,转化为不会产生冲突死锁的预留路径表,而每段预留路径可以是一条或多条边.

1)预留路径划分算法步骤

假设集群内的车辆已确定一个优先级序列,从序列中第一辆车开始,依次计算每辆车的预留路径表. 假设车辆在集群内的线路共有 M 条边,如图 11 所示,该算法具体步骤如下.

步骤 1　获取当前的死锁检测图. 对于第一辆车,死锁检测图为空. 从最后一条边 e_M 开始,尝试令前一条边 e_{M-1} 作为待预留边,边 e_M 作为预留区域.

步骤 2　使用死锁检测算法检测该预留请求(即待预留边与预留区域)与已确定的路径预留方案是否产生死锁情况. 如果无死锁,转步骤 3;如果有死锁,转步骤 4.

步骤 3　保留该组预留请求并记录在预留路径表与死锁检测图上. 然后将该预留方案中的待预留边(如步骤 1 中所示的 e_{M-1})作为下一组待确认预留请求的预留区域,令其前一条边(如步骤 1 中所示的 e_{M-2})作为待预留边,重复步骤 2 的检测过程. 直至完成对第一条边 e_1 的预留,即找到当前线路的预留路径表,停止计算过程.

步骤 4　使待预留边向前移动一条边(如步骤 1 中情况,将待预留边变更为 e_{M-2}),预留区域沿着线路向前增加一条边(如步骤 1 中情况,将预留区域变更为 $e_{M-1} \cup e_M$),再次检测该组预留请求与已确定的预留路径信息是否产生死锁情况. 如果无死锁,转步骤 3;如果有死锁,转步骤 4.

死锁检测算法的主要作用是避免在预留路径时产生潜在死锁. 定义死锁检测图 $\boldsymbol{\Gamma} = (\boldsymbol{E},\boldsymbol{R})$,其中,节点 $e \in \boldsymbol{E}$ 表示有向图 $\boldsymbol{G}$ 中的连通边 e,边 $r \in \boldsymbol{R}$ 表示图 $\boldsymbol{G}$ 中两条边的关系. 假设车辆当前正在图 $\boldsymbol{G}$ 的边 e_1 上且需要预留边 e_2,则存在 $r = (e_1,e_2) \in \boldsymbol{R}$,而反映在死锁检测图 $\boldsymbol{\Gamma}$ 上是以 e_1 为起点,e_2 为终点且具有表示该车辆路径特定颜色的一条边. 如图 12 所示,假设每一辆车的预留路径可以用一种颜色表示,如果在死锁检测图 $\boldsymbol{\Gamma}$ 上存在死锁,那么必然存在一个环形,环上的每一段路径颜色各异. 存在这样的彩色环路就意味着一组车辆正在循环请求预留已经被其他车辆占用的边,从而构成死锁. 为了避免死锁产生,在划分预留路径时,需要检验新的预留请求是否与已经确认的预留请求构成彩色的环,即可判断是否和其他车辆构成死锁. 如果死锁将要发生,当前预留方式将不可行,需要通过改变预留区域的范围调整预留路径表,从而实现无死锁的预留路径划分.

图 11　待划分预留路径的车辆线路

Fig. 11 Route of the vehicle awaiting path reservation

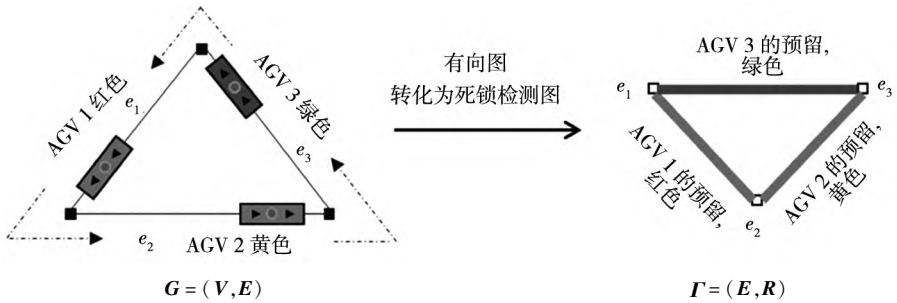

图 12　彩色路径检测死锁示意图

Fig. 12 Illustration of deadlock detection using colorful path

2）死锁检测算法步骤

假设算法需要检查当前车辆 a 的当前预留请求（待预留边 q 与预留区域 S）与已确定的路径预留方案是否产生死锁情况，当前车辆预留路径的颜色为 c_a. 若待预留边 q 与预留路径中的任意边 $s \in S$ 间存在彩色环，则表示该组预留请求与已保留的预留方案会产生死锁；若无彩色环则表示该组预留方案不会产生死锁，可以保留并进行后续运算.

步骤 1　获取当前的死锁检测图，死锁检测图记录了每辆车的预留路径表并指定了唯一的颜色. 针对车辆 a 当前的预留请求（待预留边 q 与预留路径 S），将待预留边 q 设为根节点，并作为首次迭代的父节点，执行步骤 2.

步骤 2　依次检查所有父节点的可达子节点，是否存在一条从根节点 q 到该子节点之间不包含颜色 c_a 且无重复颜色的彩色环；若是，执行步骤 3；若否，表示不存在从待预留边 q 至预留路径 S 中的边 $s \in S$ 之间的一条彩色环，则确定该车辆当前的预留请求与已确定的预留方案未产生死锁情况，算法停止.

步骤 3　如果查找到的可达子节点与其父节点的连通边的颜色不是 c_a，且不同于从根节点 q 到其父节点间的彩色环中其他颜色，判断该连通边是否是预留路径 S 中的一条边；若否，将该可达子节点作为下一次迭代的父节点，执行步骤 2；若是，则找到一条从待预留边 q 至预留路径 S 中的边 $s \in S$ 之间的一条彩色路径，则确定该车辆当前的预留请求与已确定的预留方案产生死锁情况，算法停止.

集群状态（包括空间范围、成员组成等）会随着车辆不断地运行发生动态变化. 在集群状态更新后，算法将会计算出新预留路径表，此时需要更新当前车辆对预留路径的占用情况，以避免发生死锁. 对于车辆当前所占用的预留路径，与先前的预留路径表进行对比，若算法计算出更长的预留路径，则更新预留路径表并占用新增的边. 反之，则减少原有的占用范围，并将多出来的部分设置为未占用状态；若预留路径未发生变动则按原方案行驶，以此使得道路资源在保障安全的前提下充分使用.

3）无死锁轨迹规划算法示例

以下简要介绍无死锁轨迹规划算法如何解决图 13 所示常见的循环等待问题. 如图 14 所示，假设 AGV1、AGV2 与 AGV3 先后完成了路径预留并分别在死锁检测图中留下了红色、橙色和黄色的预留方案. 此时 AGV4 由边 e_2 预留边 e_4 时（假设是绿色），会在 e_2、e_4、e_7 与 e_5 构成一个颜色各异的环. 如果保留该预留方案，意味着四辆 AGV 存在循环等待的死锁状况. 因此，需要令 e_9 同时预留 e_4 及 e_2，使死锁检测图上存在一个环，这个环上至少存在 2 条边颜色相同（绿色）. 在实际运行时，当 AGV4 即将抵达 e_9 尽头，需确保 e_2 与 e_4 同时没有其他车辆占用，继续行驶且不会陷入死锁状态.

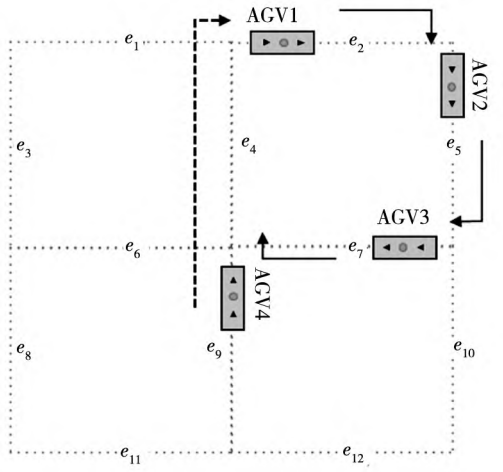

图 13　四车轨迹规划示意图

Fig. 13 Illustration of deadlock among four vehicles

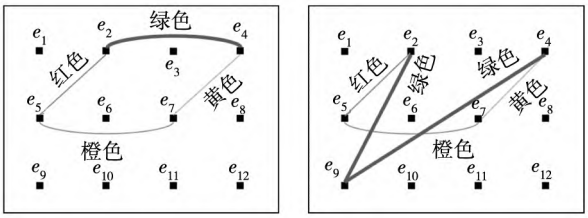

图 14　四车预留路径示意图

Fig. 14 Illustration of path reservation among four vehicles

2.4　算法时间复杂度分析

改进 A* 算法的理论时间复杂度为 $O(|\overline{B}|^{|L|})$，其中 $|\overline{B}|$ 是每个节点的平均出边数，$|L|$ 是从任务起点到终点的最短路径的长度. 通过使用改进的评价函数来估算抵达终点的代价，可大幅减少实际搜索的节点数. 因此，改进的 A*

算法的实际运算时间比理论运算时间小很多.

动态集群划分算法与轨迹规划算法的时间复杂度分别为 $O(|A|^2 \cdot |E^a| \cdot \ln|E_j|)$ 和 $O(|A| \times 2^{|E^a|} \cdot |E_j|)$,其中 $|A|$ 代表车辆数,$|E^a|$ 代表车辆预计通行区内的边数,$|E_j|$ 代表子图中的边数. 对于动态集群划分算法,由于车辆数远小于网络的节点数与边数,相对来说 $|A|^2$ 并不会给运算带来多少困扰. 对于轨迹规划算法,控制 $|E^a|$ 的大小会显著影响算法效率. 因此,对于这两个算法来说,通过限定车辆与集群轨迹运算范围,可以极大减少运算时间,同时又能保障所得车辆行驶轨迹不存在死锁.

3 仿真结果及分析

3.1 实验设计

自动化码头在完成建设、调试并投入生产后,将保持 24 h 连续运作,因而码头运营企业难以直接部署新算法或新策略来测试其对实际生产的影响. 为此,本研究设计了基于离散时间的仿真模型,并将相关算法集成在一起,实现在不同时间点对不同算法的触发,适用于验证两阶段在线算法的性能.

本研究首先以真实码头布局为基础,生成了包含 269 个逻辑点、914 条边的小规模场景用于算法测试. 随后,基于真实码头形成了包含 2 250 个逻辑点和10 064条连通边的算例. 由于上述仿真模型环境存在不确定因素较少,3.3 小节的实验结果皆为 5 次运算的平均值. 所有算法与平台在 VisualStudio 2019 环境下采用. Net 开发,在配备 8 核 2.30 GHz 的英特尔 i7 处理器及 16 GB 内存的普通台式机上运行,操作系统为 64 位的 Windows 10.

实验分为两个部分,首先验证改进 A * 算法对于线路选择以及车辆运行效率的影响;之后,对比了集群式与集中式轨迹规划算法的性能,探究了预计通行区域的距离阈值的设置对于车辆运行效率的影响,以及两阶段在线算法在大规模场景下的有效性. 集中式轨迹规划可通过将预计通行区距离阈值设为非常大的值来实现,即在任何时刻将所有车辆纳入同一个集群计算. 仿真实验模拟实际场景中车辆在码头前沿与堆场之间往复运行的行驶任务,所有的测试方案均通过在地图相应位置随机生成起点和终点,产生时间服从一定的时间间隔.

需要注意的是,该模型在离散时间点上触发相关算法,无法完全模拟车辆实时运行过程中的算法触发逻辑. 为此,本研究搭建了一套高拟真仿真与管控平台用于模拟车辆运行与中央调度系统交互③.

3.2 改进 A * 算法验证与分析

本研究首先在不考虑车辆行驶轨迹的情况下,探究改进 A * 算法对线路选择的影响;其次结合仿真模型,探究改进 A * 算法对车辆行驶效率的影响;最后分析了相关时间成本系数取值对线路选择的影响.

1)改进 A * 算法对线路选择的影响

在小规模场景且车辆数为 10 辆、50 辆、100 辆、150 辆、200 辆时,为了探究改进 A * 算法对线路选择的影响,将转弯时间成本系数 B_1、拥堵时间成本系数 B_2以及冲突时间成本系数 B_3,其中一个系数取 0.3,其他系数取 0 进行对比实验.

转弯时间成本系数 B_1对线路规划的影响如图 15 所示. 其中,改进 A * 算法的直行边选择次数均值比标准 A * 算法多 2 倍以上,并其显著降低了转弯弧度均值,意味着改进 A * 算法在线路选择时更倾向于选择直行边.

负载及冲突边集负载的设置对线路规划的影响如图 16 和图 17 所示. 改进 A * 算法能够在选择车辆通行线路时,能够将车流分散到更多负载较低的线路,通过降低连通边与冲突边上的车辆通行次数,减少车辆在实际行驶过程中的拥堵与冲突. 其中,改进 A * 算法引入拥堵时间成本系数 B_2,显著减小了单条线路的负载均值与最大值,且当车辆数量增加时,改进效果越发显著. 引入冲突时间成本系数 B_3在一定程度上降低了关联冲突边集的负载均值与最大值,且车辆数量增

③ 限于篇幅,平台设计未在本文列出,留存备索.

加使改进效果略有提升. 当车辆数为 200 时，B_2 对单条线路负载均值的改善率达到 44.5%，B_3 对单条线路冲突变边集负载均值的改善率达到 11.2%.

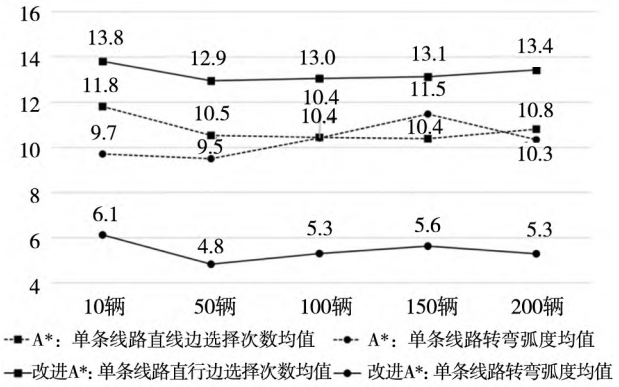

图 15　转弯时间对单条线路的影响

Fig. 15 Effects of turning time factor to single vehicle route

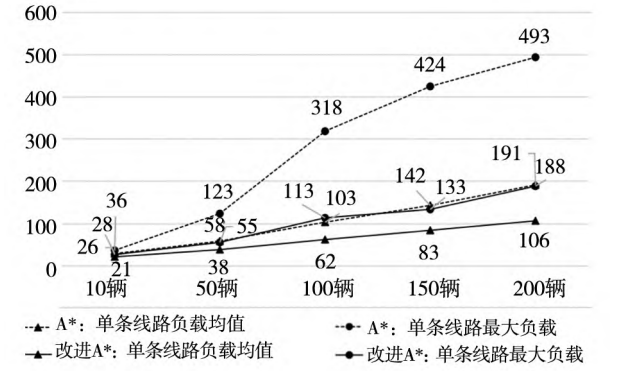

图 16　负载对单条线路的影响

Fig. 16 Effects of workload factor to single vehicle route

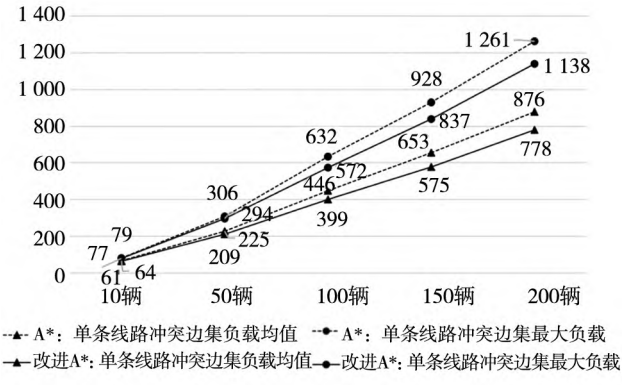

图 17　负载对单条线路冲突边集的影响

Fig. 17 Effects of workload factor to conflict arcs of single vehicle route

2）改进 A * 算法对车辆运行效率的影响

为了探究改进 A * 算法对车辆运行效率的影响，本研究在小规模场景的仿真环境下分析算法对车辆平均行驶时间和全部任务完成时间的影响. 其中，改进 A * 算法中系数 B_1、B_2 和 B_3 的值均设为 0.3，车辆数为 10、30、50、75、100.

实验结果如表 1 所示，改进 A * 算法通过减少变道转弯与分散车流，有利于减少车辆单次平均运行时间，并提升整体系统的运行效率. 其中，车辆平均行驶时间随车辆数增加而平稳增加，改善率呈增长趋势. 当车辆数达到 100 时，车辆的转弯次数多、拥堵冲突现象更为严重，因此改善率也提升到 9.6%.

3）时间成本系数取值对线路选择的影响

评价函数的四个部分虽然以时间作为统一量纲，但各个部分是对转弯、拥堵和冲突场景的估计，并非实际车辆行驶或等待时间. 因此时间成本系数的取值具有一定主观性，需要在应用中结合实际场景进行实验调整. 为了研究系数取值对线路选择的影响，实验选取 100 辆车的小规模场景，分别设置 B_1、B_2 或 B_3 的值为 0、0.1、0.2、0.3 与 0.5，同时保持另外两个系数为 0，以此进行灵敏度分析. 其中，单条边负载均值等于路网中所有边上的负载之和除以车辆使用的边的数量.

实验结果如表 2 所示，随着各个时间成本系数的增大，相应的道路选择评价指标都有所改善. 不过随着时间成本系数增加，改善率逐渐下降. 例如当系数取 0.3 和 0.5 时，各个指标的改进程度没有系数取 0.2 和 0.3 时差异显著. 因而上文取 0.3 是合理的.

3.3　两阶段在线算法验证与分析

为探究算法有效性，本研究首先基于小规模场景，对集中式与集群式的轨迹规划算法的性能进行对比，在 10 辆 ~ 100 辆车的数量范围内，将算法检查状态更新的时间间隔 Δt 设置为 1s. 考虑到刹车距离、算法触发频率等因素，预计通行区域的阈值设置不低于 100 m（即 25 条边）. 之后，在 50 辆车场景下对不同距离阈值（25 条边、30 条边、35 条边、45 条边、55 条边）进行了灵敏度实验. 最后，在车辆数量为 100 辆到 500 辆的大规模场景下验证了算法的有效性.

1）集中式与集群式的轨迹规划算法性能对比

分别使用集中式与集群式算法进行车辆无死锁轨迹规划，并在 10 辆、20 辆、30 辆、50 辆、

75 辆、100 辆车的算例下进行实验. 实验选取了三个评价指标如表 3 所示,其中,无死锁通行方案决策次数表示执行轨迹规划算法的次数,也表示集群变动或更新的次数;决策时间均值表示各个集群每次执行轨迹规划算法的时长,代表算法的时间效率.

从实验结果可知,集中式轨迹规划算法在当前地图规模下只能处理 20 辆车以下的轨迹规划,而集群式算法则可以处理 10 辆 ~ 100 辆车的场景,从而验证了两阶段算法的有效性. 此外,由实验结果可知,集群式轨迹规划算法有利于降低车辆平均行驶时间,在 10 辆和 20 辆车的规模下,车辆平均行驶时间平均改善率在 20% 左右,运算时间均值改善率为 83.0% 和 76.2%. 这一结果说明了集中式算法求解所得的路径预留方案过于保守,导致较多的等待时间,而集群式算法能够有效提高运输效率;同时,对车辆进行集群管理并分散计算任务,能够加快决策效率. 此外需要关注,集群式轨迹规划算法的决策次数较多,因此在实际应用时需要为中央控制系统配备更多的计算资源来实现分布计算.

2)距离阈值的设置对于车辆运行效率的影响

在 50 辆车的规模下,将预计通行区域距离阈值分别设置为 25 条边、30 条边、35 条边、45 条边、55 条边,统计车辆运行效率的变化情况,实验结果如表 4 所示. 实验结果表明,随着距离阈值的增加,全部运输任务完成时间及车辆平均运输时间逐渐增加,预留边最大长度逐渐增长. 意味着较大的车辆距离阈值会使更多车辆同时纳入集群. 为了避免死锁,每辆车需要预留更长的距离,对应的车辆运行方案也会更加保守. 在实际应用时需要根据具体的运行场景特性,包括车辆总数、水平运输网络大小、安全规范等,选取合适的预计通行区域距离阈值.

3)大规模场景下两阶段算法的有效性验证

为了进一步验证算法有效性,以下基于 2 250 个逻辑点和10 064 条连通边的网络,对 100 辆、200 辆、300 辆、500 辆车的随机算例进行测试. 实验结果如表 5 所示. 实验结果显示,所有车辆均能达到任务终点,说明算法能够保证车辆在行驶过程中不发生死锁. 单次决策时间平均值均在 0.001 s 的级别,最大决策时间也在秒级以内,验证了所提出的算法的有效性,能够满足调度系统实时决策的要求.

表 1 A * 算法与改进 A * 算法性能比较

Table 1 Comparison between A * algorithm and improved A * algorithm

评价指标	算法	10 辆	30 辆	50 辆	75 辆	100 辆
车辆平均行驶时间/s	A *	377	419	420	471	494
	改进 A *	374	411	413	442	446
	改善率	0.9%	1.9%	1.7%	6.1 %	9.6%
全部任务完成时间/s	A *	683	1 097	1 599	2 219	2 598
	改进 A *	677	1 081	1 504	2 053	2 571
	改善率	0.8%	1.4%	5.9%	7.5%	1.0%

表 2 改进 A * 算法时间成本系数灵敏度分析

Table 2 Sensitivity analysis of the time cost coefficients of the improved A * algorithm

系数	指标	灵敏度取值				
		0	0.1	0.2	0.3	0.5
B_1	单条线路直行边选择次数均值/次	10.4	10.7	11.9	13.0	13.3
	单条线路转弯弧度均值/(弧度)	10.4	8.6	6.4	5.3	5.2
B_2	单条边负载均值/(辆车)	3.5	2.8	2.5	2.4	2.3
B_3	单条边冲突边集负载均值/(辆车)	23.7	22.0	22.00	21.9	21.8

表 3　集中式与集群式算法性能比较

Table 3 Comparison between central mode and distributed mode algorithms

评价指标	算法模式	10 辆	20 辆	30 辆	50 辆	75 辆	100 辆
车辆平均行驶时间/s	集中式	451	572	—	—	—	—
	集群式	361	430	426	425	421	450
	改善率	19.9%	24.9%	—	—	—	—
无死锁通行方案决策次数/次	集中式	10	20	—	—	—	—
	集群式	144	1 039	1 818	4 364	6 676	9 092
决策时间均值/s	集中式	0.010 8	0.011 9	—	—	—	—
	集群式	0.001 7	0.002 8	0.002 6	0.002 2	0.002 2	0.001 9
	改善率	83.0%	76.2%	—	—	—	—

表 4　预计通行区域距离阈值对算法性能的影响

Table 4 Impact of distance thresholds for expected passage areas to algorithm performance

评价指标	25 条	30 条	35 条	45 条	55 条
车辆平均行驶时间/s	429	433	472	507	569
全部任务完成时间/s	1 443	1 488	1 736	1 878	2 205
预留边的最大长度/(条边)	3	4	11	13	16

表 5　大规模场景测试结果

Table 5 Results of large-scale test case

评价指标	100 辆	200 辆	300 辆	500 辆
车辆平均行驶时间/s	276	309	321	314
无死锁通行方案决策次数/次	6 924	20 056	28 570	50 308
决策时间均值/s	0.002 1	0.001 9	0.001 8	0.002 2
决策时间最大值/s	0.341 4	0.193 0	0.132 0	0.513 8

4　结束语

针对现有车辆路径规划算法面对复杂网络与大规模车队无法高效求解的问题，提出了包含线路规划与轨迹规划的两阶段在线算法，并将其嵌入实时决策过程. 为了验证算法在大规模自动化码头场景下的有效性，建立了基于离散时间的仿真模型. 通过大量数值实验可知，第一阶段改进 A * 算法更有助于减少变道转弯、分散路网车流和降低冲突边负载；第二阶段集群式轨迹规划算法能大幅缩短算法运算时间，克服了传统轨迹规划算法无法用于大规模车队调度的问题. 实验结果表明，提出的算法能够保障 100 辆到 500 辆 AGV 在拥有 2 250 个逻辑点和 10 064 条连通边的有向图中，无死锁、安全且高效地运行，且算法平均计算时间均在 0.001 s 级的水平. 综上所述，该算法一方面在未牺牲车队运营效率的前提下，大幅提高了面对复杂网络与大规模车队场景时的执行效率，另一方面保障了车辆行驶过程的安全性与连贯性.

值得注意的是，当前算法的主要设计思路是确保所有车辆能够无死锁的通行. 在实际场景中，可能会存在例如车辆之间错车等待、车辆反复启停等情况，进而造成额外的电量损耗或者燃油使用. 后续研究可以通过给任务动态地设置权重、设计错车优先级判定算法等方法，提升车辆行驶的连贯性，并对相关成本指标如能耗、等待时长等进行评估.

参 考 文 献：

[1]Li W, Su X. AGV path planning based on improved A * algorithm[J]. Modern Manufacturing Engineering, 2015, (10):

33 – 36.

[2]张新艳, 邹亚圣. 基于改进 A * 算法的自动导引车无碰撞路径规划[J]. 系统工程理论与实践, 2021, 41(1): 240 – 246.

Zhang Xinyan, Zou Yasheng. Collision-free path planning for automated guided vehicles based on improved A * algorithm [J]. Systems Engineering: Theory & Practice, 2021, 41(1): 240 – 246. (in Chinese)

[3]李妍峰, 高自友, 李 军. 动态网络车辆路径派送问题研究[J]. 管理科学学报, 2014, 17(8): 1 – 9.

Li Yanfeng, Gao Ziyou, Li Jun. Dynamic vehicle routing and dispatching problem[J]. Journal of Management Sciences in China, 2014, 17(8): 1 – 9. (in Chinese)

[4]周鲜成, 刘长石, 周开军, 等. 时间依赖型绿色车辆路径模型及改进蚁群算法[J]. 管理科学学报, 2019, 22(5): 57 – 68.

Zhou Xiancheng, Liu Changshi, Zhou Kaijun, et al. Improvedant colony algorithm and modelling of time-dependent green vehicle routing problem[J]. Journal of Management Sciences in China, 2019, 22(5): 57 – 68. (in Chinese)

[5]代文强, 陈 琳, 章潇月. 道路容量不确定情形下可靠应急疏散路径规划问题[J]. 系统工程理论与实践, 2022, 42(9): 2486 – 2495.

Dai Wenqiang, Chen Lin, Zhang Xiaoyue. Reliable emergency evacuation route planning under road capacity uncertainty [J]. Systems Engineering: Theory & Practice, 2022, 42(9): 2486 – 2495. (in Chinese)

[6]Li H, Bai M, Zhao Y, et al. Vehicle flow formulation for two-echelon time-constrained vehicle routing problem[J]. Journal of Management Science and Engineering, 2019, 4(2): 75 – 90.

[7]徐小峰, 姜明月, 邓忆瑞. 整合逆向物流协同配送动态路径优化问题研究[J]. 管理科学学报, 2021, 24(10): 106 – 126.

Xu Xiaofeng, Jiang Mingyue, Deng Yirui. Dynamic vehicle routing problem with simultaneous pickup and delivery in collaborative distribution under demand concurrent[J]. Journal of Management Sciences in China, 2021, 24(10): 106 – 126. (in Chinese)

[8]Fransen K J C, Van Eekelen J, Pogromsky A, et al. A dynamic path planning approach for dense, large, grid-based automated guided vehicle systems[J]. Computers & Operations Research, 2020, (123): 105046.

[9]Zhou P, Lin L, Kim K H. Anisotropic Q-learning and waiting estimation based real-time routing for automated guided vehicles at container terminals[J]. Journal of Heuristics, 2021, (29): 207 – 228.

[10]Cao Y, Yang A, Liu Y, et al. AGV dispatching and bidirectional conflict-free routing problem in automated container terminal[J]. Computers & Industrial Engineering, 2023, (184): 109611.

[11]Hu H, Yang X, Xiao S, et al. Anti-conflict AGV path planning in automated container terminals based on multi-agent reinforcement learning[J]. International Journal of Production Research, 2023, 61(1): 65 – 80.

[12]Zhong M, Yang Y, Dessouky Y, et al. Multi-AGV scheduling for conflict-free path planning in automated container terminals[J]. Computers & Industrial Engineering, 2020, (142): 106371.

[13]Hu Y, Yang H, Huang Y. Conflict-free scheduling of large-scale multi-load AGVs in material transportation network[J]. Transportation Research Part E: Logistics and Transportation Review, 2022, (158): 102623.

[14]Wu W, Xing Z, Yue H, et al. Petri-net-based deadlock detection and recovery for control of interacting equipment in automated container terminals[J]. IET Intelligent Transport Systems, 2022, 16(6): 739 – 753.

[15]He D, Ouyang B, Fan H, et al. Deadlock avoidance in closed guide-path based multi AGV systems[J]. IEEE Transactions on Automation Science and Engineering, 2023, 20(3): 2088 – 2098.

[16]Nishi T, Tanaka Y. Petri net decomposition approach for dispatching and conflict-free routing of bidirectional automated guided vehicle systems[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2012, 42(5): 1230 – 1243.

[17]Kim K H, Jeon S M, Ryu K R. Deadlock Prevention for Automated Guided Vehicles in Automated Container Terminals [M]. Container Terminals and Cargo Systems. Heidelberg: Springer Berlin, 2007: 243 – 263.

[18]Gawrilow E, Köhler E, Möhring R H, et al. Dynamic Routing of Automated Guided Vehicles in Real-time[M]. Mathemat-

ics-Key Technology for the Future. Heidelberg: Springer Berlin, 2008: 165 – 177.

[19] Gawrilow E, Klimm M, Möhring R H, et al. Conflict-free vehicle routing[J]. EURO Journal on Transportation and Logistics, 2012, 1(1): 87 – 111.

[20] Zhao Q. Module Based Studies on Large Scale AGV System in Automated Transshipment Container Terminal[D]. Singapore: National University of Singapore, 2018.

[21] Drótos M, Györgyi P, Horváth M, et al. Suboptimal and conflict-free control of a fleet of AGVs to serve online requests [J]. Computers & Industrial Engineering, 2021, (152): 106999.

[22] Zhao Y, Liu X, Wu S, et al. Spare zone based hierarchical motion coordination for multi-AGV systems[J]. Simulation Modelling Practice and Theory, 2021, (109): 102294.

[23] Brownlee A E I, Swan J, Senington R, et al. Conflict-free routing of multi-stop warehouse trucks[J]. Optimization Letters, 2020, (14): 1459 – 1470.

[24] Kim K M, Chung C H, JangY J. Deadlock Avoidance Dynamic Routing Algorithm for a Massive Bidirectional Automated Guided Vehicle System[C]//2022 Winter Simulation Conference (WSC). IEEE, 2022: 1 – 12.

[25] 王付宇, 叶春明, 王　涛, 等. 震后伤员救援车辆两阶段规划模型及算法研究[J]. 管理科学学报, 2018, 21(2): 68 – 79.
Wang Fuyu, Ye Chunming, Wang Tao, et al. Research on two stage planning model and algorithm of wounded rescue vehicle after earthquake[J]. Journal of Management Sciences in China, 2018, 21(2): 68 – 79. (in Chinese)

Deadlock-free online routing algorithm for AGV in automated container terminals

*ZHOU Chen-hao, JIAO Jun-ling, CHE A-da**

School of Management, Northwestern Polytechnical University, Xi'an 710072, China

Abstract: Unmanned heavy transport vehicles have been widely adopted in various industry scenarios, prompting a shift from individual to fleet intelligence. However, managing a large fleet of unmanned vehicles presents significant challenges, including road congestion, vehicle conflicts, and deadlock situations. This study introduces a novel two-stage online algorithm to address these issues, specifically focusing on automated guided vehicles in container terminals. The algorithm redefines the traditional static vehicle routing problem into two manageable sub-problems. In the initial stage, an improved A * algorithm is designed to identify the most efficient route for each vehicle. This improvement not only aims at optimizing traffic flow but also takes into account the dynamic nature of network workload and the likelihood of conflicts between vehicles. The subsequent stage introduces a dynamic vehicle grouping algorithm followed by a deadlock-free path planning algorithm. These algorithms are crucial for categorizing vehicles into clusters and streamlining the vehicles' movement within their respective clusters. The simulation demonstrates that the new algorithm surpasses traditional static vehicle routing methods by 20% in terms of efficiency and achieves an 80% reduction in computation time. Remarkably, it supports the real-time operation of 100 to 500 vehicles without any incidents of deadlock. The implications of these findings are significant, offering valuable insights for the future implementation of large-scale unmanned vehicle systems in complex industrial environments.

Key words: automated container terminal; AGV; vehicle route planning; deadlock-free; online algorithm