基于冲突代价 Bayesian 权重的改进 PBS 多智能体路径规划算法

钱诚泽[1],毛剑琳[1*],李睿祺[2],周雯娜[1],龚德正[1],张进宝[1]
(1. 昆明理工大学信息工程与自动化学院,云南昆明 650500;2. 昆明理工大学机电工程学院,云南昆明 650500)

摘 要: 面向多智能体路径寻找(Multi-Agent Path Finding,MAPF)问题,基于优先级搜索(Priority-Based Search,PBS)算法融合了优先级机制和冲突搜索(Conflict-Based Search,CBS)的节点拓展框架,在路径规划效率方面具有显著优势. 然而,PBS 算法中对路径代价优先的贪婪机制会导致算法在优先级树(Priority Tree,PT)拓展过程中冲突消减速度慢. 因此,本文提出基于冲突代价 Bayesian 权重的改进 PBS 算法 IPBS-ccbw(Improved Priority-Based Search with conflict cost bayesian weight). 在路径代价的基础上,引入冲突数量构造了基于路径代价和冲突数量的综合指标,并在规划拓展时对子节点的冲突代价权重进行 Bayesian 更新,以此在冲突数量与路径代价之间进行有效权衡. 进一步设计了冲突数据监测和策略性重构机制,避免算法在特定分支上陷入深度搜索陷阱. 在 Benchmark 标准测试地图上的仿真对比实验以及小规模实物实验的结果显示,IPBS-ccbw 算法在不同环境下均展现出优越的路径优化能力. 与 PBS 算法相比,在大规模密集场景中,IPBS-ccbw 算法展现出更强的冲突消减能力和更高的求解效率. 其求解时间可减少 27.3%~91.9%,在智能体数量达到最大值时,求解成功率提升幅度可达 40%~85%.

关键词: 多智能体;路径规划;贝叶斯更新;动态权重;冲突消减
基金项目: 国家自然科学基金(No.62263017)
中图分类号: TP242;TP18 **文献标识码:** A **文章编号:** 0372-2112(2025)07-2358-14
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20241074

Improved PBS Multi-Agent Path Planning Algorithm Based on Conflict Cost Bayesian Weighting

QIAN Cheng-ze[1], MAO Jian-lin[1*], LI Rui-qi[2], ZHOU Wen-na[1], GONG De-zheng[1], ZHANG Jin-bao[1]
(1. *Faculty of Information Engineering and Automation*, *Kunming University of Science and Technology*, *Kunming*, *Yunnan* 650500, *China*;
2. *Faculty of Mechanical and Electrical Engineering*, *Kunming University of Science and Technology*, *Kunming*, *Yunnan* 650500, *China*)

Abstract: In the context of multi-agent path finding (MAPF), the priority-based search (PBS) algorithm integrates a priority mechanism with the node expansion framework of conflict-based search (CBS), achieving notable efficiency in path planning. However, the greedy strategy in PBS, which prioritizes path cost, often leads to slow conflict resolution during the expansion of the priority tree (PT). To address this limitation, this paper proposes an improved PBS algorithm, improved priority-based search with conflict cost bayesian weight (IPBS-ccbw). Building upon path cost, the proposed approach incorporates conflict counts to construct a composite metric that balances path cost and conflict frequency. During the planning and expansion process, Bayesian updates are applied to the conflict cost weights of child nodes, effectively balancing conflicts and path costs. In addition, the algorithm introduces conflict monitoring and strategy reconstruction mechanisms to prevent the algorithm from falling into deep search traps. The results of simulation comparison experiments on Benchmark standard test maps as well as small-scale physical experiments show that the IPBS-ccbw algorithm exhibits superior path optimization capabilities in different environments. Compared with the PBS algorithm, the IPBS-ccbw algorithm demonstrates stronger conflict mitigation capability and higher solving efficiency in large-scale dense scenarios. The solution time can be reduced by 27.3% to 91.9%, and the solution success rate can be improved by 40% to 85% when the number of intelligences reaches the maximum.

收稿日期:2024-11-29;录用日期:2025-05-29;责任编辑:覃怀银
*通讯作者:毛剑琳

Key words: multi-agent; path planning; Bayesian update; dynamic weight; conflict reduction
Foundation Item(s): National Natural Science Foundation of China (No.62263017)

1 引言

多智能体路径寻找(Multi-Agent Path Finding, MAPF)问题[1],旨在为智能体群体中的每一个智能体找到一条从起点到终点且彼此间无冲突的路径[2],其方法在无人机[3]、物流仓储[4]、码头集装箱[5]、自动驾驶[6]及电子游戏[7]等领域均有广泛的应用,对提高多智能体系统的工作效率具有重要意义.

从理论角度,MAPF是多智能体系统对时空资源分配的非确定性多项式-hard(Nondeterministic Polynomial-hard, NP-hard)[8]问题. 在三维空间中执行任务的无人机路径规划问题中,文献[9]提出了一种双参数化深度Q网络(Double Parametrized Deep Q-Networks, DPDQN)强化学习算法,规划无人机的飞行路线和悬停位置,优化数据采集和能量补充效果. 文献[10]则从元启发式优化角度出发,设计了融合Levy飞行、自适应柯西变异与精英群遗传策略的人工兔优化算法,用于高维多约束路径优化问题. 这些研究为MAPF问题在复杂环境下的建模与求解提供了有益参考.

在二维栅格地图中的路径规划问题中,文献[11]提出了一种具备路径代价最优性的基于冲突搜索算法(Conflict-Based Search, CBS),该算法为两层算法,将MAPF问题的路径规划与冲突处理两个子问题分开求解;文献[12]提出了改进的CBS算法(Improved CBS, ICBS),根据冲突类型优先处理对搜索影响较大的冲突,并通过调整智能体路径绕过部分冲突,避免不必要的拆分操作,减少了搜索树的规模;文献[13]针对CBS中标准的分裂策略提出了不相交分割(Disjoint Splitting, DS),通过引入正约束(Positive Constraints, PC)强制某个智能体在特定时间处于指定位置,从而确保子问题不重叠,减少重复搜索. 然而,此类算法以路径最优为求解目标,导致算法的求解时间随着智能体数量增加呈指数级增长,难以应对大规模多智能体路径规划.

对此,研究人员提出了次优算法,通过牺牲部分最优性来获得求解速度和可扩展性. 文献[14]引入层级抽象与反向搜索,提出了层级协作A*(Hierarchical CA*, HCA*)算法,通过在抽象层中执行逆向搜索,能够更准确地估算距离启发函数. 同时,抽象层仅需考虑空间维度,从而为路径规划提供了一致且可接受的启发值,提高了搜索效率;文献[15]将时空A*算法改进为安全区间规划(Safe Interval Path Planning, SIPP)算法. 在动态障碍物环境中,该算法将连续的时间步合并为安全区间和碰撞区间,并在每个安全区间内进行路径搜索,而不是逐步搜索,从而缩小了搜索空间,求解速度较HCA*提高了一个数量级. 部分研究者通过在单机路径搜索过程中加入次优权重,构造了有界次优的加权动态安全区间路径规划(Weighted dynamic Safe Interval Path Planning, WdSIPP)[16]、加权重扩展安全区间路径规划(Weighted re-expansion Safe Interval Path Planning, WrSIPP)[16]以及加权重扩展安全区间路径规划与启发式树引导(Weighted re-expansion Safe Interval Path Planning with Heuristic Tree Guidance, WrSIPP-HTG)[17]等算法,进一步提高了求解速度. 这一类算法以优先级规划为基础,简化了状态空间规模,具有更强的求解效率表现[18]. 但由于此类算法的优先级是静态的,该类型算法的求解成功率和路径质量对智能体的优先级顺序较为敏感[19]. 因此,文献[20]将优先级规划与CBS框架相结合提出了优先级搜索(Priority-Based Search, PBS)算法,在高层执行深度优先搜索(Depth-First Search, DFS)并构建优先级排序,从而实现了动态优先级排序,显著提高了求解成功率. 然而,随着智能体数量的不断增加,优先级调整的复杂性也随之提高,使其成为算法的求解性能面临的重要瓶颈.

近年来,研究者们针对上述瓶颈提出了多种新思路,以改善大规模智能体环境下的求解性能. 文献[21]提出了显式估计CBS算法(Explicit Estimation CBS, EECBS),EECBS算法采用了CBS处理冲突的上层树结构,在面对智能体数量增加时,算法的求解时间呈现指数级增长;文献[22]在PBS算法的基础上提出贪婪PBS算法(Greedy Priority-Based Search, GPBS),该算法通过引入贪心策略、部分扩展和目标推理等技术,提升了搜索效率和成功率,有效缩短了求解时间,然而该策略在一定程度上牺牲了路径代价,导致生成的路径总长有所增加. 文献[23]提出了基于大邻域搜索的算法MAPF-LNS2(Multi-Agent Path Finding-Large Neighborhood Search 2),该算法在第一次运行时允许较大的次优解,通过大邻域搜索方法,反复选择部分存在冲突的智能体,并重新规划路径以减少冲突. 然而,MAPF-LNS2算法的初始解质量较低,且路径修复过程对初始解的质量有较高依赖性. 文献[24]提出了针对MAPF的懒惰约束添加搜索(Lazy Constraints Addition search for MAPF, LaCAM)算法和该算法的升级版本LaCAM*[25],LaCAM*算法利用高层配置生成与低层约束搜索的双层结构,实现了动态优先级调整和约束处理,通过惰性生成后续节点减少了搜索空间,从而能够快速生成具有完备性的可行解. 但随着智能体数量的

不断增加，搜索空间迅速扩大，算法生成解的质量呈快速下降趋势.

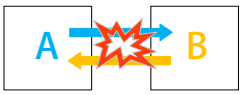
针对上述栅格地图中的路径规划算法，当智能体数量增加时存在的求解效率下降、路径质量损失较大等问题，本文提出了一种改进的PBS算法—基于冲突代价Bayesian权重的改进PBS算法IPBS-ccbw（Improved Priority-Based Search with conflict cost bayesian weight）算法. IPBS-ccbw结合了路径代价与冲突数量，通过权重的线性组合来评估路径，采用Bayesian更新方法，根据路径规划过程中的表现动态调整权重. 此外，IPBS-ccbw引入了冲突数据监测和策略性重构机制，能够在检测到深度搜索陷阱时主动进行路径重构，从而提升全局优化效果.

2 MAPF问题定义及PBS算法问题描述

2.1 MAPF问题定义

MAPF问题定义为：在无向且连通的图$G=(V,E)$上，存在一个由N_{agent}个智能体组成的群体. 每个智能体都有一个独特的起始点和目标点. 算法为每个智能体规划出无碰撞路径，即从各自的起始点安全且有效地移动到相应的目标点，确保在整个移动过程中，任意两个智能体之间的路径不会在任何时刻发生冲突[26].

MAPF问题的总体目标是为每个智能体找到一个可以无冲突执行的解决方案. 因此，在MAPF规划过程中使用冲突的概念，在匀速离散场景中，本文将常见的冲突定义列举如图1所示，主要分为两类：交换冲突和点冲突[27]. 其中，交换冲突是指两个智能体在单个时间步内发生位置交换，如图1(a)中智能体A和智能体B将在下一个时间步内发生交换冲突，从而导致两个智能体的对撞. 点冲突是指智能体被规划在同一时间步占据同一个顶点，如图1(b)中智能体A和智能体B将在下一个时间步，同时占据中间栅格发生点冲突.



(a) 交换冲突

(b) 点冲突

图1 路径规划冲突类型

MAPF问题的ILP（Integer Linear Programming）形式为[28]

$$\min \sum_{i=1}^{N_{\text{agent}}} z_i \tag{1}$$

$$0 \leqslant \sum_{i=1}^{N_{\text{agent}}} x_i[e] \leqslant 1 \quad \forall e \in E_T \tag{2}$$

$$\sum_{e \in \delta^+(v)} x_i[e] - \sum_{e \in \delta^-(v)} x_i[e] = 0 \quad i \in \{1,2,\cdots,N_{\text{agent}}\}, \forall v \in V_T, v \notin \{(s_i)_0^{\text{out}}, (g_i)_T^{\text{out}}\} \tag{3}$$

$$\sum_{e \in \delta^-((s_i)_0^{\text{out}})} x_i[e] = \sum_{e \in \delta^+((g_i)_T^{\text{out}})} x_i[e] = 1 \tag{4}$$

$$x_i[e] \in \{0,1\} \tag{5}$$

$$z_i \geqslant t \cdot x_i[e] \quad \forall t \in \{L_1, L_2, \cdots, L_{N_{\text{agent}}}, u_f\}, \forall e \in \delta^+\left(\left(g_i\right)_t^{\text{in}}\right) \tag{6}$$

其中，T为时间步t的最大值；V_T为有向网络图的节点集合，随着时间步t的增加而扩展；E_T为与节点集V_T相关的有向边集，指向不同时间步的邻近区域. 式(1)定义了优化目标函数，旨在最小化所有智能体任务完成的总时间之和z_i. 进一步，式(2)中$x_i[e]$表示智能体a_i是否在边e上移动，对于每一条边$e \in E_T$所有智能体a_i的占用决策$x_i[e]$的和必须大于等于0且小于等于1，式(2)确保每条边e在任意时刻最多只能有一个智能体通过，以避免路径冲突. 此外，式(3)和式(4)规定了流守恒约束和起点终点约束，其中$\delta^+(v)$($\delta^-(v)$)代表v的入边(出边)集合，$v \neq (s_i)_0^{\text{out}}$与$v \neq (g_i)_T^{\text{out}}$表示允许智能体在起点时只出不进，在终点时只进不出. 因此，起点终点不满足流守恒，需要从约束中排除. 式(3)确保每个智能体在中间节点的进入流量等于离开流量，从而保证路径的连贯性和完整性. 式(4)强制每个智能体都要从自己专属的起点节点$(s_i)_0^{\text{out}}$出发，并且只能选择一条终点边到达终点$(g_i)_T^{\text{out}}$，有且仅有一条出发边和一条终点边. 式(5)通过二元决策变量$x_i[e]$明确了智能体a_i是否选择某条边e作为其路径的一部分. 式(6)中L_i表示a_i从起点s_i到终点g_i的最短路径长度，u_f表示具有最小总代价解的最大完工时间上界，式(6)定义了各智能体的实际路径代价值.

2.2 PBS算法优先级树中的深度搜索陷阱

常规优先级算法（HCA*、WdSIPP、WrSIPP）中优先级的不合理配置会导致以下结果：

(1)无法找到全局解. 以图2为例，S_n、G_n分别代表智能体a_n的起点和终点. a_1、a_2与a_3智能体尝试到达各自终点，若a_3先行占据了关隘口，则无法找到全局解.

(2)影响路径质量. 在图2中，使用优先级规划仅有两种解，即$a_1 \prec a_2 \prec a_3$与$a_2 \prec a_1 \prec a_3$，它们对应的路径代价不同，前者的路径总长为13，后者为11.

这表明：除了寻找全局解外，如何合理分配优先级

以优化路径代价,并提高规划效率,亦是至关重要的.

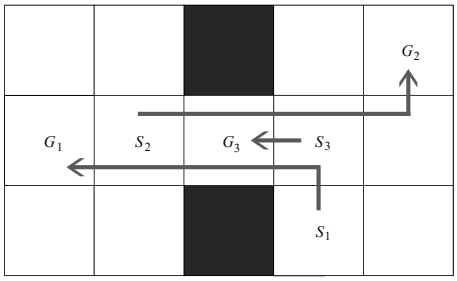


图2　优先级规划中关隘口封堵问题

PBS算法是一种针对多智能体路径规划的两级动态优先级算法. 高层通过DFS动态生成优先级排序,进而构建出对应的优先级树(Priority Tree,PT). 当遇到冲突时,PBS算法在两个子节点中贪婪地选取路径代价较低的一方进行DFS,以决定哪个智能体应被赋予更高优先权. 如果在当前分支中无法找到解决方案,则进行回溯并尝试其他分支. 算法在DFS的过程中,不断扩展并调整优先级序列,直到所有冲突都被解决[20]. 在路径规划过程中,PBS算法通过PT结构允许动态调整各智能体的优先级,以应对当前的冲突. 这种设计不仅提高了灵活性,也优化了整个多智能体系统的路径规划效率.

在PBS算法中,PT作为一个包含优先级信息的决策框架,通过生成二叉树来为每个冲突分配智能体优先级. 在高密度场景下,PBS算法的PT在特定分支上容易陷入“深度搜索陷阱”. 深度搜索陷阱是指在DFS时,算法过度深入无效分支,导致大量计算资源被浪费在无法通向有效解的子树上,且无法及时回溯或切换到其他有解分支. 其核心特征包括:无效分支被过度扩展,以及由于DFS固有策略,难以及早评估分支的潜在无效性,从而在确认路径不可行前已消耗过多资源. 深度搜索陷阱显著增加算法的复杂度. 实验数据显示:某些分支由于频繁解决冲突,导致求解时间大幅延长. 这种深度搜索陷阱会使全局路径规划陷入停滞,从而降低算法的求解效率并影响成功率.

以图3给出的场景为例,PBS算法生成了图4(a)所示的PT结构(采用实线和箭头表征节点间的父子关系,中间节点和未拓展节点已做省略处理以突显关键路径). 该PT显示,以No.17作为父节点的无效子树规模膨胀至约23万个节点. 因此,分析No.17节点所处理的冲突,如图4(b)所示,其涉及的智能体路径在地图左上角区域出现较为明显的重叠与交叉. 其中可见,智能体a_0占据了关隘口,而智能体a_{17}必须通过该关隘口到达其终点,二者的强耦合关系导致a_{17}与a_0累计发生47 313次路径冲突(占总冲突对数量的20%),此外,a_2与a_1间的冲突达12 193次. 算法在约23万次无效扩展后回溯至No.16节点,生成No.232552节点,随后经过517次扩展

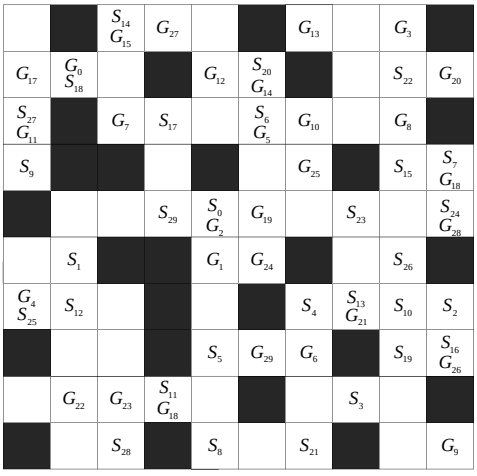
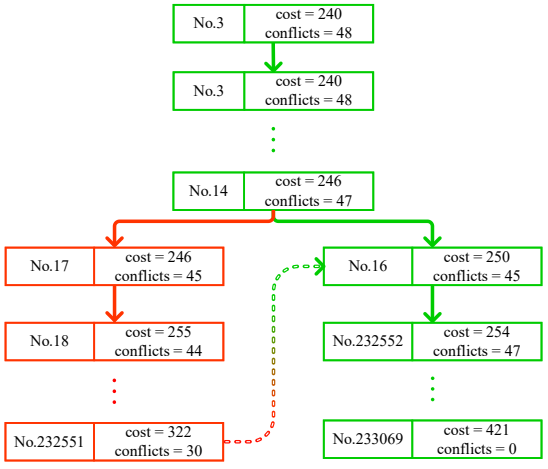
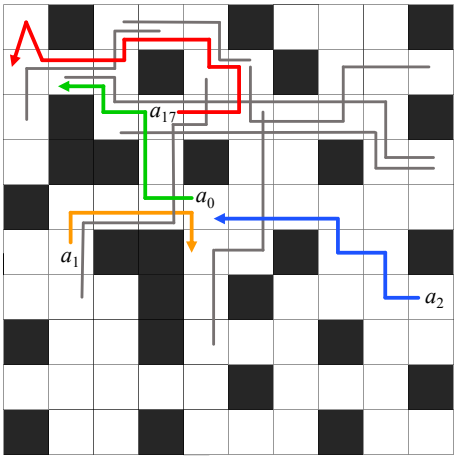


图3　密集场景测试算例



(a) PT



(b) No.17节点高冲突智能体路径

图4　PT及No.17节点涉及的高冲突智能体路径

后到达No.233069节点,才最终获得可行解. 由此可见,PBS算法的DFS策略在无效分支识别上存在局限性. 由此可见,在PT中不同分支的选择策略会影响路径代

价和冲突数量. 若分支选择策略不合理,将导致陷入深度搜索陷阱或较大的路径代价. 在PT上设计合理的选枝策略至关重要.

3 IPBS-ccbw算法

3.1 基于Bayesian更新冲突代价的策略

为了更好地探索PT节点扩展路径,更快地完成冲突消减,引入Bayesian理论. Bayesian理论是概率论与统计学中的一种重要理论框架,其核心思想在于利用新获得的观测数据对原有信念进行更新. 基于此思想,提出了一种动态冲突权重调整策略. 根据Bayesian公式:

$$P(H|E)=\frac{P(E|H)\cdot P(H)}{P(E)} \tag{7}$$

其中,$P(H|E)$为后验概率给定证据E后假设H的概率;$P(E|H)$为似然函数(假设H下证据E的概率);$P(H)$为先验概率(未考虑证据前的假设概率);$P(E)$为证据的边际概率.

结合PBS算法,在PT搜索树中,从某一节点分出的每个分支均代表在面对冲突或其他不确定性信息时所采取的不同解决方案,这些分支相当于新的"观测数据",每条分支都包含相应的路径代价变化和冲突数目变化,为当前状态提供了额外的信息. 即对各个节点当前路径冲突情况和代价的估计,与Bayesian模型通过先验概率表示初始状态类似,当生成节点时,便依据已有的路径信息和冲突情况构建初步状态. 当新的冲突信息或路径代价出现时,将新观测数据融入更新过程.

特别地,为满足实时系统对轻量化的要求[29],将冲突数量的变化建模为动态观测证据,并通过固定平滑因子与线性运算简化似然估计,从而避免了在传统Bayesian方法中高维积分或马尔可夫链蒙特卡罗方法所带来的高昂计算开销,确保将单个节点的更新时间复杂度严格控制在$O(1)$内.

根据上述描述,在PT选枝策略中构建综合节点的代价指标Cost_j,其中,j为PT节点的ID号,随PT节点扩展顺序而递增的编号,该指标定义为

$$\mathrm{Cost}_j=\mathrm{PathCost}_j+W'_{\mathrm{conf}}\times N_{\mathrm{conf}} \tag{8}$$

其中,$\mathrm{PathCost}_j$为路径代价值,即在当前扩展节点j下得到的路径代价;W'_{conf}为节点冲突代价权重,作为动态调整的多目标效用参数;N_{conf}为节点的冲突数量. 在路径冲突数量上引入冲突权重W'_{conf},可以引导算法在PT中拓展更优的节点. 由此,如何自适应确定冲突权重W'_{conf}是一个重要问题.

因此,本文引入Bayesian方法来评估和预测N_{conf}的变化趋势,并在每个拓展节点上采用Bayesian更新机制对冲突权重W'_{conf}进行参数自学习. 具体实现方法如下.

定义父节点的冲突数量为N_{parent},子节点的冲突数量为$N_{\mathrm{child}_{1,2}}$,通过启发式冲突差值$\Delta N_{\mathrm{child}_{1,2}}$替代复杂状态转移建模,以适配高动态场景的实时性约束,其中$j\in\{1,2\}$. 父节点与两个子节点的冲突变化计算式为

$$\Delta N_{\mathrm{child}_{1,2}}=N_{\mathrm{child}_{1,2}}-N_{\mathrm{parent}} \tag{9}$$

基于式(9),引入平滑因子ϵ. 为限制冲突差值的极端影响、防止子节点冲突过大的权重突变,令$\epsilon=1$,采用式(10)进行冲突变化的似然函数估计,通过线性映射简化观测模型,从而分别得到冲突增加和减少时的似然值$L_{\mathrm{conf}_{1,2}}$:

$$L_{\mathrm{conf}_{1,2}}=\begin{cases}1.0+\dfrac{\Delta N_{\mathrm{child}_{1,2}}}{N_{\mathrm{parent}}+\epsilon}, & \Delta N_{\mathrm{child}_{1,2}}>0\\ 1.0-\dfrac{|\Delta N_{\mathrm{child}_{1,2}}|}{N_{\mathrm{parent}}+\epsilon}, & \Delta N_{\mathrm{child}_{1,2}}<0\end{cases} \tag{10}$$

为防止权重因短暂冲突减少而过度下调,令最小似然值为$\eta=0.5$,并从两个子节点的似然值中选择最大值:

$$L_{\mathrm{conf}}=\max(\eta,L_{\mathrm{conf}_1},L_{\mathrm{conf}_2}) \tag{11}$$

在Bayesian的更新过程中,令$W_{\max}$为权重最大值,取先验概率$P_{\mathrm{prior}}=W_{\mathrm{conf}}/W_{\max}$为当前权重$W_{\mathrm{conf}}$的归一化表达,即其范围为$[0,1]$,简化了复杂状态转移的建模,进一步得到证据$P_{\mathrm{evid}}$为

$$P_{\mathrm{evid}}=L_{\mathrm{conf}}\cdot P_{\mathrm{prior}}+(1-L_{\mathrm{conf}})\cdot(1-P_{\mathrm{prior}}) \tag{12}$$

根据Bayesian公式,后验概率P_{post}的计算式为

$$P_{\mathrm{post}}=\frac{L_{\mathrm{conf}}\cdot P_{\mathrm{prior}}}{P_{\mathrm{evid}}} \tag{13}$$

进一步使用学习率α控制权重更新的速度,学习率选用典型的小步长更新策略$a=0.1$,决定新数据对当前权重的修正幅度,对后验概率进行尺度因子λ调整,调节权重更新的范围,限制冲突权重的上限,并更新冲突权重W_{conf},可得冲突权重更新策略为

$$W'_{\mathrm{conf}}=\alpha\cdot P_{\mathrm{post}}\cdot\lambda+(1-\alpha)\cdot W_{\mathrm{conf}} \tag{14}$$

最后,根据当前节点的路径代价$\mathrm{PathCost}_j$、冲突数量N_{conf}和冲突权重W'_{conf}计算子节点的综合代价指标Cost_j. 算法将选择综合代价指标Cost_j较低的节点进行优先拓展. 通过上述Bayesian更新方式修正冲突权重,并结合启发式规则设计,算法能统计性地结合历史信息和最新冲突观测,对搜索方向进行动态调整. 由此,在路径代价与冲突数量之间找到平衡,引导算法选择更优质的节点,减少不必要的回溯与调整,从而加快冲突消减速度. 需要说明的是,当系数λ取值过大时,算法会牺牲路径代价;当λ取值过小时,算法在冲突处理上的效率显著下降,导致求解时间大幅增加. 根据经验可知,当λ取值为4~6时,在时间效率与路径质量之间

可实现较好的平衡.

3.2 策略性重构机制

在动态冲突权重调整策略基础上,为更好地判断和规避深度搜索陷阱,对求解成功和求解失败算例中节点的冲突对做进一步统计分析,分析结果如表 1 所示. 其中,分别选取了 10 个求解成功的算例(其求解时间均接近平均值)和 10 个求解失败的算例(运行 60 s 仍未找到解,远高于平均求解时间),统计了各冲突对出现的次数,其中"Top-1"表示在统计结果中出现次数最多的冲突对,以此类推. "Top-1 占比"表示出现次数最多的冲突对在所有冲突对中的占比.

由表 1 可知,成功求解的结果统计中 Top-1 冲突对在总对数的占比较小. 而求解失败的算例中,Top-1 冲突对在总对数的占比为 21.8%~47.4%. 当冲突频次较高且呈现重复性、集中性时,算法在限定时间内无法找到可行解,这意味着算法陷入了深度搜索陷阱. 深度搜索陷阱的形成本质是在多智能体路径规划中冲突频次与优先级调整能力之间的失衡. 强耦合的智能体路径依赖会导致冲突反复累积,从而造成计算资源的指数级浪费.

表 1　算例中冲突频次对比

	Top-1/次	Top-2/次	Top-3/次	Top-1 占比/%
成功算例	1	1	1	0.17
	1	1	1	0.14
	1	1	1	0.13
	2	1	1	0.26
	1	1	1	0.13
	3	2	2	0.54
	1	1	1	0.23
	2	2	1	0.41
	1	1	1	0.20
	1	1	1	0.23
失败算例	238	44	40	32.70
	275	99	67	33.20
	240	30	26	31.60
	583	106	55	31.10
	429	377	66	29.50
	428	211	210	33.90
	2 190	291	268	47.40
	255	67	56	25.20
	176	94	92	21.80
	320	118	59	39.20

因此,基于冲突数据的监测,进一步提出了策略性重构机制. 在每次节点生成时,记录当前冲突涉及的智能体,假设智能体 a_p 和 a_q 之间的冲突表示为 $F(a_p,a_q)$. 冲突频次的累积式为

$$F(a_p,a_q)=\sum_{j\in M}\delta_{\text{conflict}}\left(a_p,a_q\middle|j\right) \tag{15}$$

其中,$\delta_{\text{conflict}}\left(a_p,a_q\middle|j\right)$ 为冲突指示函数,当智能体 a_p 和 a_q 在节点 j 发生冲突时,$\delta_{\text{conflict}}\left(a_p,a_q\middle|j\right)=1$,否则为 0. 当 $\delta_{\text{conflict}}\left(a_p,a_q\middle|j\right)=K$ 时,系统判定该分支进入了深度搜索陷阱,触发重构机制. 当 K 值的范围在 15~50 时,能够有效且快速地识别深度搜索陷阱.

当重构时,清空 PT 的所有子节点,将冲突置于 PT 的底部,从而避免回溯规模庞大的问题,确保高密度场景下的全局规划. 与此同时,设定最大重构次数,当重构次数达到预设上限后,算法继续沿剩余搜索分支深入探索,遍历所有可能的优先级关系.

3.3 IPBS-ccbw 算法伪代码

综上所述,通过在 PT 扩展过程中对子节点冲突代价权重进行 Bayesian 更新,从而做出更优的决策. 同时,采用基于冲突对统计的策略性重构机制,共同构成 IPBS-ccbw 多智能体路径规划算法,其流程图如图 5 所示.

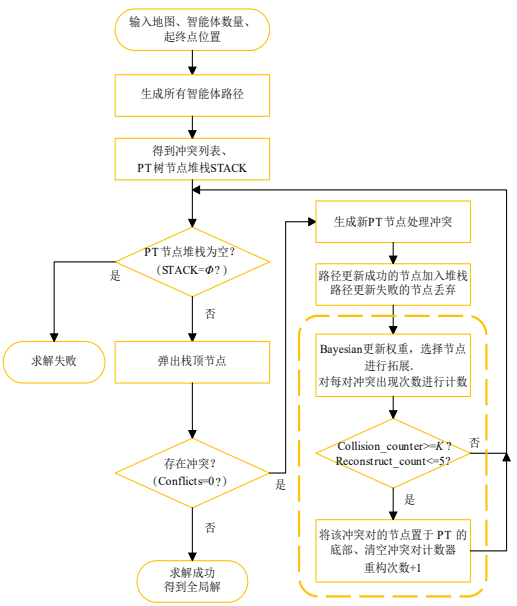


图 5　算法流程图

具体地,其伪代码如算法 1 所示.

其中,算法始于 Root 节点,包含初始优先级排序 $\prec_0$,对于(标准)PBS,这是空的优先级排序(见第 2 行). 然后,为每个智能体找到一个单独的最佳路径(见第 5 行). 当算法扩展一个 PT 节点 J 时,PBS 生成两个子节点,分别与有序对 $j\prec i$ 和 $i\prec j$ 相对应. 对于每个子 PT 节点,调用路径更新,为智能体找到一个单独的最优路径,以避免与所有较高优先级智能体的路径发生碰撞(见第 24 行). 生成一个新节点 J' 并成功规划出新路径

算法 1 High-Level Search of IPBS-ccbw

输入: MAPF instance, $\prec_0$(=∅by default)
输出: J.plan or "No Solution"

```
1   Reconstruct _count←0;
2   ≺Root ←≺0;
3   Root.plan ←∅;
4   FOREACH i ∈ [N_agent]DO
5     success ← UpdatePlan(Root, a_i);
6     IF¬success THEN
7       RETURN "No Solution";
8   Root.cost ← sum of the arrival times in Root.plan;
9   STACK ← {Root};
10  WHILE True:
11    Collision_counter.clear();
12    Solution_found ← false;
13    WHILE STACK ≠ ∅ DO
14      J ← top node in STACK;
15      STACK ← STACK\{J};
16      IF J.plan has no collision THEN
17        RETURN J.plan;
18      C ← first vertex or edge collision ⟨a_i, a_j, ⋯⟩ in J.plan;
19      FOREACH a_i involved in C DO
20        J′ ← new node;
21        J′.plan ← J.plan;
22        J′.constraints ← J.constraints W_conf;
23        ≺J′ ← ≺J W_conf W_conf;
24        success ← UpdatePlan(J′, a_i);
25        IF success THEN
26          J′.cost ← sum of the arrival times in J′.plan;
27          BayesianUpdateWeights(W_conf, W′_conf);
28          Cost_J ← PathCost_J + W′_conf · N_conf;
29      Insert new nodes J′ into STACK in non-increasing order
        of Cost_J;
30      Collision_counter[a_p, a_q]←Collision_counter[a_p, a_q] +1;
31      IF Collision_counter[a_p, a_q] = K AND Reconstruct_count<5
    THEN
32        Solution_found ← false;
33        BREAK
34    IF Solution_found = false THEN
35      IF Reconstruct _count < 5 THEN
36        Reconstruct _count ←Reconstruct _count + 1;
37        Place the node with the conflicting pair (a_p, a_q) at the
          bottom in the STACK;
38        BREAK
39      ELSE
40        CONTINUE
41    RETURN "No Solution";
42  Function UpdatePlan(J, a_i)
43  LIST ← topological sorting on partially ordered set ({i} ∪
    {j|i ≺N j},≺N);
44    FOREACH j ∈ LIST DO
45    IF j= i or∃a_k : k≺N j, a_j collides with a_k in
    J.plan THEN
46      Update J.plan by invoking a low- level search for that
        a voids colliding with all agents a_k with higher priorities
        (k ≺N j);
47      IF no path is returned by the low-level search THEN
48        RETURN false;
49    RETURN true
```

后,算法会更新该节点的路径代价(见 26 行),通过 Bayesian 调整冲突权重,再结合路径成本和加权后的冲突成本(见第 28 行),最终按照子节点的总代价从低到高进行排序,并将排序后的节点插入 PT 栈中(见第 29 行),从而使总代价较低的节点更接近栈顶,优先得到扩展.

此外,算法对当前冲突对进行计数(见第 30 行),如果某个冲突对累计次数达到预设阈值 K(即 Collision_counter≥K,见第 31 行),则标记 Solution_found 为 false,表示当前未找到解. 接下来,如果重构计数器尚未达到最大重构次数(例如小于 5),则增加重构计数器,并将有冲突对 (a_p, a_q) 的节点置于 PT 节点堆栈 STACK 的底部(见第 35~37 行);否则,如果重构计数器已经达到上限,则不再进行重构,只要 PT 栈 STACK 中还有节点待扩展,算法就会持续搜索,直到找到无冲突的路径解或穷尽所有可能的搜索分支.

3.4 各算法复杂度分析

由于算法的时间与空间复杂度决定了其在大规模场景下的可行性与效率,本节对 PBS、GPBS、LaCAM、LaCAM*及 IPBS-ccbw 五种算法进行复杂度分析.

在时间复杂度上,PBS 类算法需要分裂子节点进行冲突处理. 在最坏的情况下,需遍历 $O(b^d)$ 个子节点,其中 b 为分支因子($0<b\leqslant 2$),d 为树的深度($1\leqslant d\leqslant N_{\text{agent}}$). 同时,算法为每个智能体执行优先级排序和低层路径搜索,单次调用时间为 $O(N_{\text{agent}}\cdot T_L)$($N_{\text{agent}}$ 为智能体个数,T_L 为单次调用底层路径的搜索时间). IPBS-ccbw 算法额外维护了一个冲突计数,时间复杂度为常数项 $O(1)$. 在 GPBS 算法中引入冲突选择优化需构建 $N_{\text{agent}}\cdot N_{\text{agent}}$ 的 IC(Induced Constraints)矩阵,时间为 $O(N_{\text{agent}}^2)$. 在空间复杂度上,PBS 类算法为每个节点保存所有智能体的路径,空间为 $O(b^d\cdot N_{\text{agent}}\cdot L_{\max})$,其中 $L_{\max}$ 为最长的路径长度. IPBS-ccbw 使用冲突计数器,空间复杂度为 $O(N_{\text{agent}}^2)$. 在 GPBS 算法中引入 IC 矩阵所用的空间复杂度为 $O(N_{\text{agent}}^2)$.

在 LaCAM 类算法中,每个智能体在其当前位置上的决策为可停留或选择 d_{neigh} 个邻居方向($1\leqslant d_{\text{neigh}}\leqslant 4$),故在时间复杂度上,设分支因子 $b=d_{\text{neigh}}+1$,基于约束树来逐步扩展配置,每个时间步需要遍历所有智能体的可行移动,单个时间步处理的时间为 $O(N_{\text{agent}}\cdot d_{\text{neigh}})$,总时间为 $O((d_{\text{neigh}}+1)^{L_{\max}}\cdot N_{\text{agent}}\cdot d_{\text{neigh}})$. LaCAM*算法引入优先级队列和启发式函数,相对于每个时间步处理的时间来说,这部分时间可省略. 在空间复杂度上,LaCAM 类算法需存储所有已探索的配置,其复杂度为 $O((d_{\text{neigh}}+1)^{L_{\max}})$,约束树与临时路径数据占用 $O(N_{\text{agent}}\cdot L_{\max})$. 总空间复杂度为 $O((d_{\text{neigh}}+1)^{L_{\max}}+N_{\text{agent}}\cdot L_{\max})$.

综上所述，得出PBS、GPBS、IPBS-ccbw及LaCAM类算法的复杂度如表2所示.

表2　各算法的时间、空间复杂度

算法	时间复杂度	空间复杂度
PBS	$O(b^d \cdot N_{agent} \cdot T_L)$	$O(b^d \cdot N_{agent} \cdot L_{max})$
GPBS	$O(b^d \cdot (N_{agent} \cdot T_L + N_{agent}^2))$	$O(b^d \cdot (N_{agent} \cdot L_{max} + N_{agent}^2))$
IPBS-ccbw	$O(b^d \cdot N_{agent} \cdot T_L)$	$O(b^d \cdot N_{agent} \cdot L_{max} + N_{agent}^2)$
LaCAM LaCAM*	$O((d_{neigh}+1)^{L_{max}} \cdot N_{agent} \cdot d_{neigh})$	$O((d_{neigh}+1)^{L_{max}} + N_{agent} \cdot L_{max})$

IPBS-ccbw算法在传统路径代价的基础上，引入冲突数量构建新的综合代价指标，并在节点拓展过程中对子节点的冲突权重进行增量的Bayesian更新，充分利用冲突信息的同时，仍然保持可控的复杂度，从而在路径代价与冲突数量之间实现平衡. 避免引入高阶复杂度项，时间开销与PBS保持同阶. 相较于GPBS的约束和LaCAM*的冗余绕行，IPBS-ccbw通过动态权重的决策机制，既减少了分支选择的盲目性，又规避了路径质量的牺牲. 这一设计实现了冲突信息的高效利用，为路径规划提供了复杂度可控、解质量高、实际部署高效的解决方案. 需要说明的是，即使智能体数量相同，由于起点、终点和地图的差异，表2中的参数b、d、L_{max}仍会发生变化. 因此，目前大多数研究仍以标准算例中的计算时间作为算法性能评价的依据.

4　实验

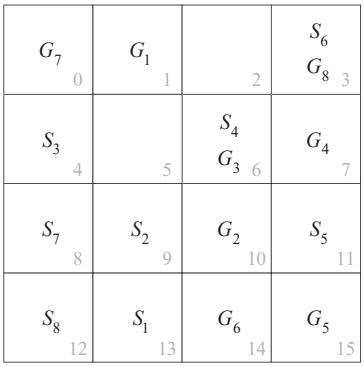
4.1　实验设置

为验证所提算法的有效性，采用不同地图进行仿真实验，采用4×4和8×8的无障碍地图作为小规模的仿真实验. 使用标准算例集合的典型算例(empty-32-32、random-32-32-20、room-32-32-4、maze-32-32-2，障碍密度分别为0%、20%、33.4%、35%)[27]进行大规模性能测试，以验证所提算法在求解成功率与求解时间优化上的有效性. 对比算法为WdSIPP[16]、WrSIPP[16]、EECBS[21]、PBS[20]、PBS+sr，其中PBS+sr算法为融合策略性重构机制的PBS算法. 各算法的参数设置如下：WdSIPP和WrSIPP所用的次优权重为1.75. EECBS算法的次优因子设置为1.2，EECBS启用了WDG启发式(WDG Heuristic)[21]、矩形推理(Rectangle Reasoning)[21]、走廊推理(Corridor Reasoning)[21]、目标推理(Target Reasoning)[21]以及绕行(By Pass)[21]等策略. IPBS-ccbw算法采用$a=0.1$、$\lambda=5$、重构阈值$K=15$以及最大重构次数=5进行仿真实验. 所有算法均在Ubuntu20.04的16 GB虚拟机上运行.

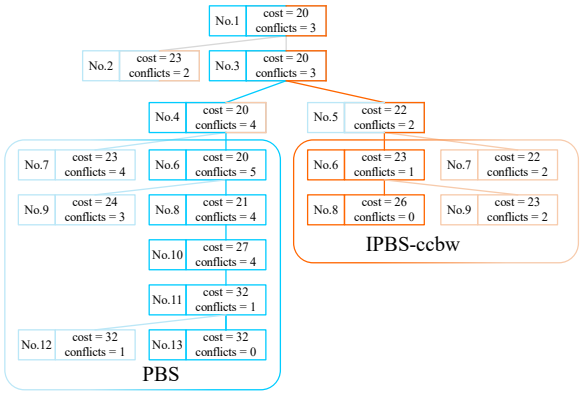
采用求解成功率、平均求解时间以及平均路径代价三个指标来衡量该方法的性能. 求解成功率、平均求解时间以及平均路径代价的计算方式分别为“成功求解次数/20”“20次任务求解时间之和/20”以及“成功求解路径代价之和/成功求解次数”. 若算例未在截止时间内完成，则其求解时间按截止时间计入计算. 若成功求解的算例少于总算例的50%，则该组路径代价数据不足以提供有效统计. 在empty-32-32场景中，由于智能体数量较多，截止时间设为120 s；其他场景的截止时间均设为60 s.

4.2　小地图算例测试及实验结果

图6(*a*)给出了一个4×4的空地图，地图上放置了8个智能体. 图6(*b*)则展示了两种算法在空地图示例中的PT节点示意图，其中，父子节点通过连接线相连，每个节点包含路径代价信息和冲突数量信息.



(*a*) 4×4的空地图



(*b*) PT

图6　冲突消减问题

图6(*b*)展示了两种算法在同一规划任务中对冲突进行分层消减的不同策略. 其中，蓝色表示PBS算法拓展的节点，橙色表示IPBS-ccbw算法拓展的节点. 2种算法求解得到的路径见表3. 针对上述地图，我们采用SONY公司生产的Toio移动机器人平台进行实物实验，以验证本文提出的IPBS-ccbw算法在实际环境中的路径可迁移性和可行性. 实物实验结果见视频链接：https://b23.tv/e9UFyfH.

表3 算法求解路径对比

智能体	PBS路径	IPBS-ccbw路径
a_1	**13→9→5→1**	13→14→13→9→5→1
a_2	9→8→8→8→8→9→10	**9→9→10→11→10**
a_3	**4→5→6**	4→5→5→6
a_4	6→10→14→13→9→10→11→7	**6→7**
a_5	11→15	11→15
a_6	3→7→11→10→14	3→2→6→10→14
a_7	8→4→0	8→4→0
a_8	12→13→9→9→10→11→7→3	**12→13→9→5→1→2→3**

注:加粗字体为最优路径.

由表3结果可知,在PBS算法中,智能体的路径存在长时间等待或绕路现象,而IPBS-ccbw算法通过有效的冲突消减策略和路径优化机制,成功避免了冗余路径.PBS算法每次优先拓展路径代价最小的节点,最终的路径代价为32,最长时间步为7步,在PT上拓展了7层节点.在首次选择节点时,IPBS-ccbw的冲突权重值为1.经计算后,选择节点代价函数值较小的No.3节点进行优先拓展.在动态权重机制下冲突的权重上升,No.4节点和No.5节点其各自的代价函数值为$Cost_4=24.2$,$Cost_5=24.1$.因此,算法选择了代价函数值更小的No.5节点进行优先拓展.最终,IPBS-ccbw仅生成了9个节点便找到解,且路径代价为26,与PBS算法相比,最长时间步数减少了6步,在PT上仅拓展了4层节点.可见,IPBS-ccbw算法能够实现高效的冲突消减,提升路径规划的成功率与整体效率.

4.3 小地图小规模任务测试结果

在小地图中,为更好地对比平均意义下的算法性能,在8×8的空地图上开展随机任务的仿真实验,设置了不同的智能体数量,每个智能体的任务随机生成.每组实验均进行20次测试,结果如表4所示.

表4 小规模实验数据统计表

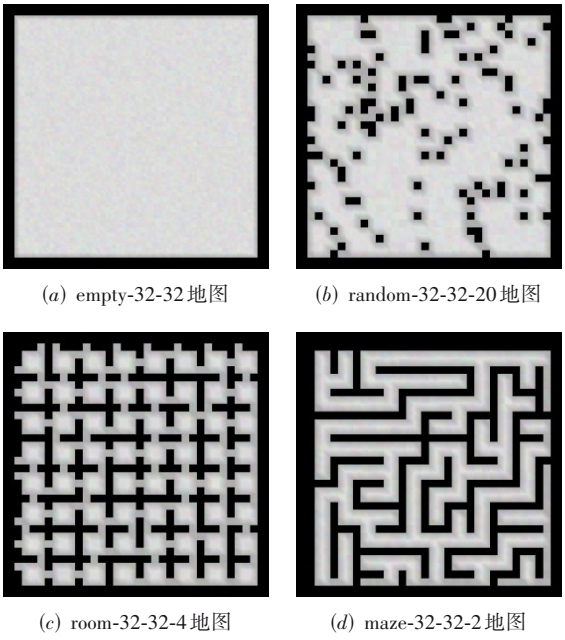
智能体数量/个	算法	平均节点数/个	平均路径代价/步	平均求解时间/s
10	PBS	4.7	**49.0**	**0.000 2**
	IPBS-ccbw	**4.3**	**49.0**	**0.000 2**
20	PBS	32.7	105.1	0.001 3
	IPBS-ccbw	**21.6**	**104.7**	**0.001 0**
30	PBS	94.6	206.7	0.007 8
	IPBS-ccbw	**57.2**	**181.1**	**0.003 6**

注:加粗字体为最优结果.

根据表4的数据分析,IPBS-ccbw算法在小地图下的多智能体路径规划任务中展现了显著的性能优势.尤其是在智能体数量逐步增加的情况下,其平均路径代价和求解时间的增长趋势较为平缓,同时平均节点数显著低于PBS算法,体现了其高效的节点扩展能力和路径优化能力.当智能体数量为30时,IPBS-ccbw算法在PT节点数、平均路径代价以及求解时间上提升幅度分别为12.4%、39.5%和53.8%.

4.4 大规模随机任务测试结果

为测试大规模高密集度场景的算法性能,在四种32×32的地图上进行仿真实验,如图7所示.采用了平均求解时间、平均路径代价和平均求解成功率等指标评价算法的求解性能.每组实验均进行20次随机任务测试.



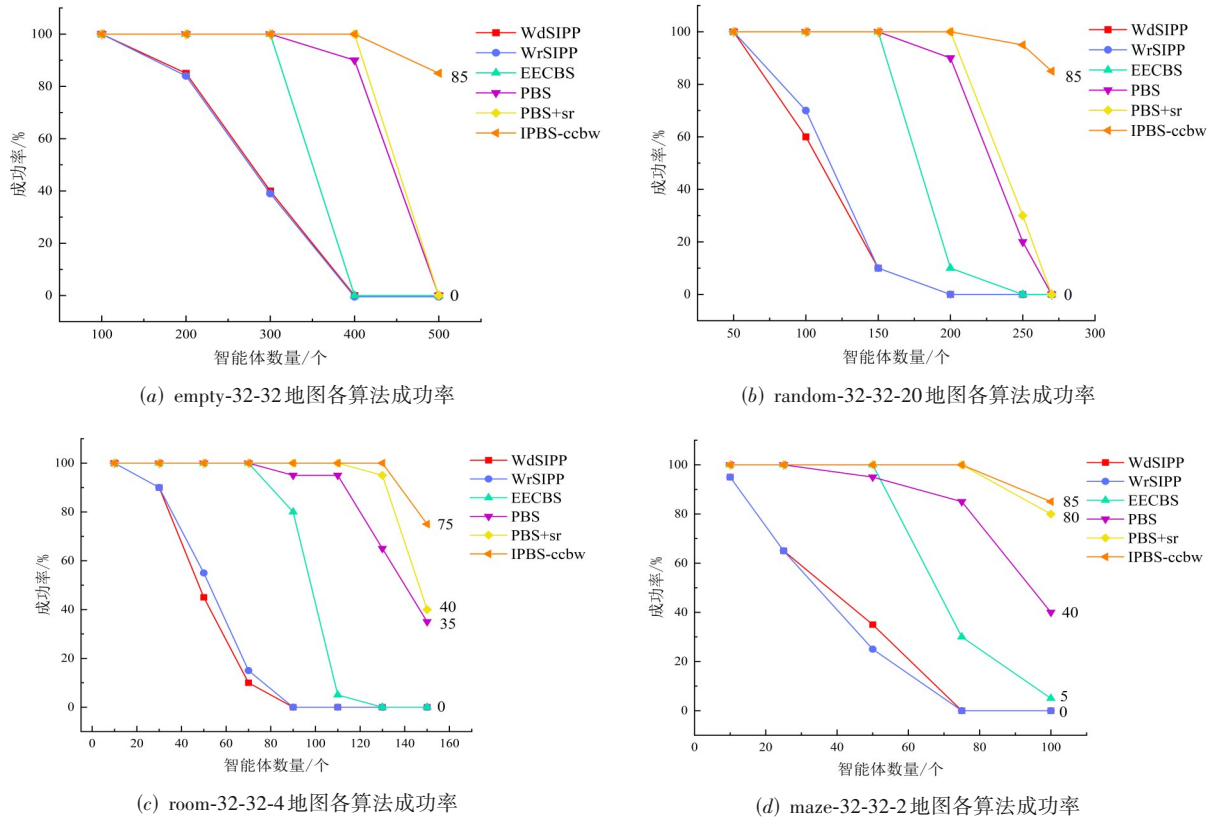
(a) empty-32-32地图 (b) random-32-32-20地图

(c) room-32-32-4地图 (d) maze-32-32-2地图

图7 随机任务实验地图

图8展示了4种地图下6种算法的求解成功率.由图8可知,6种算法的求解成功率随智能体数量增加逐渐下降.WdSIPP和WrSIPP的求解成功率下降最快.当EECBS、PBS、PBS+sr和IPBS-ccbw算法在智能体数量较少(如30~40个)时,均可实现100%的成功率,但随着智能体数量的增加,EECBS算法的冲突树节点数量呈非线性增长,导致该算法的求解能力陡然下降.与此不同,PBS算法在智能体密集场景中表现出较高的求解成功率,这是由于其优先级机制相较于EECBS算法,能有效避免许多潜在的冲突.PBS算法的复杂度较低,搜索空间的增长较为平稳,能够有效控制PT节点扩展的数量.尤其当智能体设置为60个时,PBS算法仍能保持100%的成功率.因此,当面对智能体数量变化时,PBS算法相比EECBS算法表现出更好的成功率.

PBS+sr算法加入了策略性重构机制,当智能体数量增加时,能够有效识别深度搜索陷阱,并在重构后成功求解算例.在四种地图的场景中,PBS算法最大可提



(a) empty-32-32地图各算法成功率　　(b) random-32-32-20地图各算法成功率

(c) room-32-32-4地图各算法成功率　　(d) maze-32-32-2地图各算法成功率

图8　各地图下各算法的求解成功率

升40%的成功率. 然而,PBS+sr算法依然延续了原始PBS算法中以较低路径代价节点优先拓展的贪婪策略,导致其在冲突消减效率受到影响. 尤其在处理大规模问题时,冲突消减速度显著降低,导致其求解成功率迅速下降.

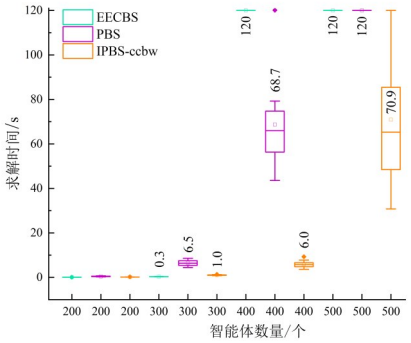
IPBS-ccbw在求解成功率上显著优于其他对比算法. 当智能体数量达到上百时,IPBS-ccbw算法依然能够保持75%以上的成功率. 在密集场景下,IPBS-ccbw算法较EECBS算法可多处理50%的智能体. 与PBS相比,IPBS-ccbw在处理大规模、多冲突任务场景中表现较好,求解成功率可提升40%~85%. 图9给出了不同算法在4类地图随机算例的求解时间.

由图9可知,不同算法在各类地图与不同智能体数量下的求解时间存在较大差异. 箱体顶部标注的数字为该组算例的平均求解时间. 箱线图中"□"代表平均求解时间所在位置. "◆"表示数据中的异常值,指远离数据集中其他观测点的数值.

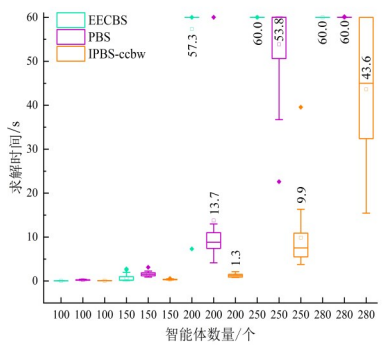
从平均求解时间的中位数及箱线图(25%和75%分位数)的分布情况来看,当智能体数量相对较少时,PBS、EECBS和IPBS-ccbw的中位求解时间较短,箱体高度极小,几乎没有上下须线,表明这三种算法在低规模场景下具有较高的稳定性和快速性.

EECBS算法在少量智能体的情况下求解时间较短,能够迅速找到较优解,在小规模问题中表现良好. 随着智能体数目的不断增加,EECBS算法的计算复杂度迅速上升,涉及的约束条件和搜索空间也随之扩大,导致求解时间呈指数级增长. 尤其是当处理room-32-32-4地图上110个智能体、random-32-32-20地图上200个智能体以及empty-32-32地图上400个智能体时,算法的性能显著下降. 在大规模多智能体系统中,这一问题愈发突出,严重制约了算法的实际应用和可扩展性.

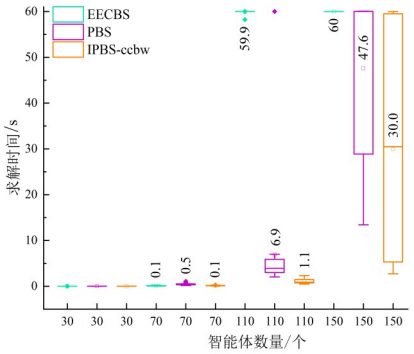
在小规模场景中,PBS算法求解时间的平均值相较于EECBS算法略长. 在中规模问题中,PBS算法相较EECBS算法表现出更强的韧性和稳定性,能够更好地应对复杂的约束与冲突,保持较高的求解能力. 然而,随着智能体数量的进一步增加,PBS算法的求解时间中位数迅速上升,箱体范围显著扩大,导致求解性能下降. 在计算过程中,PBS算法经常出现位于顶部的异常值. 在random-32-32-20地图上,对其中一个求解失败算例进行分析,当智能体数量达到200时,该算例在6 000 s内生成了9万余个节点,仍存在318个未解决的冲突,最终求解失败. 可见,PBS算法在中小规模场景中表现优越,但在大规模问题中,其求解效率和稳定性仍存在明显不足.



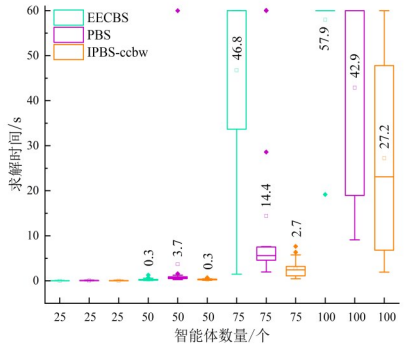
(*a*) empty-32-32地图各算法平均求解时间



(*b*) random-32-32-20地图各算法平均求解时间



(*c*) room-32-32-4地图各算法平均求解时间



(*d*) maze-32-32-2地图各算法平均求解时间

图9 四种地图下各算法的平均求解时间(图中小数为平均值)

随着智能体数量增加,IPBS-ccbw算法的平均求解时间上升速度较缓. 在random-32-32-20地图280智能体时,PBS算法全部求解失败,此时将PBS算法的求解时间记为60 s,与PBS算法相比,IPBS-ccbw算法的求解速度缩短了27.3%. 在maze-32-32-2地图上,当智能体数量为50个时,PBS算法的平均求解时间为3.7 s,IPBS-ccbw算法的平均求解时间仅为0.3 s,IPBS-ccbw算法的求解速度缩短了91.9%. 结果表明:在处理大规模智能体路径规划问题时,IPBS-ccbw算法具有更强的适应性和更高的求解效率.

表5展示了各算法在四种地图随机算例下的平均路径代价统计数据.

由表5可知,SIPP的两种算法仅在maze-32-32-2地图25个智能体场景下路径质量优于其他三种算法;随着智能体的数量增加,SIPP能提供的有效数据显著减少,表明WdSIPP与WrSIPP算法在复杂场景和大规模智能体任务中求解能力较弱. 当智能体密度增加时,这两种算法难以有效协调多智能体间的时空冲突,导致求解失败.

在empty-32-32和random-32-32-20低障碍密度场景中,总体来看,PBS算法展现出较优的路径代价,IPBS-ccbw、EECBS算法的路径代价略高于PBS算法,GPBS和LaCAM*的路径代价较长. 当智能体的数量较大且随机地图智能体的数量超过200时,IPBS-ccbw的路径代价开始优于PBS算法. 随着智能体的数量不断增大,EECBS算法和PBS算法逐渐无法求解,IPBS-ccbw、GPBS和LaCAM*依然能成功求解. 在低障碍密度场景的路径代价结果上,GPBS较IPBS-ccbw高出10.1%~56.1%,LaCAM*较IPBS-ccbw高出19.1%~69.8%,这是由于GPBS引入IC矩阵消减了冲突,但约束策略强制智能体绕行或延迟启动,解的平均路径代价显著增长. LaCAM*通过反复探索邻居配置以寻求可行解,在智能体密集场景中,局部探索缺乏全局冲突感知能力,难以动态平衡路径代价与冲突消减需求.

在room-32-32-4和maze-32-32-2高障碍密度场景中,IPBS-ccbw算法的路径代价总体优于其他算法. 以room-32-32地图70智能体场景为例,EECBS、PBS、GPBS、LaCAM*较IPBS-ccbw的路径代价分别增长了2.7%、2.5%、11.6%和23.6%. 在所有算例上,GPBS和LaCAM*的路径代价均高于其他算法. 当智能体数量较大时,IPBS-ccbw算法的路径代价优势明显,以room-32-32-4地图150个智能体为例,GPBS和LaCAM*算法相较于IPBS-ccbw算法的路径代价分别增长了26.4%、32.6%.

由上述各算法的路径代价分析可知,当面对不同类型的环境时,IPBS-ccbw具备较强的路径优化能力,能够提供更优的路径规划解决方案.

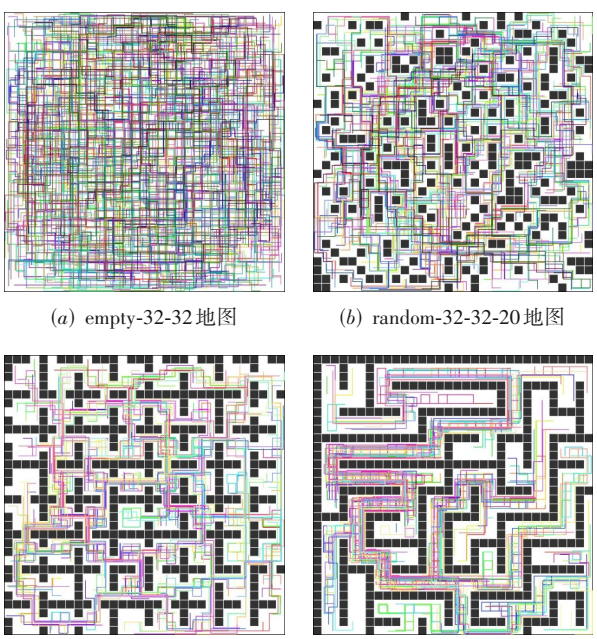
图10给出了IPBS-ccbw算法在最高求解能力下的

表 5　各地图上每个算法的平均路径代价

地图	智能体数量/个	WdSIPP	WrSIPP	EECBS	PBS	GPBS	LaCAM*	IPBS-ccbw
empty-32-32	200	4 749.2	4 749.1	4 539.8	**4 392.4**	5 386.0	6 327.4	4 555.5
	300	7 425.6	7 419.9	7 458.9	**7 142.0**	8 748.6	12 703.6	7 481.0
	400	—	—	—	**10 969.0**	13 545.0	16 301.2	11 156.5
	500	—	—	—	—	25 251.1	23 715.2	**16 176.1**
random-32-32-20	50	1 218.5	1 219.6	1 156.7	**1 140.4**	1 288.8	1 364.2	1 145.5
	100	2 511.7	2 525.6	2 459.8	**2 395.4**	2 681.7	3 145.6	2 436.3
	150	—	—	3 969.3	**3 929.4**	4 393.0	5 342.7	3 942.1
	200	—	—	—	5 804.2	6 765.7	8 325.4	**5 714.1**
	250	—	—	—	—	10 288.4	11 895.8	**8 038.0**
	270	—	—	—	—	12 140.2	13 522.5	**9 032.0**
room-32-32-4	30	835.6	834.0	829.5	819.1	862.8	929.2	**815.7**
	50	—	1 415.5	1 485.3	1 454.2	1 571.2	1 708.9	**1 445.8**
	70	—	—	2 211.1	2 207.8	2 402.5	2 662.6	**2 153.7**
	90	—	—	**2 901.6**	3 035.5	3 380.5	3 779.1	2 946.8
	110	—	—	—	3 981.4	4 526.7	4 966.8	**3 825.3**
	130	—	—	—	5 071.2	5 918.3	6 413.4	**4 924.7**
	150	—	—	—	—	7 788.6	8 170.2	**6 161.1**
maze-32-32-2	25	1 398.0	**1 391.8**	1 464.3	1 446.6	1 600.7	1 723.6	1 448.3
	50	—	—	3 251.8	3 161.5	3 681.8	3 942.2	**3 154.2**
	75	—	—	—	5 212.5	6 150.8	6 311.8	**5 205.5**
	100	—	—	—	—	9 431.0	9 131.2	**7 565.8**

注：加粗字体代表最优结果.

路径规划图，其中，在 empty-32-32 地图、random-32-32-20 地图、room-32-32-4 地图以及 maze-32-32-2 地图中能够求解的智能体数量分别为：500、270、150 和 100 个.



(*a*) empty-32-32 地图　(*b*) random-32-32-20 地图

(*c*) room-32-32-4 地图　(*d*) maze-32-32-2 地图

图 10　四种地图下各算法的平均求解时间

5　结论

本文提出了一种具有参数自学习机制的多智能体路径规划 IPBS-ccbw 算法. 该算法在路径代价的基础上引入冲突数量，构造了一个基于路径代价和冲突数量的综合指标，并在路径规划拓展过程中，通过 Bayesian 更新对子节点的冲突代价权重，以实现冲突数量与路径代价之间的有效权衡. 此外，策略性重构机制的引入有效判断和避免算法陷入 DFS 陷阱，从而进一步提升了路径规划的整体效率. 实验结果表明：IPBS-ccbw 算法展现出了较强的适应性，特别在高障碍密度场景下，能够有效处理更多的路径规划任务. 当智能体的数量相同时，相较于 WdSIPP[16]、WrSIPP[16]、EECBS[21]、PBS[20]、GPBS[22] 以及 LaCAM*[25]，算法 IPBS-ccbw 算法能够保证较高的路径质量，较 PBS[20] 和 EECBS[21] 算法显著缩短了求解时间，通过在多种地图类型上的对比实验，验证了该算法在复杂环境中的求解速度以及路径质量的优势.

本文的研究可为密集复杂场景下的多智能体路径规划问题提供一种新的基于参数学习的规划思路和方法，未来的研究将进一步探讨和结合更多的机器学习方法，以提升算法对复杂规划问题的求解能力.

参考文献

[1] SEMIZ F, YORGANCı M A, POLAT F. Solving an industry-inspired generalization of lifelong MAPF problem including multiple delivery locations[J]. Advanced Engineering Informatics, 2023, 57: 102026.

[2] YU J J, LAVALLE S M. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics[J]. IEEE Transactions on Robotics, 2016, 32(5): 1163-1177.

[3] HO F, SALTA A, GERALDES R, et al. Multi-agent path finding for UAV traffic management[C]//Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. New York: ACM, 2019: 131-139.

[4] ZHANG Y L, FONTAINE M C, BHATT V, et al. Multi-robot coordination and layout design for automated warehousing (extended abstract)[J]. Proceedings of the International Symposium on Combinatorial Search, 2024, 17: 305-306.

[5] LI J Y, SUN K X, MA H, et al. Moving agents in formation in congested environments[C]//Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. New York: ACM, 2020: 726-734.

[6] WU M D, YAN W Y, HASAN H, et al. A review of multi-agent path finding algorithms[C]//Proceedings of the 11th International Conference on Information Systems and Computing Technology. Piscataway: IEEE, 2023: 69-73.

[7] MA H, YANG J X, COHEN L, et al. Feasibility study: Moving non-homogeneous teams in congested video game environments[J]. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2017, 13(1): 270-272.

[8] YU J J, LAVALLE S M. Structure and intractability of optimal multi-robot path planning on graphs[C]//Proceedings of the 27th AAAI Conference on Artificial Intelligence. New York: ACM, 2013: 1443-1449.

[9] 高思华, 李军辉, 李建伏, 等. 面向公平性数据采集和能量补充的无人机路径规划算法研究[J]. 电子学报, 2024, 52(11): 3699-3710.
GAO S H, LI J H, LI J F, et al. Research on UAV path planning algorithm for fairness data collection and energy supplement[J]. Acta Electronica Sinica, 2024, 52(11): 3699-3710. (in Chinese)

[10] 王文涛, 叶晨, 田军. 基于多策略改进人工兔优化算法的三维无人机路径规划方法[J]. 电子学报, 2024, 52(11): 3780-3797.
WANG W T, YE C, TIAN J. A 3D UAV path planning method based on multi-strategy improved artificial rabbit optimization algorithm[J]. Acta Electronica Sinica, 2024, 52(11): 3780-3797. (in Chinese)

[11] SHARON G, STERN R, FELNER A, et al. Conflict-based search for optimal multi-agent pathfinding[J]. Artificial Intelligence, 2015, 219: 40-66.

[12] BOYARSKI E, FELNER A, STERN R, et al. ICBS: The improved conflict-based search algorithm for multi-agent pathfinding[J]. Proceedings of the International Symposium on Combinatorial Search, 2015, 6(1): 223-225.

[13] LI J Y, HARABOR D, STUCKEY P J, et al. Disjoint splitting for multi-agent path finding with conflict-based search[J]. Proceedings of the International Conference on Automated Planning and Scheduling, 2019, 29: 279-283.

[14] SILVER D. Cooperative pathfinding[J]. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2005, 1(1): 117-122.

[15] PHILLIPS M, LIKHACHEV M. SIPP: Safe interval path planning for dynamic environments[C]//2011 IEEE International Conference on Robotics and Automation. Piscataway: IEEE, 2011: 5628-5635.

[16] YAKOVLEV K, ANDREYCHUK A, STERN R. Revisiting bounded-suboptimal safe interval path planning[J]. Proceedings of the International Conference on Automated Planning and Scheduling, 2020, 30: 300-304.

[17] RYBECKY T, KULICH M, ANDREYCHUK A, et al. Towards narrowing the search in bounded-suboptimal safe interval path planning[J]. Proceedings of the International Symposium on Combinatorial Search, 2021, 12(1): 136-140.

[18] REN Z Q, RATHINAM S, LIKHACHEV M, et al. Multi-objective safe-interval path planning with dynamic obstacles[J]. IEEE Robotics and Automation Letters, 2022, 7(3): 8154-8161.

[19] ČÁP M, NOVÁK P, KLEINER A, et al. Prioritized planning algorithms for trajectory coordination of multiple mobile robots[J]. IEEE Transactions on Automation Science and Engineering, 2015, 12(3): 835-849.

[20] MA H, HARABOR D, STUCKEY P J, et al. Searching with consistent prioritization for multi-agent path finding[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2019, 33(1): 7643-7650.

[21] LI J Y, RUML W, KOENIG S. EECBS: A bounded-suboptimal search for multi-agent path finding[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2021, 35(14): 12353-12362.

[22] CHAN S H, STERN R, FELNER A, et al. Greedy priority-

based search for suboptimal multi-agent path finding[J]. Proceedings of the International Symposium on Combinatorial Search, 2023, 16(1): 11-19.

[23] LI J Y, CHEN Z, HARABOR D, et al. MAPF-LNS2: Fast repairing for multi-agent path finding via large neighborhood search[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2022, 36(9): 10256-10265.

[24] OKUMURA K. LaCAM: Search-based algorithm for quick multi-agent pathfinding[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2023, 37(10): 11655-11662.

[25] OKUMURA K. Improving lacam for scalable eventually optimal multi-agent pathfinding[EB/OL]. (2023-05-05) [2024-11-10]. https://arxiv.org/abs/2305.03632.

[26] KADURI O, BOYARSKI E, STERN R. Experimental evaluation of classical multi agent path finding algorithms[J]. Proceedings of the International Symposium on Combinatorial Search, 2021, 12(1): 126-130.

[27] STERN R, STURTEVANT N, FELNER A, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks[J]. Proceedings of the International Symposium on Combinatorial Search, 2019, 10(1): 151-158.

[28] MA H. Target Assignment and Path Planning for Navigation Tasks with Teams of Agents[D]. Los Angeles: University of Southern California, 2020.

[29] KOCHENDERFER M J. Decision Making under Uncertainty: Theory and Application[M]. Cambridge, Mass: The MIT Press, 2015.

作者简介

钱诚泽　男,1999年12月生,云南昭通人.现为昆明理工大学信息工程与自动化学院硕士研究生.主要研究方向为多智能体路径规划.
E-mail: AzerovO@outlook.com

周雯娜　女,2001年8月生,湖南永州人.现为昆明理工大学信息工程与自动化学院硕士研究生.主要研究方向为多智能体路径规划.
E-mail: 2929399036@qq.com

毛剑琳　女,1976年5月生,广西桂林人.现为昆明理工大学教授、博士生导师.研究方向为多机器人协同调度与优化、无线传感器网络资源分配、智能优化算法研究等.
E-mail: 1318524654@qq.com

龚德正　男,2001年9月生,湖南岳阳人.现为昆明理工大学信息工程与自动化学院硕士研究生.主要研究方向为多智能体路径规划.
E-mail: 2948151791@qq.com

李睿祺　男,1996年7月生,云南昆明人.现为昆明理工大学机电工程学院博士研究生.主要研究方向为智能调度.
E-mail: 20231103005@stu.kust.edu.cn

张进宝　男,2002年7月生,湖南永州人.现为昆明理工大学信息工程与自动化学院硕士研究生.主要研究方向为多智能体路径规划.
E-mail: 941138436@qq.com