自动化学报
ACTA AUTOMATICA SINICA

基于讨价还价博弈机制的B-IHCA* 多机器人路径规划算法

张凯翔 毛剑琳 向凤红 宣志玮

B-IHCA*, a Bargaining Game Based Multi-agent Path Finding Algorithm

ZHANG Kai-Xiang, MAO Jian-Lin, XIANG Feng-Hong, XUAN Zhi-Wei

在线阅读 View online: https://doi.org/10.16383/j.aas.c220065

基于讨价还价博弈机制的 B-IHCA* 多机器人路径规划算法

张凯翔 [1] 毛剑琳 [2] 向凤红 [2] 宣志玮 [2]

摘 要 针对密集场景中大规模冲突导致多机器人路径规划 (Multi-agent path finding, MAPF) 成功率低的问题, 引入讨价还价博弈机制并以层级协作 A* (Hierarchical cooperative A*, HCA*) 算法为内核, 提出一种基于讨价还价博弈机制的改进层级协作 A* (Bargaining game based improving HCA*, B-IHCA*) 算法. 首先, 在 HCA* 算法基础上, 对导致路径无解的冲突双方或多方进行讨价还价博弈. 由高优先级机器人先出价, 当低优先级机器人在该条件下无法求解时, 则其将不接受该出价, 并通过降约束求解方式进行还价. 再由其他冲突方对此做进一步还价, 直至各冲突方都能协调得到可接受的路径方案. 其次, 为避免原始 HCA* 算法由于高优先级的阻碍陷于过长或反复无效搜索状态, 在底层 A* 搜索环节加入了熔断机制. 通过熔断机制与讨价还价博弈相配合可在提升路径求解成功率的同时兼顾路径代价. 研究结果表明, 所提算法在密集场景大规模机器人路径规划问题上较现有算法求解成功率更高、求解时间更短, 路径代价得到改善, 验证了算法的有效性.

关键词 多机器人, 路径规划, 讨价还价博弈, 解耦, 协作

引用格式 张凯翔, 毛剑琳, 向凤红, 宣志玮. 基于讨价还价博弈机制的 B-IHCA* 多机器人路径规划算法. 自动化学报, 2023, **49**(7): 1483−1497

DOI 10.16383/j.aas.c220065

B-IHCA*, a Bargaining Game Based Multi-agent Path Finding Algorithm

ZHANG Kai-Xiang[1] MAO Jian-Lin[2] XIANG Feng-Hong[2] XUAN Zhi-Wei[2]

Abstract Large-scale conflict of paths is a reason which can hugely reduce the success rate for multi-agent path finding (MAPF) in dense scenarios. For this problem, a bargaining game based improving hierarchical cooperation A* (B-IHCA*) algorithm is proposed by introducing bargaining game mechanism and taking hierarchical cooperation A* (HCA*) algorithm as the core. Firstly, based on the HCA* algorithm, the bargaining game is performed between the two or more parties that leads to conflicts and the unsolved state. The high-priority agent finds a path and bids first. If the low-priority agent cannot get a path with constraint of the bid, it will does not accept the bid and counter-offer another path by reducing the constraint. And then the other conflicting parties will make further counter-offers until all conflicting parties can coordinate to obtain an acceptable path scheme. Secondly, in order to avoid the state of too long or repeated invalid search of HCA* algorithm due to the obstruction of high-priority agent, a fusing mechanism is added to its bottom A* algorithm. Accordingly, by the cooperation of bargaining game and fusing mechanism, the success rate of the paths finding can be improved while taking into account the path costs. The results show that, compared with the existing algorithms, the proposed algorithm has higher success rate, shorter time consumption and ameliorative path costs for large-scale MAPF problem in dense scenarios, which verifies the effectiveness of the algorithm.

Key words Multi-agent, path finding, bargaining game, decoupling, cooperation

Citation Zhang Kai-Xiang, Mao Jian-Lin, Xiang Feng-Hong, Xuan Zhi-Wei. B-IHCA*, a bargaining game based multi-agent path finding algorithm. *Acta Automatica Sinica*, 2023, **49**(7): 1483−1497

收稿日期 2022-01-25 录用日期 2022-09-09
Manuscript received January 25, 2022; accepted September 9, 2022
云南省重点研发计划项目 (202002AC080001), 国家自然科学基金 (62263017) 资助
Supported by Provincial Major Research Program of Yunnan (202002AC080001) and National Natural Science Foundation of China (62263017)
本文责任编委 程龙
Recommended by Associate Editor CHENG Long
1. 昆明理工大学机电工程学院 昆明 650500 2. 昆明理工大学信息工程与自动化学院 昆明 650500
1. Faculty of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming 650500 2. Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500

当前, 智能机器人的集群协作技术越来越受到国内外学者的广泛关注[1]. 多机器人路径规划 (Multi-agent path finding, MAPF) 作为集群协作的一项关键性问题[2], 其主要任务是为群体中的每一个机器人找到一条从起点到终点且彼此间无冲突的路径[3]. MAPF 对提升集群协作的效率、质量和安全性有直接作用, 因此在仓储系统、交通控制、机场运营、军资运输等各领域都展现出了巨大的研究潜力和价值[4−5].

MAPF 问题是典型的 NP-Hard 问题[6], 其解空

间规模与机器人数 N_{robot}、时间步 T_{step} 均呈指数关系. 当前, 针对该问题的求解方法按扩展状态的组合形式主要可分为耦合型算法和解耦型算法[7]. 其中, 耦合型算法在求解思路上将所有机器人视为一个整体问题, 扩展时考虑到每个机器人的潜在状态组合, 并在该状态组合中搜寻可行路径方案. 耦合型算法具有完备性和最优性的特点, 但其求解效率通常较低[8]. 如单机器人路径规划中使用广泛的 A* 算法[9], 若将其直接扩展到 MAPF 问题, 将由于搜索空间随机器人数量和时间步的指数级增长而出现搜索空间爆炸现象, 从而在很大程度上削弱了算法的求解能力. 为改善搜索空间爆炸对耦合型算法造成的求解效率低、占用内存量大等突出问题[10], Standley[11] 提出算子分解 (Operator decomposition, OD) 方法, 通过引入单机扩展的中间态, 对 MAPF 问题的搜索过程进行引导和剪枝, 在一定程度上缩小了搜索空间规模. Sharon 等[12] 提出增值树搜索算法 (Increasing cost tree search, ICTS), 以递增的路径代价为引导标准, 对搜索过程进行剪枝, 提升了算法的求解效率. 但耦合型算法始终受限于搜索规模指数级增长的固有问题, 随着机器人数量的进一步增加, 其在有限时间内的求解成功率将在很大程度上被削弱.

解耦型算法应用于求解更多机器人数量的问题场景, 该类型算法将 MAPF 问题在某种程度上降维到了多个独立机器人的单机规划问题[13], 在获得的单机器人路径基础上再进行路径间的冲突判断与处理. 如 Sharon 等[14] 提出的基于冲突搜索算法 (Conflict-cased search, CBS), 将 MAPF 问题分为冲突处理和单机路径规划两个子问题, 并构造了一个双层框架分别在上下层对所属子问题进行求解[15]. 在此基础上, Li 等[16] 针对 CBS 算法在顶层冲突树扩展时可能出现的子节点重复问题, 提出不相交分割 (Disjoint splitting) 方法对冲突树进行剪枝, 从而避免了大量重复搜索工作, 进一步提升了搜索效率. CBS 算法具备最优性特点, 并提供了一个清晰的问题求解框架用于后续的研究和改进工作[17−18]. 然而, 当存在冲突时, 该类型算法在上层仅对冲突的双方作单独的带约束重新规划, 并没有考虑到其他机器人对两者的影响. 当在窄道环境或问题场景高度耦合时, 容易陷入解决完一个冲突又生成新冲突的状态, 由此反复循环, 从而导致该方法难以得到可行解.

解耦型算法的另一类模式是在单机规划的同时即考虑冲突的避免, 而不对两者做分层处理. Silver[19] 提出协作 A* (Cooperative A*, CA*) 算法, 按优先级顺序[20] 依次为每个机器人进行单机路径规划, 并将前面规划的高优先级路径信息作为约束加入到后面机器人的规划中. 在此基础上, 为获得更有效的单机规划启发函数, Silver[19] 进一步提出了层级协作 A* (Hierarchical CA*, HCA*) 算法, 在增加的启发函数计算层中执行反向搜索操作以获得更为准确的距离启发值, 提高了搜索效率. 由于在单机规划的同时就考虑对其他机器人的冲突避免, 该类型算法通常具有更高的求解效率, 但无法保证求解结果的最优性[21], 且求解成功率和路径质量对机器人的优先级顺序较为敏感[22]. Li 等[23] 通过建立无人机群的耦合度矩阵, 再根据耦合度指标的高低进行各机器人的优先级分配, 提高了算法在多机耦合场景中的解耦能力与稳定性. Wu 等[24] 在分布式系统中分析了路径的同调类 (Homology classes) 数量, 并以备选路径的多少来安排机器人的优先级顺序, 起到了平衡路径总耗费与最大完工时间的作用, 但求解过程未涉及终点封堵的类型, 很可能因为前面高优先级机器人在终点位置阻挡了低优先级机器人的通过而使算法求解失败. Yakovlev 等[25] 提出加权的安全区间路径规划算法 (Weighted safe-interval path planning, WdSIPP), 通过引入次优权重系数, 提高了算法在大规模多机问题中的解耦效率. 然而, 在路径高耦合的密集场景中, 机器人间的拥塞与封堵问题更加凸显, 固定的排序规则往往难以适应问题场景的动态变化.

近年来, 在多机协作问题中引入了博弈机制, 为 MAPF 问题的求解扩展了新的思路. 李珣等[26] 针对纺织车间中多搬运机器人的任务分配问题, 引入博弈论中的 Nash 均衡理论, 在定义智能体对任务的偏好关系基础上, 通过多轮博弈和迭代过程, 建立了全局最优的 Nash 稳定分区. Wu 等[27] 针对多智能体的动态任务分配问题, 基于势博弈法, 设计了一种新的动态环境下分布式多智能体任务分配框架, 并保证了任意 Nash 均衡解的次优性在 50% 以内. 郑延斌等[28] 在两阶段的路径规划算法中, 采用博弈论构建多智能体之间的动态避障模型, 并利用虚拟行动法来解决多个机器人在冲突位点的博弈问题及多 Nash 均衡的选择问题, 减小了路径总代价并提高了算法的收敛速度.

本文针对密集场景中的高耦合 MAPF 问题, 采用降约束求解策略并结合讨价还价博弈机制, 动态调整冲突机器人间的优先级顺序以提高问题求解成功率. 针对 HCA* 算法在求解过程中受到封堵的机器人, 降低部分约束条件获得一条降约束路径并将其作为对高优先级冲突方的还价. 在后续几轮的讨价还价过程中, 以降约束路径作为基准价, 进而

使得与之冲突的高优先级机器人妥协并变更路径. 由此间接调整冲突机器人间的优先级顺序, 并使得整体路径冲突逐渐消解, 获得可行解. 进一步, 在 HCA* 底层中加入熔断机制, 避免原始 HCA* 在单机搜索过程中陷于过长搜索或死循环状态. 通过熔断机制与讨价还价博弈机制相结合, 对规划过程中的过长路径进行熔断和重新规划, 以使路径总代价指标获得改善. 通过多种算例仿真实验对算法的性能进行验证.

1　问题描述

1.1　MAPF 问题定义

MAPF 问题定义为: 在无向连通图 $G=(V, E)$ 中, 存在由 N_{robot} 个机器人组成的机器人群体 A, 如图 1 所示, 每个机器人 $\{a_i \in A | i=1, 2, \cdots, N_{\text{robot}}\}$ 均有各自的起点 s_i 和终点 g_i. 系统需为每个机器人 a_i 规划出一条从起点到终点的路径 p_i, 且任意机器人 a_i 与 a_j 的路径 p_i 与 p_j 间不能发生冲突[29]. 在此基础上, 令所有机器人的路径代价总和最小[30] 是本文的优化目标.

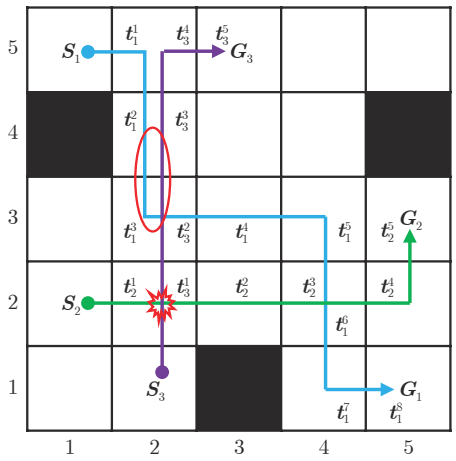


图 1　MAPF 问题描述

Fig. 1　MAPF problem description

考虑匀速离散场景, MAPF 问题的冲突类型主要分为两类: 位置冲突和相向冲突[31]. 其中, 位置冲突指两个机器人在相同的时刻占据相同的位置, 如图 1 中 a_2 与 a_3 将在 $t=1$ 时刻共同占据地图点 $v=(2, 2)$ 的位置. 相向冲突指两个机器人在前后时间步内发生相向移动, 如点 $v_i=(2, 3)$ 与点 $v_j=(2, 4)$ 构成的边 $e=(v_i, v_j)$ 上, a_1 与 a_3 将在 $t=[2, 3]$ 时间段内发生相向移动, 从而造成两者的对撞. 由此, MAPF 问题的数学描述如下[32]:

优化目标

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

约束条件

$$\sum_{e\in\delta^-((s_i)_0^{\text{out}})} x_i(e) = \sum_{e\in\delta^+((g_i)_T^{\text{out}})} x_i(e) = 1 \tag{2}$$

$$0 \le \sum_{i=1}^{N_{\text{robot}}} x_i(e) \le 1,\ \forall e \in E_T \tag{3}$$

$$\sum_{e\in\delta^+(v)} x_i(e) - \sum_{e\in\delta^-(v)} x_i(e) = 0$$
$$\forall v \in V_T,\ v \ne (s_i)_0^{\text{out}},\ v \ne (g_i)_T^{\text{out}} \tag{4}$$

$$x_i(e) \in \{0, 1\},\quad i=1, 2, \cdots, N_{\text{robot}}, \forall e \in E_T \tag{5}$$

$$z_i \ge t \times x_i(e),\ \forall t \in \{L_i, \cdots, u_f\}$$
$$\forall e \in \delta^+\left((g_i)_t^{\text{in}}\right) \tag{6}$$

式中, $x_i(e)$ 为决策变量, 表示 a_i 是否经过边 e. $\delta^+(v)$ ($\delta^-(v)$) 表示流入 (流出) 节点 v 的边集. T 为时间步 t 的最大值. V_T 为有向网络图的节点集, 由地图点集 V 随时间步 t 不断扩展所构成, E_T 为 V_T 对应的有向图边集, 其指向为下一个时间步的地图邻域. L_i 表示实际地图中 a_i 从起点 s_i 到终点 g_i 的最短路径长度, u_f 表示任意具有最小总代价解的最大完工时间上界. 由该模型可知, MAPF 问题构造了带时间维的多物网络流, 其网络规模以实际地图为基础, 并随着时间步 t 不断扩增. 其中, 式 (2) 为各机器人的起点和终点约束; 式 (3) 为容量约束, 式 (4) 为流守恒约束, 两者共同构成机器人在运动过程中的位置冲突与相向冲突约束; 式 (6) 定义了各机器人的实际路径代价值.

1.2　密集场景中的占位与拥塞问题

根据机器人到达终点后是否始终停留在终点位置, MAPF 问题分为 One-shot 和 Long-term 两种类型[32]. 本文针对典型的 One-shot 类型开展研究, 其在路径规划过程将发生占位问题, 即部分机器人到达终点后将独占该位置而持续影响其他机器人的通行, 从而使得 MAPF 问题的场景复杂度随着过程的推进而动态增加. 如图 2 所示, 随着 a_2 和 a_3 陆续到达指定位置, 地图将衍生出只容许单个机器人通过的关隘地形. 而在更为密集的环境中, 地图的动态演变过程中将出现各类复杂地形甚至全封闭地形.

在地图的可通行区域被动态缩减的同时, 路径间的耦合度也在急剧升高. 这将在很大程度上增加

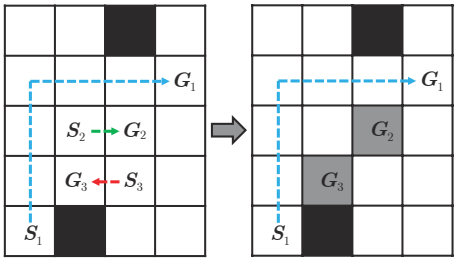
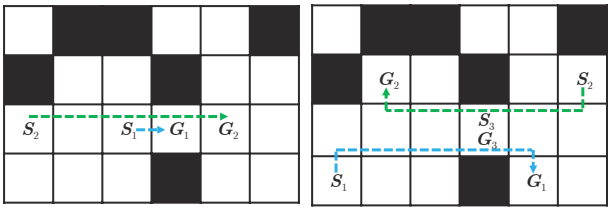


图 2 One-shot 类型 MAPF 问题的终点占位现象

Fig. 2 Goal occupations in one-shot MAPF problem

拥塞发生的概率. 拥塞问题的出现将导致算法的求解效率和求解成功率显著下降, 并直接影响求解方案的路径代价.

如图 3 的案例中, 均存在一个拥塞点致使各机器人的路径产生耦合. 图 3(a) 中, 若按 $(a_1 \prec a_2)$ 的优先级顺序, 即先 1 号后 2 号, 则 a_2 将由于 a_1 移动至关隘内部并占据通道而导致无法求解. 反之, 若采用 $(a_2 \prec a_1)$ 顺序则可完成求解. 图 3(b) 中, 若优先级顺序为 $(a_1 \prec a_2 \prec a_3)$, 则通过 HCA* 算法规划所得路径方案的总代价为 24; 若优先级顺序为 $(a_2 \prec a_1 \prec a_3)$, 则 HCA* 所得总代价为 21.



(a) 类型 A (a) Type A (b) 类型 B (b) Type B

图 3 拥塞问题示例

Fig. 3 Example of congestion problem

综上所述, 在 MAPF 问题中, 场景复杂度和路径耦合度随过程的推进而动态增加. 尤其在机器人数量更多的密集场景中, 其对算法的拥塞处理能力和冲突处理效率提出了更高的要求. 优先级顺序不再仅仅是影响现有算法的求解效率, 而是更关系着问题能否顺利求解. 因此, 在现有算法的基础上, 如何动态调整优先级顺序以提高密集场景中 MAPF 问题的求解成功率和路径工时质量是本文的研究目标.

2 基于讨价还价博弈机制的 B-IHCA* 算法

2.1 HCA* 算法

HCA* 是基于 A* 搜索的解耦型 MAPF 算法, 由 CA* 和反向回溯 A* (Reverse resumable A*, RRA*) 算法构成[19]. 其中, CA* 在求解思路上将 MAPF 问题分解为有前后联系的单机路径规划问题, 按优先级顺序依次对机器人 $a_1, a_2, \cdots, a_{N_{\text{robot}}}$ 做单机路径规划, 并把前面规划的机器人路径信息作为障碍约束添加到后面的机器人规划当中. 即在单机 A* 搜索基础上额外设立了一个约束表, 当完成第 i 个机器人 a_i 的路径规划后, 便将该路径信息 p_i 放入约束表并与前面已有的列表项共同组成下一个机器人 a_{i+1} 的带时限障碍物约束, 按顺序依次执行上述操作, 直至完成全部机器人的规划方案.

CA* 算法在单机 A* 搜索过程中使用曼哈顿距离作为启发函数, 然而该方式在复杂环境中的表现较差. 因此, Silver[19] 在 CA* 框架下增设启发函数计算层构造 HCA* 算法, 并使用 RRA* 从目标点到当前点进行反向 A* 搜索, 以获得当前点到目标点的实际距离, 将该距离作为启发函数, 进一步提高了算法的搜索效率.

2.2 降约束求解

HCA* 算法由于在单机规划的同时已考虑了冲突避免, 因此具有较好的求解效率. 但其求解过程对机器人的优先级顺序较为敏感, 尤其在密集场景下, 频繁地占位与拥塞所产生的封堵问题将直接导致算法无法求解. 对此, 提出降约束求解策略. 即在 HCA* 顺序求解过程中, 若当前机器人遇到约束表中高优先级机器人路径的阻塞而无法抵达终点时, 则去除约束表中的部分内容, 降低约束条件再对其进行求解, 以获得该机器人的一条不完全满足约束条件的路径方案.

由此, 定义 3 种约束类型: 零约束、欠约束和全约束. 其中, 零约束表示机器人 a_i 在完全不考虑其他路径存在的情况下进行规划, 由该方式求解所得的路径定义为零约束路径, 用 $p^n_{i,b}$ (b 为当前求解轮次) 表示; 欠约束表示机器人 a_i 在部分考虑其他路径存在的情况下进行规划, 所得路径为欠约束路径, 用 $p^u_{i,b}$ 表示; 全约束表示机器人 a_i 在完全考虑其他路径存在的情况下进行规划, 所得路径为全约束路径, 用 $p_{i,b}$ 表示. 并规定由零约束和欠约束方式进行的求解统称降约束求解.

与 3 种约束类型相对应的为 3 类约束表: 全约束表、欠约束表和零约束表. 其中, 零约束表为空集; 欠约束表中包含此前求解的所有降约束路径 $\{\forall p_j \in P^u_b \cup P^n_b | j \prec i\}$ (P^u_b 为欠约束路径集, P^n_b 为零约束路径集); 全约束表则包含此前求解的所有路径 $\{\forall p_j \in P_b \cup P^u_b \cup P^n_b | j \prec i\}$ (P_b 为全约束路径

集). 按照约束强度, 求解过程中的每条单机路径首先将在现有的全部路径约束条件下进行求解. 若无法求解, 则去掉部分约束, 使用前面已获得的所有降约束路径为条件进行求解; 若仍无法求解, 则在不添加任何路径约束的条件下进行求解, 零约束可以保证有解并获得一条降约束路径, 使得后续机器人的求解过程得以继续. 因此, 在整个求解过程中形成以下的使用顺序: 全约束表–欠约束表–零约束表.

如图 4 中, 多个机器人将分占关隘口封堵后续多个机器人通过. 针对该问题, 首先求解 a_1, 由于 a_1 为最高优先级, 因此获得全约束路径 $p_{1,0}$. 将 $p_{1,0}$ 加入全约束表后求解 a_2, 由于 a_1 的终点阻塞了关隘内部通道, a_2 在全约束条件下无法求解, 转而使用降约束求解策略, 在当前欠约束表为 $\emptyset$ 的条件下求得 a_2 的欠约束路径 $p^u_{2,0}$, 并同时更新全约束表和欠约束表. 求解 a_3 时, 在全约束表下受到 a_1 阻挡后尝试使用欠约束表求解, 但由于 $p^u_{2,0}$ 的阻挡无法求解, 转而使用零约束表即空集作为约束进行求解, 获得零约束路径 $p^n_{3,0}$, 并同时更新全约束表和欠约束表信息. 重复上述过程, 将继续获得 $p^n_{4,0}$ 路径.

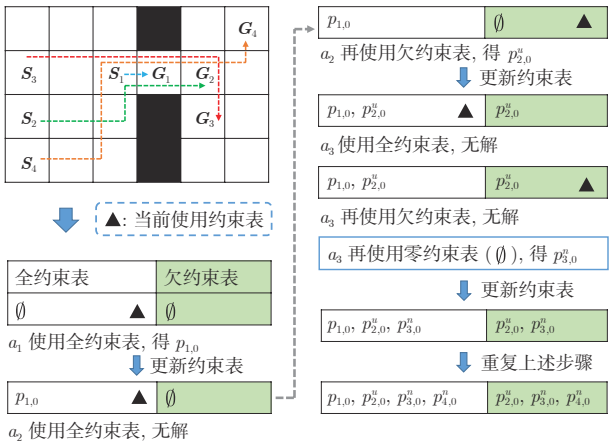


图 4　约束表使用与更新

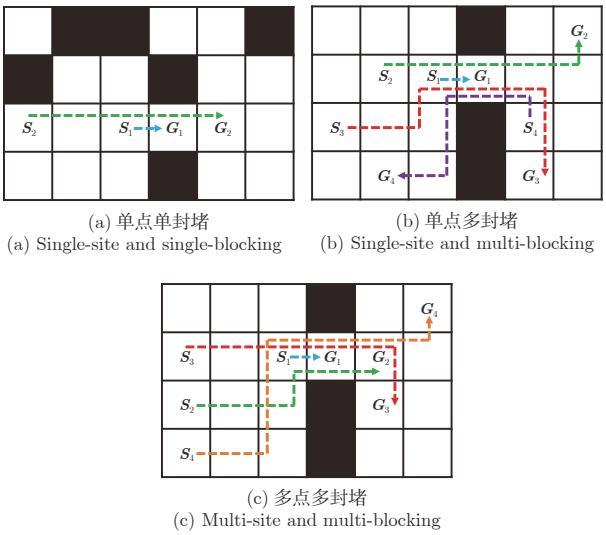
Fig. 4　Use and update of constraint tables

通过降约束求解方式, 可使求解过程中遇到阻塞的机器人去除部分阻塞性约束以获得 $p^u_{i,b}$ 或 $p^n_{i,b}$ 两类降约束路径. 由此, 顺序求解过程得以继续, 并最终获得所有机器人包含全约束路径或降约束路径在内的初步方案.

2.3　讨价还价博弈机制

在降约束求解的基础上, 算法进一步的目标是通过讨价还价博弈, 调整冲突各方的妥协解, 以期将方案中的降约束路径 $p^u_{i,b}$ 或 $p^n_{i,b}$ 在进行多次协商后均转为全约束路径 $p_{i,B}$ (B 为最终求解总轮数), 由此获得全局无冲突的可行路径方案.

以图 5 中 3 类常见的封堵问题为例, 阐述讨价还价博弈机制. 其中, 单点单封堵问题为一个机器人在关隘封堵了后续一个机器人的通过; 单点多封堵为一个机器人封堵了后续多个机器人的通过; 多点多封堵为多个机器人分占关隘口封堵后续多个机器人的通过.



(a) 单点单封堵
(a) Single-site and single-blocking

(b) 单点多封堵
(b) Single-site and multi-blocking

(c) 多点多封堵
(c) Multi-site and multi-blocking

图 5　密集场景中典型封堵问题

Fig. 5　Typical blocking problems in dense scenarios

1) 基本思想

如图 6 所示, 对于单点单封堵问题, 首先使用 HCA* 算法对各机器人进行顺序求解, 获得 a_1 的路径 $p_{1,0}$ 并加入约束表, 对 a_2 出价 $p_{1,0}$. 由于路径 $p_{1,0}$

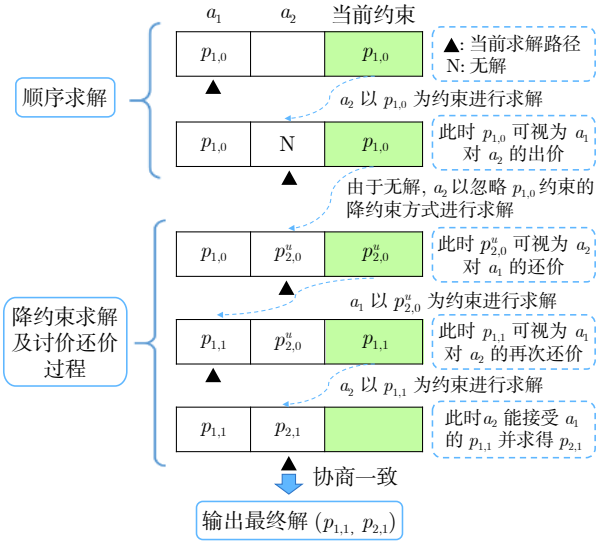


图 6　讨价还价博弈基本思路

Fig. 6　Basic example of bargaining game process

在终点处对关隘通道进行封堵, 导致 a_2 在该约束下无法求解, 则 a_2 不同意 a_1 的 $p_{1,0}$ 路径方案, 转而采取降约束求解的方式.

采用不包含 $p_{1,0}$ 的欠约束表进行 a_2 的降约束求解, 获得欠约束路径 $p^u_{2,0}$, 则 a_2 以 $p^u_{2,0}$ 对 a_1 进行还价. 保留上述过程获得的全约束表, 清空欠约束表, 进入讨价还价环节.

从 a_1 开始依次求解, 与顺序求解环节不同的是: 讨价还价过程中每个待求机器人 a_i 不再仅考虑高优先级的路径约束, 而需考虑除自身以外的其他所有机器人的全局约束, 即在全约束表中包含除当前机器人外的所有路径 $\{\forall p_j \in P_b \cup P^u \cup P^n | j \neq i\}$. 讨价还价环节可进行多个轮次 (文中设置上限 $B_L = 10$), 下一轮开始时将继承上一轮获得的全约束表, 并将欠约束表清零. 如图 6 中, 在继承顺序求解环节所得全约束表的基础上, a_1 放弃原先的 $p_{1,0}$ 路径方案, 转而在考虑 a_2 的 $p^u_{2,0}$ 基础上重新规划路径得 $p_{1,1}$, 并再次向 a_2 出价. 此时的新路径方案 $p_{1,1}$ 由于考虑了 $p^u_{2,0}$ 的行进路径, 因此在封堵关隘前为 a_2 预留了一定的行动间隙. 在 $p_{1,1}$ 基础上, a_2 得以重新规划并获得与 a_1 无冲突的全约束路径 $p_{2,1}$. 由此, 满足所有路径均为全约束路径 $p_{i,B}$ 的条件, 获得最终结果 $(p_{1,1}, p_{2,1})$, 如表 1 所示.

表 1 单点单封堵类型路径规划结果

Table 1 Solution of the single-site and single-blocking

机器人	路径	具体方案
a_1	$p_{1,1}$	[(3, 2), (3, 2), (3, 1), (3, 2), (4, 2)]
a_2	$p_{2,1}$	[(1, 2), (2, 2), (3, 2), (4, 2), (5, 2)]

2) 单点多封堵问题

单点多封堵问题在顺序求解环节中将产生多条降约束路径, 如图 7 所示, 通过使用欠约束表, 将前面已求的欠约束路径 $\{\forall p_j \in P^u_0 \cup P^n_0 | j \prec i\}$ 添加作为后续待求欠约束路径 $p^u_{i,0}$ 的约束. 由于该问题中欠约束路径间不存在相互封堵的情况, 因此可获得相互无冲突的欠约束路径集 $(p^u_{2,0}, p^u_{3,0}, p^u_{4,0})$.

在讨价还价环节, a_1 以 $(p^u_{2,0}, p^u_{3,0}, p^u_{4,0})$ 为约束

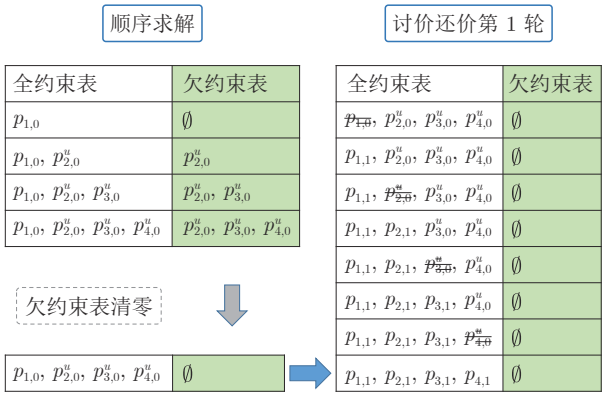


图 7 单点多封堵求解过程

Fig. 7 Solving process of the single-site and multi-blocking

重新规划, 新的出价方案 $p_{1,1}$ 将考虑到对后续机器人整体的封堵问题, 并留出相应的时间间隙予以解决. 由此, 在 $p_{1,1}$ 基础上继续规划将不再导致封堵现象, 进而可获得最终的全约束路径 $(p_{1,1}, p_{2,1}, p_{3,1}, p_{4,1})$, 如表 2 所示.

通过欠约束表添加前后的约束关系, 单点多封堵问题中的被封堵对象形成彼此间无冲突的降约束路径方案, 并以此作为一个整体与单一的封堵方形成对立, 使得问题得以简化为单点单封堵类型, 进而得以求解.

3) 多点多封堵问题

多点多封堵问题存在欠约束路径 $p^u_{i,b}$ 封堵零约束路径 $p^n_{k,b}$ $(i \prec k)$ 的冲突现象. 因此, 针对该问题, 需对欠约束表中涉及的所有路径进行二次规划以使欠约束路径 $p^u_{i,b}$ 和零约束路径 $p^n_{k,b}$ 冲突消解.

二次规划作为一个独立模块, 位于主体求解环节即顺序求解或每一轮的讨价还价之后. 其求解过程参照讨价还价环节, 并设置二次规划全约束表和零约束表. 其中, 二次规划全约束表继承本轮主体求解环节所得的欠约束表信息. 二次规划将执行单轮的求解任务, 求解过程中每个降约束个体 a_i 需考虑除自身以外的其他所有降约束路径的约束, 即在二次规划全约束表中包含除当前机器人以外的所有降约束路径 $\{\forall p_j \in P^u \cup P^n | j \neq i\}$, 依次对降约束

表 2 单点多封堵类型路径规划结果

Table 2 Solution of the single-site and multi-blocking

机器人	路径	具体方案
a_1	$p_{1,1}$	[(3, 3), (3, 2), (3, 3), (3, 2), (3, 3), (3, 3), (3, 3), (3, 3), (3, 4), (3, 3), (4, 3)]
a_2	$p_{2,1}$	[(2, 3), (3, 3), (4, 3), (5, 3), (5, 4), (6, 4)]
a_3	$p_{3,1}$	[(1, 2), (1, 3), (2, 3), (3, 3), (4, 3), (5, 3), (5, 2), (5, 1)]
a_4	$p_{4,1}$	[(5, 2), (5, 3), (5, 3), (5, 2), (5, 3), (5, 4), (5, 3), (4, 3), (3, 3), (3, 2), (3, 1), (2, 1)]

个体进行求解, 并在求解完成后用所得结果对本轮主体求解结果中对应个体的路径进行更新.

如图 8 所示, 针对多点多封堵问题在顺序求解环节产生的所有降约束路径即图 4 欠约束表中 $(p^u_{2,0}, p^n_{3,0}, p^n_{4,0})$ 路径集执行二次规划任务. 首先, a_2 删除原有的路径 $p^u_{2,0}$, 在 $(p^n_{3,0}, p^n_{4,0})$ 的约束下重新求解并获得路径 $p^{u'}_{2,0}$ 作为新的出价.

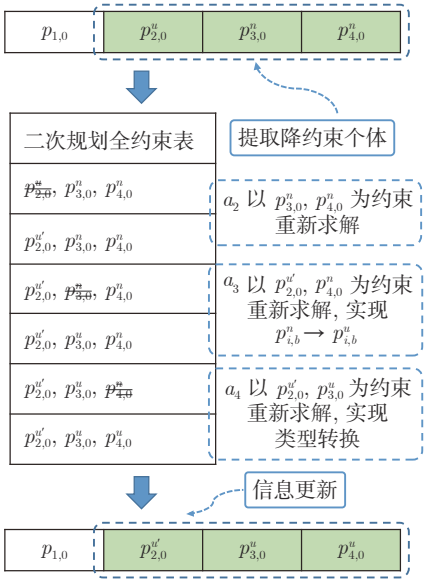


图 8　多点多封堵问题降约束个体二次规划

Fig. 8　Replanning of reduced constraint individuals for the multi-site and multi-blocking

由于 $p^{u'}_{2,0}$ 考虑了路径 $(p^n_{3,0}, p^n_{4,0})$ 及可能存在的封堵问题, 与单点多封堵问题类似, 继续执行 $(p^n_{3,0}, p^n_{4,0})$ 的重新规划时将不再出现封堵问题, 从而获得无冲突的降约束路径集 $(p^{u'}_{2,0}, p^u_{3,0}, p^u_{4,0})$. 再对主体结果对应的 $(p^u_{2,0}, p^n_{3,0}, p^n_{4,0})$ 进行更新, 完成本轮主体求解所得降约束路径中零约束路径 $p^n_{j,b}$ 向欠约束路径 $p^u_{i,b}$ 的转换. 在此基础上, 通过 1 轮讨价还价, 获得路径方案如表 3 所示.

通过二次规划, 可将降约束路径中的冲突问题转换为单点多封堵问题, 从而获得无冲突的降约束路径整体方案. 在此基础上逐层递进, 使得多点多封堵问题最终简化为单点单封堵问题, 并获得求解.

2.4　HCA* 搜索熔断机制

在原始 HCA* 算法的单机路径搜索过程中加入熔断机制, 以改善因长时绕道而造成的路径总体代价增加以及关口死循环而造成的求解效率下降等问题.

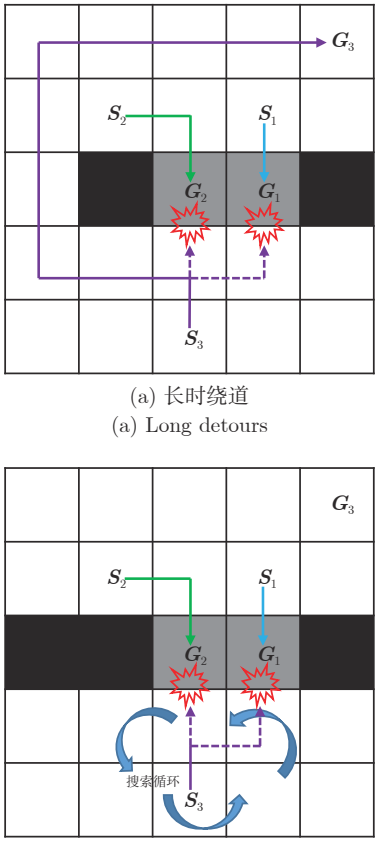
如图 9(a) 所示, 由于高优先级的 a_1, a_2 到达终点并阻碍 a_3 的通过, 因此 a_3 需绕道左边关口通行, 在很大程度上增加了路径代价. 更进一步, 若形成图 9(b) 的全封闭地形, 则 a_3 由于无法通过该关口, 将返回之前搜索过的位置并以时间步为区分进行反复搜索和扩展, 形成死循环状态. 因此, 需要在 HCA* 算法的单机规划过程中设置相应的熔断机制. 当发现长时绕道, 尤其是死循环时将其强制退出, 并进入到下一个单机规划任务中.

通过对比单机规划层中 A* 算法的已扩展节点数 (N_s) 和启发函数计算层中 RRA* 算法的已扩展

表 3　多点多封堵类型路径规划结果

Table 3　Solution of the multi-site and multi-blocking

机器人	路径	具体方案
a_1	$p_{1,1}$	[(3, 3), (3, 3), (3, 4), (3, 4), (3, 4), (3, 4), (3, 3), (4, 3)]
a_2	$p_{2,1}$	[(1, 2), (1, 3), (2, 3), (3, 3), (3, 2), (3, 3), (4, 3), (5, 3)]
a_3	$p_{3,1}$	[(1, 3), (2, 3), (3, 3), (4, 3), (5, 3), (5, 2)]
a_4	$p_{4,1}$	[(1, 1), (1, 2), (1, 3), (2, 3), (3, 3), (4, 3), (5, 3), (5, 4), (6, 4)]



(a) 长时绕道

(a) Long detours

(b) 关口死循环

(b) Search loop at the pass

图 9　HCA* 底层算法低效搜索状态

Fig. 9　Inefficient searching in underlying of HCA*

节点数 (N_r), 为 HCA* 算法设置动态熔断机制. 其中, 单机规划层使用时空 A* 算法, 对于相同坐标位置的节点可重复扩展并以时间步作为区分; 启发函数计算层的 RRA* 只有空间维度, 相同位置的节点不重复扩展. 由于单机规划层 A* 算法的新增节点需先调用 RRA* 以获得该坐标位置的启发函数值, 因此, 单机规划 A* 中的已扩展节点 (close-sp) 是由 RRA* 中的已扩展节点 (close-rra) 的子集以时间步扩展的方式组合而成. 在正常没有阻塞的情况下, N_s 要小于 N_r; 而在路径发生部分堵塞的情况下, N_s 将随着时间维的扩展而激增. 由此, 通过对比 N_s 和 N_r 的数值关系而为 HCA* 算法设置动态熔断机制, 即

$$F_d = \begin{cases} \text{True}, & N_s > \omega N_r \\ \text{False}, & N_s \le \omega N_r \end{cases} \tag{7}$$

式中, F_d 为 True 时, 表示到达动态熔断阈值, A* 算法将停止本次搜索任务并进入下一个任务; F_d 为 False 时, A* 算法将继续本次搜索任务直至求解完毕或到达动态熔断阈值. ω 为拥堵缓冲系数, 用于缓冲暂时性的拥堵以减小误判的概率, ω 可根据地图场景的复杂度和机器人数量进行设置和动态调整, 一般设为 $2 \sim 12$ 之间.

在此基础上, 通过设置静态熔断机制来限制单机规划层 A* 搜索的总次数, 以对动态熔断机制进行补充, 并与其形成"或" 逻辑关系. A* 算法在搜索过程中将逐轮地不断执行节点扩展, 并对扩展点进行是否与其他机器人路径存在冲突的有效性检测, 直至找到目标点或 Open 表为空. 因此, 以该过程中 A* 算法所执行的轮数为对象, 设置搜索轮数限制, 当轮数超过最大限制 R_L 时, 则跳出本次任务进入下一机器人的规划任务中, 即

$$F_s = \begin{cases} \text{True}, & R > R_L \\ \text{False}, & R \le R_L \end{cases} \tag{8}$$

式中, F_s 用于标识搜索过程是否到达静态熔断阈值, R 为本次单机任务中 A* 算法搜索主循环所执行的轮数, R_L 为最大轮数限制, 可根据场景复杂度和任务规模进行设置和调整, 一般设为 6 000 ~ 15 000 之间.

2.5 基于讨价还价博弈机制的 B-IHCA* 算法

综合上述分析, 通过讨价还价博弈机制构造求解框架并加入带熔断机制的改进 HCA* 算法作为内核, 形成基于讨价还价博弈机制的改进 HCA* (Bargaining game based improving HCA*, B-IHCA*) 多机器人路径规划算法, 如算法 1 所示.

算法 1. B-IHCA* 算法

输入. 机器人集合 A, 地图 M

输出. 机器人路径方案 P

1: 初始化: $P, F_r, F_c \leftarrow \emptyset$; $b = 0$
2: **while** $b < B_L$ **do**
3: 初始化: $U_r, U_c, A_t \leftarrow \emptyset$
4: **for** a_i in A **do**
5: $p_i \leftarrow$ 带熔断 A* 搜索 (a_i, M, F_r, F_c)
6: **if** $p_i = \emptyset$ **then**
7: $p_i \leftarrow$ 带熔断 A* 搜索 (a_i, M, U_r, U_c)
8: **if** $p_i = \emptyset$ **then**
9: $p_i \leftarrow$ 带熔断 A* 搜索 (a_i, M, $\emptyset$, $\emptyset$)
10: **end if**
11: 降约束信息表添加: $U_r, U_c \leftarrow p_i$; $A_t \leftarrow a_i$
12: **end if**
13: 全约束信息表添加: $P, F_r, F_c \leftarrow p_i$
14: **if** $b \ge 1$ and $i < N_{\text{robot}}$
15: P, F_r, F_c 移除 p_{i+1} 信息
16: **end if**
17: **end for**
18: **for** a_i in A_t **do**
19: U_r, U_c 移除 p_i 信息
20: $p_i \leftarrow$ 带熔断 A* 搜索 (a_i, M, U_r, U_c)
21: **if** $p_i = \emptyset$ **then**
22: $p_i \leftarrow$ 带熔断 A* 搜索 (a_i, M, $\emptyset$, $\emptyset$)
23: **end if**
24: 添加二次规划信息: $U_r, U_c \leftarrow p_i$
25: 更新全约束信息: $P, F_r, F_c \leftarrow p_i$
26: **end for**
27: P, F_r, F_c 移除 p_0 信息
28: $b = b + 1$
29: **return** P, 若当前 P 中各路径均为全约束路径
30: **end while**
31: **return** $P = \emptyset$, 若 b 到达上限仍无解.

算法 1 整体由外层的讨价还价博弈框架嵌套多个内层带熔断机制的 A* 算法 (算法 2) 构成. 算法 1 中, 各机器人首先进行全约束求解 (步骤 4 和步骤 5), 若无法求解, 再进行降约束求解 (步骤 6 ~ 10), 并对欠约束表和全约束表进行更新 (步骤 11 ~ 13). 在完成每 1 轮的主体求解环节后, 执行二次规划 (步骤 18 ~ 26), 对获得的降约束路径进行再调整. 其中, p_i 为机器人 a_i 的路径, P 为全体路径集, A_t 为求解过程中的降约束个体集; F_r 用于保存全约束表中各机器人的路径信息, F_c 用于保存全约束表中各机器人的终点信息, 两者共同构成全约束表; U_r 和 U_c 共同构成欠约束表.

算法 2 (子模块). 带熔断 A* 搜索

输入. 机器人 a_i, 地图 M, 路径约束 IN_r, IN_c

输出. 单机路径 p_i

1: 初始化: $close\text{-}sp \leftarrow \emptyset$, $R = 1$, $N_s = 0$
2: S, $G \leftarrow$ 获得任务信息 (a_i)
3: 生成 *close-rra* 表: RRA* (S, G, M)
4: $N_r \leftarrow$ 统计节点数 (*close-rra*)
5: 添加起点: $open \leftarrow S$
6: **while** $open \neq \emptyset$ and $N_s \leq \omega N_r$ and $R \leq R_L$ **do**
7: 　获取 *open* 表中代价最小节点 C
8: 　添加: $close\text{-}sp \leftarrow C$
9: 　**if** $C = G$ **then**
10: 　　溯源父节点 (C), **return** p_i
11: 　**end if**
12: 　$Q \leftarrow$扩展子节点 (C, M)
13: 　更新 *close-rra* 表: RRA* (Q, G, M)
14: 　添加或更新: $open \leftarrow Q$, 若 Q 与 IN_r, IN_c 无冲突且不在 *close-sp* 中
15: 　N_s, $N_r \leftarrow$ 统计节点数 (*close-sp*, *close-rra*)
16: 　$R = R + 1$
17: **end while**
18: **return** $p_i = \emptyset$, 若达到熔断条件或 *open* 为空.

3　仿真实验结果

3.1　实验设置

为分析算法对不同密集度地图的求解性能, 采用参考文献中的典型地图算例进行仿真实验, 分别为 8×8 地图[11], 20×20 地图[16] 和 32×32 地图[33]. 这些典型地图展示了不同的密集程度和地形特征.

为更好地体现地图及其测试算例的特征, 设定环境密集指数 α 和机器密集指数 β, 定义如式 (9) 和式 (10) 所示. 式中, N_{total} 为地图栅格总数, N_{obstacle} 为障碍栅格数量, N_{robot} 为机器人总数.

$$\alpha = \frac{N_{\text{obstacle}}}{N_{\text{total}}} \tag{9}$$

$$\beta = \frac{N_{\text{robot}}}{N_{\text{total}} - N_{\text{obstacle}}} \tag{10}$$

由此, 设置两类实验, 实验 I 为固定任务, 被测试算例 N_{robot} 固定且起点、终点由人为设定; 实验 II 为随机任务, 分别在各实验地图上测试 6 组不同 N_{robot} 规模的任务, 且起点、终点均随机生成.

分别采用 CBS[14]、CBS-DS[16]、HCA* [19]、Wd-SIPP[25] 算法与本文的 B-IHCA* 算法对测试算例进行求解. 其中, 为平衡求解成功率与路径质量, Wd-SIPP 算法次优权重系数设置为 $w = 1.75$, 并设置 $w = 1.01$ 作为路径质量对照[25]. 所有算法均在 AMD Ryzen 4800 处理器, 内存为 16 GB 的 PC 机上以 Python 程序运行. 针对任务 N_{robot} 的不同规模, 设置相应求解限制时间如表 4 所示, 所有算法需在限制时间内完成对问题的求解, 超出限制时间则视为求解失败.

表 4　不同 N_{robot} 规模对应的求解限制时间
Table 4　Time limits for different sizes of N_{robot}

机器人个数	限制时间 (s)
$0 < N_{\text{robot}} \leq 50$	150
$50 < N_{\text{robot}} \leq 100$	300
$N_{\text{robot}} > 100$	500

3.2　实验 I. 小规模固定任务测试

采用 8×8 地图[11], 并设定 10 个机器人的起始点和终点如图 10 所示, 其他相关实验参数设置及其指标如表 5 所示. 该算例中, 可以发现 (2, 5) 为一个关隘且被 a_2 占据, 阻挡了低优先级的 a_3, a_4 通过. 同时, a_8, a_{10} 的起点各占据一个通道, 在它们到达终点后将使图中 (4, 2), (7, 1) 等位置点成为地图的新增关隘, 对任务环境造成进一步的拥塞和限制.

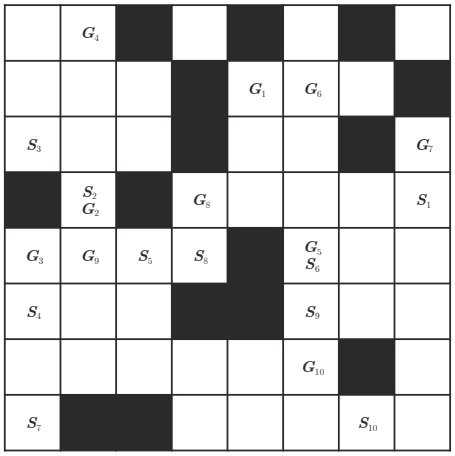


图 10　8×8 地图[11] 及任务设置
Fig. 10　8×8 map[11] and its tasks

表 5　实验 I 参数设置
Table 5　Parameters for Experiment I

地图类型	N_{robot}	α (%)	β (%)	ω	R_L
8×8 grid	10	23.4375	20.4082	3	6000

各算法对图 10 算例的测试结果如表 6 所示. 表 6 中, 若算法在限制时间内无法完成对该算例的

表 6 实验 I 求解结果
Table 6 Results of Experiment I

算法	路径总代价 (步)	求解时间 (s)
HCA*	NA	NA
CBS	NA	NA
CBS-DS	62	132.0
WdSIPP ($w = 1.01$)	NA	NA
WdSIPP ($w = 1.75$)	NA	NA
B-IHCA*	73	0.1

求解, 则结果以 NA 表示; 若在限制时间内完成求解任务, 则记录实际的路径总代价指标.

由表 6 可知, 由于该算例的任务构造了多个动态关隘, 形成较为复杂的地形结构, 只有 CBS-DS 与 B-IHCA* 算法在限定时间内完成了求解任务. 从结果来看, B-IHCA* 相较于 CBS-DS 算法在路径总代价上有所增加, 但在求解时间上具有明显优势. 此外, B-IHCA* 算法对该任务的求解过程共经过两轮, 各阶段的路径约束情况及冲突信息如表 7 所示.

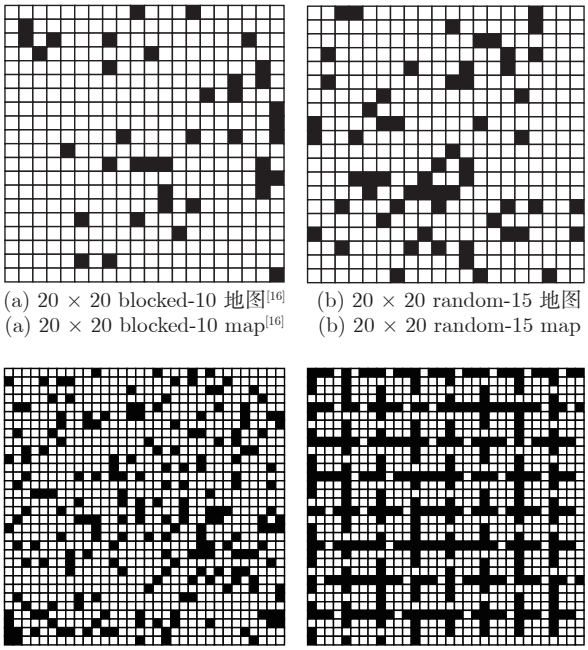
表 7 B-IHCA* 求解过程路径与冲突信息
Table 7 Path and conflict information during the solving process of B-IHCA*

求解阶段	信息类型	信息值
顺序求解	全约束路径	$p_{1,0}$, $p_{2,0}$, $p_{5,0}$, $p_{6,0}$, $p_{7,0}$, $p_{9,0}$, $p_{10,0}$
	欠约束路径	$p^u_{3,0}$, $p^u_{4,0}$, $p^u_{8,0}$
	零约束路径	$\emptyset$
	冲突机器人	(a_2, a_3), (a_2, a_4), (a_5, a_8)
讨价还价 第 1 轮	全约束路径	$p_{1,1}$, $p_{2,1}$, $p_{3,1}$, $p_{4,1}$, $p_{5,1}$, $p_{6,1}$, $p_{7,1}$, $p_{8,1}$, $p_{9,1}$, $p_{10,1}$
	欠约束路径	$\emptyset$
	零约束路径	$\emptyset$
	冲突机器人	$\emptyset$

上述结果表明, B-IHCA* 算法在其他算法难以有效求解的高耦合算例中, 通过讨价还价间接调整了冲突机器人的优先级顺序, 有效提高了求解成功率. 此外, 结合表 7 的两轮求解过程可知, 在原始 HCA* 对 a_3, a_4 的单机规划陷入死循环时, 本文设置的熔断机制及时地进行了熔断操作, 并进入下一个规划任务, 有效避免了计算资源和时间的浪费.

3.3 实验 II. 大规模随机任务测试

采用 20×20、32×32 等不同规格且环境密集度依次增加的 4 类地图如图 11 所示, 每类地图各设置 6 组机器人, 每组机器人均测试 20 次随机任务. 其中, 图 11(b) 由本文方法根据随机机制生成.



(a) 20×20 blocked-10 地图[16]
(a) 20×20 blocked-10 map[16]
(b) 20×20 random-15 地图
(b) 20×20 random-15 map
(c) 32×32 blocked-20 地图[33]
(c) 32×32 blocked-20 map[33]
(d) 32×32 room 地图[33]
(d) 32×32 room map[33]

图 11 大规模随机任务实验地图
Fig. 11 Maps for large scale randomized tasks

此外, 若图内存在不可达点, 则视为障碍物. 各地图的环境密集指数 α 与 B-IHCA* 算法 R_L 参数设置如表 8 所示, 其余各数量组设置及其对应的相关参数如表 9 所示.

表 8 实验 II 参数设置 (1)
Table 8 The parameters for Experiment II (1)

地图类型	α (%)	R_L
20×20 blocked-10 地图	9.7500	10 000
20×20 random-15 地图	15.0000	15 000
32×32 blocked-20 地图	19.9219	15 000
32×32 room 地图	33.3984	15 000

1) 求解能力分析

各算法的求解成功率以及求解时间的离散分布统计结果如图 12 所示. 其中, 若算例在限制时间内无法求解, 则以限制时间作为统计时间. 同时, 为验证所提二次规划模块的作用, 图 12 在求解成功率的测试中加入 B-IHCA* 算法不带二次规划模块的求解结果: B-IHCA* (NR).

从图 12 中可以看出, B-IHCA* 算法相较于对比算法在求解成功率和稳定性上均有较大优势. 其中, CBS 与 CBS-DS 算法由于在密集场景中, 上层最优搜索的扩展节点数量激增, 求解能力限制在 50 个以内. HCA* 算法求解成功率则相对更高, 这

表 9　实验 II 参数设置 (2)

Table 9　The parameters for Experiment II (2)

20 × 20 blocked-10 地图			20 × 20 random-15 地图			32 × 32 blocked-20 地图			32 × 32 room 地图		
N_{robot}	β (%)	ω	N_{robot}	β (%)	ω	N_{robot}	β (%)	ω	N_{robot}	β (%)	ω
20	5.5402	2	15	4.4118	2	30	3.6585	4	10	1.4663	4
40	11.0803	4	30	8.8235	5	50	6.0976	5	20	2.9326	5
60	16.6205	6	45	13.2353	6	70	8.5366	6	30	4.3988	6
80	22.1607	6	60	17.6471	6	90	10.9756	6	40	5.8651	8
100	27.7008	7	75	22.0588	8	110	13.4146	8	50	7.3314	10
120	33.2410	8	90	26.4706	10	130	15.8537	10	60	8.7977	12

是由于 HCA* 算法以损失路径质量作为代价, 求解能力得到增强. 但随着机器人数量增加到 80 个以上时, HCA* 算法的求解成功率出现明显下降. WdSIPP 算法在 HCA* 的基础上有进一步改善, 表明其对较大规模问题具有更为高效和灵活的解空间探索能力, 但当问题规模增加到上百个机器人时, 由于机器人间拥塞和封堵概率的增加, 其求解成功率也降到 60% 以下. 本文的 B-IHCA* 算法通过动态消解机器人间的封堵问题, 在上百个机器人的任务规模中, 始终能够保证 80% 以上的求解成功率.

同时, 从 B-IHCA* (NR) 与 B-IHCA* 的对比中可以发现, 不带二次规划模块的 B-IHCA* (NR) 在图 12(c) 和图 12(d) 的高数量段中, 求解成功率较 B-IHCA* 出现下降. 表明在 α 值更大的 blocked-20 与 room 等地图环境中, 更易出现多点多封堵问题, 而二次规划起到了进一步提升算法求解能力的作用.

此外, CBS 和 CBS-DS 算法对机器密集度参数 β 更为敏感, 即在 β 较大的 blocked-10 与 random-15 地图的测试组上, CBS 和 CBS-DS 算法的求解成功率下降更快, 这是由于其所需处理的冲突节点数量随 β 激增所致. 而 HCA*、WdSIPP 和 B-IHCA* 算法在 β 参数升高时下降趋势较平缓, 因为其所需避免的冲突约束量与 β 只呈比例关系, 说明该类型算法对高密集度场景中可调度空间的探索度更好.

在求解时间上, 由图 12 中可以发现, B-IHCA* 算法在中位数值和数据波动方面较 CBS 和 CBS-DS 算法均有更好的表现, 表明 B-IHCA* 继承了 HCA* 算法高效的特点. 在低数量段上, B-IHCA* 与 HCA* 算法的时间分布基本一致, 并在中位数上接近求解效率最高的 WdSIPP (1.75) 算法. 在高数量段上, B-IHCA* 算法的数据波动小于 WdSIPP (1.01) 和 WdSIPP (1.75) 算法, 同时可以看出, B-IHCA* 的下四分位和下边缘接近 HCA* 算法, 甚至在某些数量段要低于 HCA* 算法. 这是因为熔断机制对长时绕道的问题进行了及时处理, 提高了算法的求解效率, 表明 B-IHCA* 在大规模问题中具有更好的效率表现.

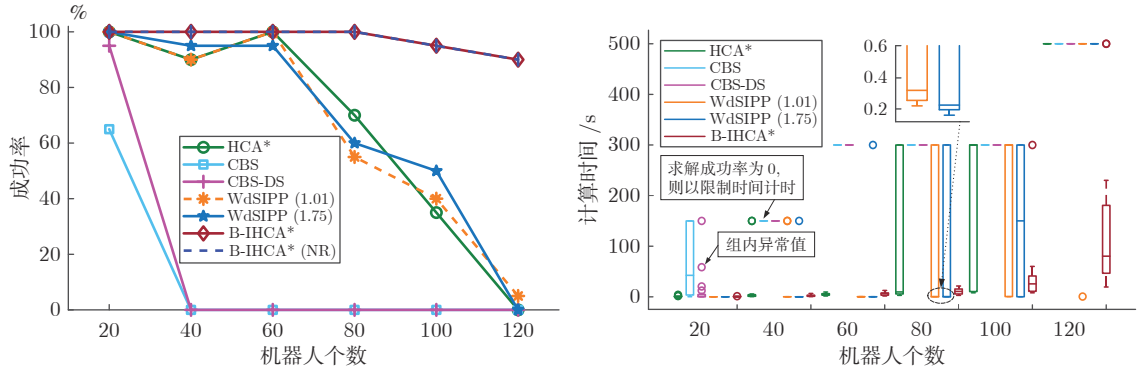
图 13 展示了 B-IHCA* 算法在 4 类地图最高数量段其中一个算例的求解结果. 图 13 中, 为避免同一栅格内的路径重叠显示, 对各路径做了偏移处理, 每个机器人路径在栅格内均有固定且彼此不重叠的显示位置. 从图 13 中可以发现, 部分窄通道或关隘位置的路径耦合度极高, 成为限制 HCA* 算法求解性能的重要因素. 通过讨价还价博弈机制, B-IHCA* 将拥塞机器人的优先级顺序进行调整, 逐步消解封堵, 获得可行解, 充分表明了该算法对密集场景中高耦合 MAPF 问题的求解优势.

2) 求解质量分析

在此基础上, 进一步对各算法所规划方案的路径总代价进行分析. 由于求解成功率的差异, 平均总代价需在公共算例的基础上做统计分析. 公共算例为同一数量段的 20 个测试算例中, 至少 2 种算法能共同求解的算例. 统计结果如表 10 所示, 表中, "NA" 表示该项目可统计数量为 0, "—" 表示该项目可统计数量小于 5 个. 此外, 对 B-IHCA* 求解轮数的统计分为两类: 一类为所对比的公共算例的平均求解轮数; 另一类为剩余未进行对比且可求解的算例的平均求解轮数.

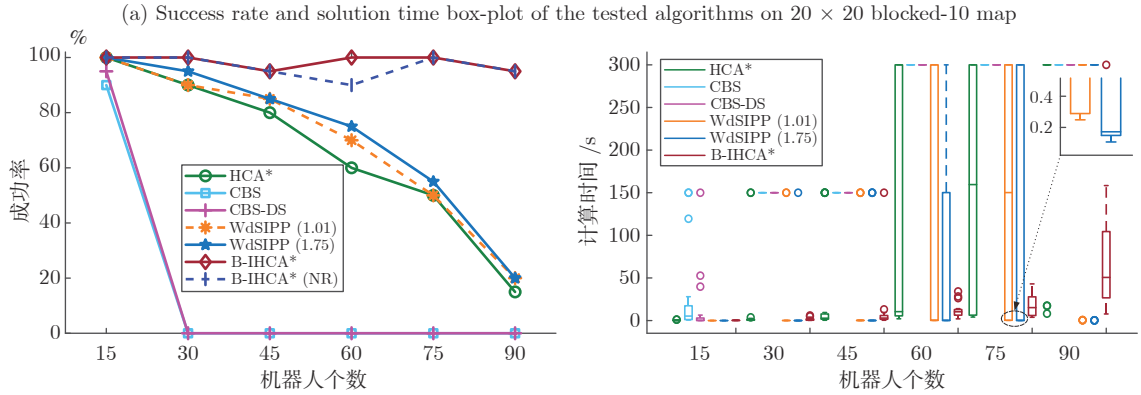
由表 10 可知, B-IHCA* 在各个数量段上的规划方案平均总代价相较于 HCA*、WdSIPP (1.75) 算法均有一定程度的改善, 并普遍优于 WdSIPP (1.01) 所得结果, 表明 B-IHCA* 通过熔断和调解, 有效地优化了路径的长时绕道问题. 在与 CBS、CBS-DS 两类具有最优性的算法对比中, B-IHCA* 的路径代价与其差距均在 2.2% 以内.

从 B-IHCA* 算法的求解轮数中可以发现, 公共算例平均经过 1 轮的讨价还价博弈即可完成求解. 而对于其他算法难以求解的非公共算例, B-IHCA* 也只需要经过平均 2 轮次的博弈与调解, 便可获得求解. 表明本文的博弈机制在高度拥塞的密集环境中, 具有较高的冲突化解效率. 同时, 路径优化效果与求解轮数呈正相关, 求解轮数越多, 则算法对过



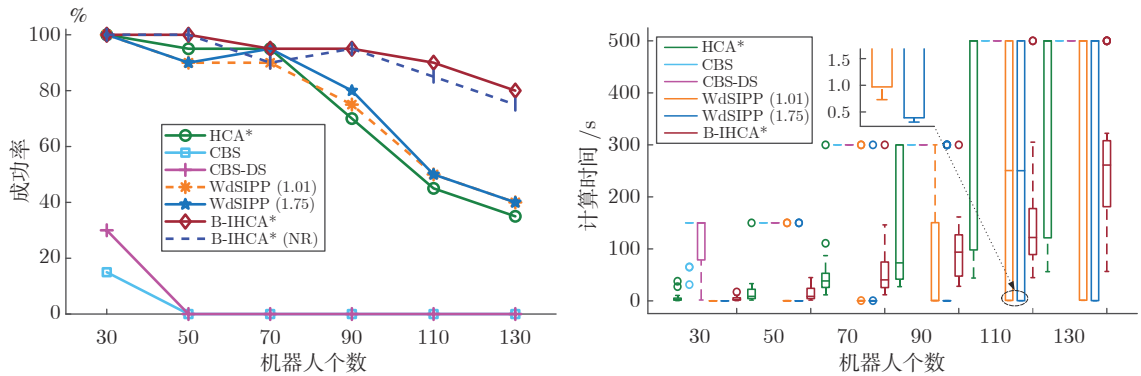
(a) 20 × 20 blocked-10 地图算法成功率与求解时间箱线图

(a) Success rate and solution time box-plot of the tested algorithms on 20 × 20 blocked-10 map



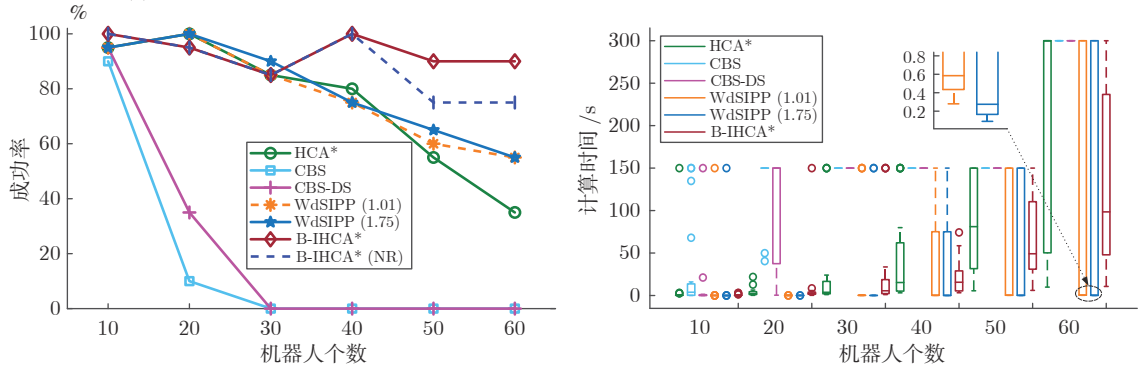
(b) 20 × 20 random-15 地图算法成功率与求解时间箱线图

(b) Success rate and solution time box-plot of the tested algorithms on 20 × 20 random-15 map



(c) 32 × 32 blocked-20 地图算法成功率与求解时间箱线图

(c) Success rate and solution time box-plot of the tested algorithms on 32 × 32 blocked-20 map

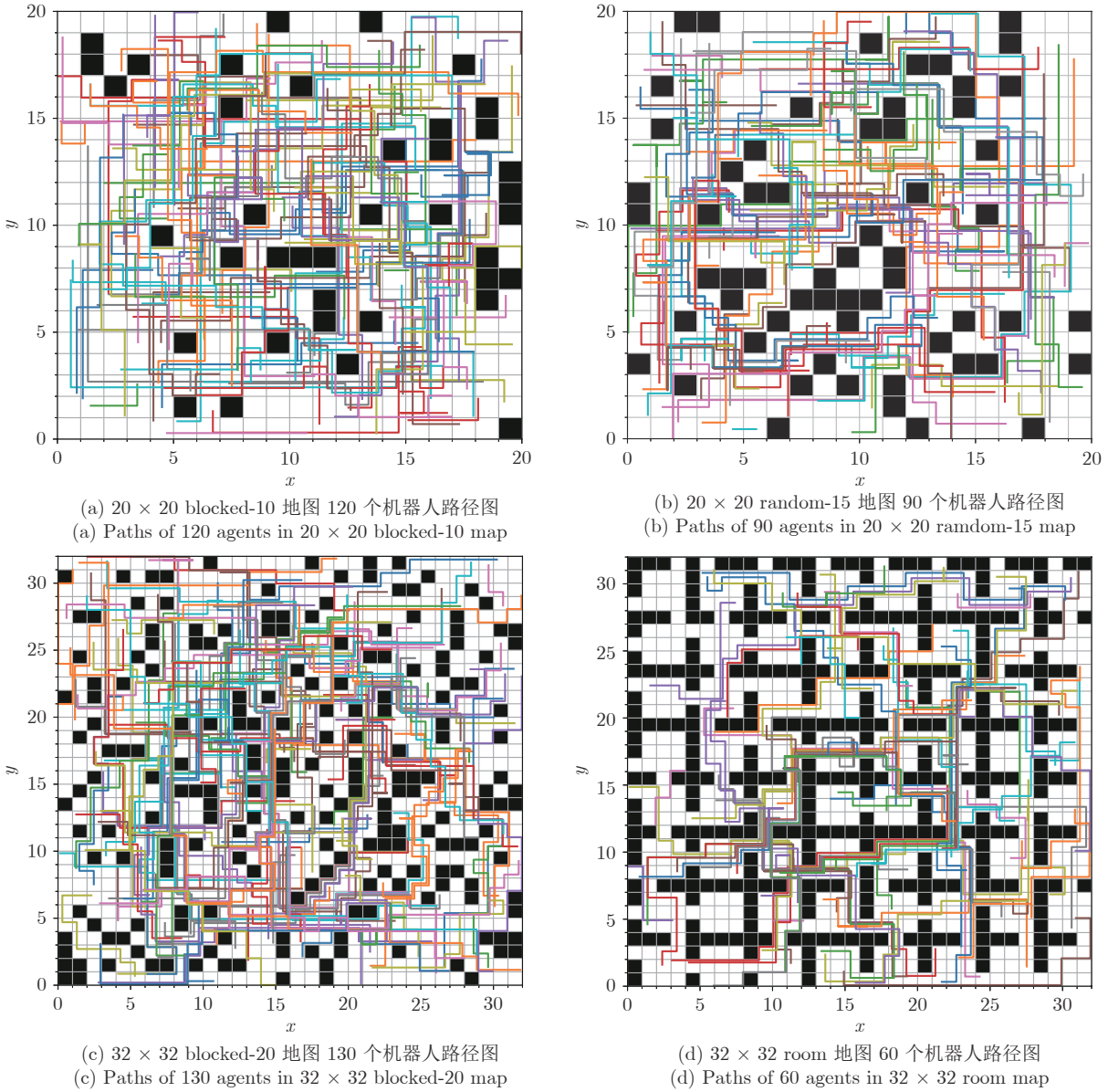


(d) 32 × 32 room 地图算法成功率与求解时间箱线图

(d) Success rate and solution time box-plot of the tested algorithms on 32 × 32 room map

图 12 各算法测试成功率及求解时间离散分布统计

Fig. 12 Statistics on success rate and solution time of the tested algorithms



(a) 20 × 20 blocked-10 地图 120 个机器人路径图
(a) Paths of 120 agents in 20 × 20 blocked-10 map

(b) 20 × 20 random-15 地图 90 个机器人路径图
(b) Paths of 90 agents in 20 × 20 ramdom-15 map

(c) 32 × 32 blocked-20 地图 130 个机器人路径图
(c) Paths of 130 agents in 32 × 32 blocked-20 map

(d) 32 × 32 room 地图 60 个机器人路径图
(d) Paths of 60 agents in 32 × 32 room map

图 13　4 类地图最高数量 B-IHCA* 算法求解路径图

Fig. 13　Path scheme of B-IHCA* algorithm for four tested maps in case of the highest number of robots

长路径的调整程度越大, 优化效果也越明显. 由于本文在测试中的主要目标是在限定的时间内以最快的方式获得可行解, 因此在求解轮数方面并未充分利用给定的限制时间. 若在限制时间内, 通过适当调整熔断阈值、增加求解轮数, 路径优化效果可进一步提高.

4　结束语

针对密集场景中的 MAPF 问题, 在 HCA* 算法基础上, 引入降约束求解策略、讨价还价博弈机制和搜索熔断机制构造 B-IHCA* 算法. 对顺序求解过程中遇到阻碍的机器人采取降约束求解策略获得降约束解, 以该解作为还价方案, 结合多轮讨价还价过程, 间接对冲突机器人的优先级顺序进行动态调整, 消解冲突, 提高了算法的求解成功率. 通过设置底层 A* 算法的搜索熔断机制, 有效避免了由于高优先级机器人的阻挡而使搜索陷于过长或反复无效搜索的状态, 减少了 HCA* 算法的平均无效求解时间. 仿真结果表明, 通过讨价还价博弈与熔断机制的相互配合, 使得 B-IHCA* 算法对多类地图的测试算例较现有方法在求解成功率上更具优势, 在机器人数量达到上百个的高度耦合环境中依旧具有较好的求解性能, 并在路径总代价指标上较 HCA*、WdSIPP 等算法有所改善, 验证了算法的有效性.

表 10 算法路径质量统计

Table 10 Path cost statistics of the tested algorithms

地图类型	N_{robot}	公共算例数	平均总代价 (步)						平均求解轮数	
			CBS	CBS-DS	HCA*	WdSIPP (1.01)	WdSIPP (1.75)	B-IHCA*	公共算例	非公共算例
20 × 20 blocked-10	20	13	**263.1**	267.1	273.0	275.8	281.5	267.2	1.54	1.43
	40	17	NA	NA	603.8	604.9	622.1	**591.9**	1.41	1.67
	60	19	NA	NA	933.2	928.2	969.5	**926.3**	1.32	2.00
	80	10	NA	NA	1 340.9	1 347.7	1 392.3	**1 327.8**	1.40	2.00
	100	6	NA	NA	**1 731.3**	1 744.2	1 820.7	**1 731.3**	1.00	2.62
20 × 20 random-15	15	18	**209.1**	211.1	219.1	220.1	224.4	213.7	1.50	1.00
	30	18	NA	NA	473.0	472.8	488.2	**470.2**	1.33	2.50
	45	15	NA	NA	745.2	745.1	771.3	**737.5**	1.27	2.25
	60	11	NA	NA	1 076.8	1 072.3	1 115.2	**1 055.9**	1.91	2.22
	75	9	NA	NA	1 391.2	1 390.4	1 437.4	**1 386.7**	1.33	2.09
32 × 32 blocked-20	30	6	—	**618.3**	626.2	630.0	655.7	623.7	1.17	1.29
	50	18	NA	NA	1 237.2	1 342.9	1 311.6	**1 234.0**	1.11	1.50
	70	17	NA	NA	1 781.5	1 778.6	1 881.8	**1 763.0**	1.41	2.50
	90	14	NA	NA	2 396.9	2 383.5	2 538.5	**2 368.1**	1.64	2.80
	110	9	NA	NA	3 010.3	3 008.6	3 190.3	**2 981.8**	1.89	2.33
	130	6	NA	NA	3 596.3	**3 555.5**	3 757.7	3 558.5	1.67	2.70
32 × 32 room	10	17	**248.7**	249.9	253.1	254.6	264.2	252.5	1.06	1.67
	20	6	—	**550.5**	566.5	563.7	585.8	558.2	1.50	1.62
	30	15	NA	NA	841.1	838.5	874.7	**830.8**	1.40	2.00
	40	14	NA	NA	1 188.3	**1 174.5**	1 208.9	1 175.6	1.43	1.83
	50	11	NA	NA	1 536.3	1 528.3	1 573.7	**1 512.4**	1.91	2.43
	60	6	NA	NA	1 891.5	1 892.8	1 949.7	**1 881.2**	1.67	2.83

References

1 Shi Wei, Feng Yang-He, Cheng Guang-Quan, Huang Hong-Lan, Huang Jin-Cai, Liu Zhong, et al. Research on multi-aircraft cooperative air combat method based on deep reinforcement learning. *Acta Automatica Sinica*, 2021, **47**(7): 1610−1623
(施伟, 冯旸赫, 程光权, 黄红蓝, 黄金才, 刘忠, 等. 基于深度强化学习的多机协同空战方法研究. 自动化学报, 2021, **47**(7): 1610−1623)

2 Li J Y, Surynek P, Felner A, Ma H, Kumar T K S, Koenig S. Multi-agent path finding for large agents. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI). Hono lulu, USA: AAAI, 2019. 7627−7634

3 Yu J J, LaValle S M. Optimal multi-robot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 2016, **32**(5): 1163−1177

4 Li Kun-Peng, Liu Teng-Bo, Ruan Yan-Qiu. Research on intelligent AGV routing scheduling with applications in semi-conductor production. *Computer Integrated Manufacturing Systems*, 2022, **28**(9): 2970−2980
(李昆鹏, 刘腾博, 阮炎秋. 半导体生产车间智能 AGV 路径规划与调度. 计算机集成制造系统, 2022, **28**(9): 2970−2980)

5 Morris R, Pasareanu C S, Luckow K, Malik W, Ma H, Kumar T K S, et al. Planning, scheduling and monitoring for airport surface operations. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI). Phoenix, USA: AAAI, 2016. 608−614

6 Yu J J, LaValle S M. Structure and intractability of optimal multi-robot path planning on graphs. In: Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI). Bellevue Washington, USA: AAAI, 2013. 1443−1449

7 Wagner G, Choset H. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 2015, **219**: 1−24

8 Han S D, Rodriguez E J, Yu J J. SEAR: A polynomial-time multi-robot path planning algorithm with expected constant-factor optimality guarantee. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Madrid, Spain: IEEE, 2018. 7967−7974

9 Duan Shu-Yong, Wang Qi-Fan, Han Xu, Liu Gui-Rong. Improved A-star algorithm for safety insured optimal path with smoothed corner turns. *Chinese Journal of Mechanical Engineering*, 2020, **56**(18): 205−215
(段书用, 王启帆, 韩旭, 刘桂荣. 具有确保安全距离的 A* 路径优化方法. 机械工程学报, 2020, **56**(18): 205−215)

10 Standley T, Korf R E. Complete algorithms for cooperative pathfinding problems. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI). Barcelona, Spain: AAAI, 2011. 668−673

11 Standley T. Finding optimal solutions to cooperative pathfinding problems. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI). Atlanta, GA, USA: AAAI, 2010. 173−178

12 Sharon G, Stern R, Goldenberg M, Felner A. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 2013, **195**: 470−495

13 Sharon G, Stern R, Felner A, Sturtevant N R. Meta-agent conflict-based search for optimal multi-agent path finding. In: Proceedings of the 5th Annual Symposium on Combinatorial Search (SoCS). Niagara Falls, Ontario, Canada: AAAI, 2012. 97−104

14 Sharon G, Stern R, Felner A, Sturtevant N R. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelli-*

gence, 2015, **219**: 40−66

15 Ma H, Harabor D, Stuckey P J, Li J Y, Koenig S. Searching with consistent prioritization for multi-agent path finding. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI). New Orleans, USA: AAAI, 2018. 7643−7650

16 Li J, Harabor D, Stuckey P J, Ma H, Koenig S. Disjoint splitting for multi-agent path finding with conflict-based search. In: Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS). Berkeley, USA: AAAI, 2019. 279−283

17 Boyarski E, Felner A, Stern R, Sharon G, Tolpin D, Betzalel O, et al. ICBS: Improved conflict-based search algorithm for multi-agent pathfinding. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI). Buenos Aires, Argentina: AAAI, 2015. 740−746

18 Andreychuk A, Yakovlev K, Boyarski E, Stern R. Improving continuous-time conflict based search. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI). Virtually, USA: AAAI, 2021. 11220−11227

19 Silver D. Cooperative pathfinding. In: Proceedings of the 1st AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE). California, USA: AAAI, 2005. 117−122

20 Erdmann M A, Lozano-Pérez T. On multiple moving objects. *Algorithmica*, 1987, **2**: 477−521

21 Sharon G. Novel Search Techniques for Path Finding in Complex Environment [Ph.D. dissertation], Ben-Gurion University, Israel, 2015

22 Čáp M, Novák P, Kleiner A, Selecký M. Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Transactions on Automation Science and Engineering*, 2015, **12**(3): 835−849

23 Li H, Long T, Xu G T, Wang Y J. Coupling-degree-based heuristic prioritized planning method for UAV swarm path generation. In: Proceedings of the Chinese Automation Congress (CAC). Hangzhou, China: IEEE, 2019. 3636−3641

24 Wu W Y, Bhattacharya S, Prorok A. Multi-robot path deconfliction through prioritization by path prospects. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Paris, France: IEEE, 2020. 9809−9815

25 Yakovlev K, Andreychuk A, Stern R. Revisiting bounded-suboptimal safe interval path planning. In: Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS). Nancy, France, USA: AAAI, 2020. 279−283

26 Li Xun, Nan Kai-Kai, Zhao Zheng-Fan, Wang Xiao-Hua, Jing Jun-Feng. Task allocation of handling robot in textile workshop based on multi-agent game theory. *Journal of Textile Research*, 2020, **41**(7): 78−87
(李珣, 南恺恺, 赵征凡, 王晓华, 景军锋. 多智能体博弈的纺织车间搬运机器人任务分配. 纺织学报, 2020, **41**(7): 78−87)

27 Wu H, Shang H L. Potential game for dynamic task allocation in multi-agent system. *ISA Transactions*, 2020, **102**: 208−220

28 Zheng Yan-Bin, Wang Lin-Lin, Xi Peng-Xue, Fan Wen-Xin, Han Meng-Yun. Multi-agent path planning algorithm based on ant colony algorithm and game theory. *Journal of Computer Applications*, 2019, **39**(3): 681−687
(郑延斌, 王林林, 席鹏雪, 樊文鑫, 韩梦云. 基于蚁群算法及博弈论的多 Agent 路径规划算法. 计算机应用, 2019, **39**(3): 681−687)

29 Kaduri O, Boyarski E, Stern R. Experimental evaluation of classical multi agent path finding algorithms. In: Proceedings of the 14th International Symposium on Combinatorial Search (SoCS). Guangzhou, China: AAAI, 2021. 126−130

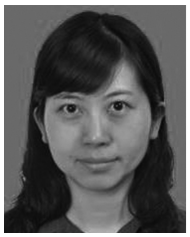
30 Li J Y, Andrew T, Scott K, Durham J W, Kumar T K S, Koenig S. Lifelong multi-agent path finding in large-scale warehouses. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI). Virtually, USA: AAAI, 2021. 11272−11281

31 Wan Qian. Research on Multi-AGV Path Planning for Logistics Sorting Based on CBS Algorithm [Master thesis], Harbin Institute of Technology, China, 2018
(万千. 基于 CBS 算法的物流分拣多 AGV 路径规划的研究 [硕士学位论文], 哈尔滨工业大学, 中国, 2018)

32 Ma H. Target Assignment and Path Planning for Navigation Tasks With Teams of Agents [Ph.D. dissertation], University of Southern California, USA, 2020

33 Sturtevant N R. Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012, **4**: 144−148

张凯翔 昆明理工大学机电工程学院博士研究生. 主要研究方向为移动机器人路径规划.
E-mail: kaixiangzhang35@163.com
(**ZHANG Kai-Xiang** Ph.D. candidate at the Faculty of Mechanical and Electrical Engineering, Kunming University of Science and Technology. His main research interest is mobile robot path planning.)



毛剑琳 昆明理工大学信息工程与自动化学院教授. 主要研究方向为通信网络资源分配与协议优化, 智能优化与调度算法, 多机器人系统协同控制研究. 本文通信作者.
E-mail: jlmao@kmust.edu.cn
(**MAO Jian-Lin** Professor at the Faculty of Information Engineering and Automation, Kunming University of Science and Technology. Her research interest covers communication network resource allocation and protocol optimization, intelligent optimization and scheduling algorithm, and cooperative control of multi-agent systems. Corresponding author of this paper.)

向凤红 昆明理工大学信息工程与自动化学院教授. 主要研究方向为智能控制理论与应用, 计算机网络控制系统. E-mail: xiangfh5447@sina.com
(**XIANG Feng-Hong** Professor at the Faculty of Information Engineering and Automation, Kunming University of Science and Technology. His research interest covers intelligent control theory and its applications, control system of computer network.)

宣志玮 昆明理工大学信息工程与自动化学院硕士研究生. 主要研究方向为移动机器人路径规划.
E-mail: rangxuan@foxmail.com
(**XUAN Zhi-Wei** Master student at the Faculty of Information Engineering and Automation, Kunming University of Science and Technology. His main research interest is mobile robot path planning.)