DOI: 10.19650/j.cnki.cjsi.J2312330

面向多无人车的目标点分配和协同路径规划算法

谷依田[1,2],张 涛[1,2],张 亮[1,2],杨泰泓[1,2]
(1. 东南大学仪器科学与工程学院 南京 210096; 2. 微惯性仪表与先进导航技术教育部重点实验室 南京 210096)

摘 要:针对多智能体路径搜索算法在非指定式多车协同路径规划问题中路径冗长,计算效率低等缺陷,提出协同目标点分配路径规划算法 Nutcracker-CBS。首先构建紧耦合目标点分配 MAPF 框架,实现目标点分配和路径构建的联合寻优;针对目标点分配模块,提出改进的星鸦优化算法,增量式求解分配问题,缩短模块用时;针对路径构建模块,提出改进的 MAPF 算法,通过回退式约束构建机制,引入避碰路径估计的绕道机制和数据共享底层路径规划机制,提升效率和路径质量。数据集实验中,Nutcracker-CBS 时耗相比 SOTA 算法减少 90.37%;目标点分配模块求解耗时减少 86.76%;MAPF 模块 6 s 内构建 100 辆无人车路径,平均路径长度缩短 6.058%。实际实验中路径总和与系统运行时长分别减少 55.26% 和 61.29%,提升了多机器人系统的效率,降低了路径长度。

关键词: 协同路径规划;多智能体路径搜索;基于冲突的搜索;目标点分配
中图分类号: TP242.6 TH39 **文献标识码**: A **国家标准学科分类代码**: 520·20

Goal allocation and cooperative path finding for multiple UGVs

Gu Yitian[1,2], Zhang Tao[1,2], Zhang Liang[1,2], Yang Taihong[1,2]
(*1. School of Instrument Science & Engineering, Southeast University, Nanjing 210096, China; 2. Key Laboratory of Micro-inertial Instrument and Advanced Navigation Technology of Ministry of Education, Nanjing 210096, China*)

Abstract: Aiming at the shortcomings of MAPF in the scenarios of anonymous cooperative path finding for multiple carlike robots, including long path and poor efficiency, a cooperative goal allocation and path finding algorithm for multiple unmanned ground vehicles, Nutcracker-CBS, is proposed. Firstly, a tightly coupled MAPF framework with goal allocation is constructed to optimize goal allocation and path finding jointly. In goal allocation module, an improved nutcracker optimization algorithm is proposed to solve goal allocation incrementally, which can shorten the duration of the module. In path finding module, an improved MAPF algorithm is proposed with traceback constraint construction mechanism, bypass mechanism for the length estimation of collision-free path, and low-level path planning mechanism for data sharing, to enhance efficiency and quality of path finding. In benchmark experiments, time cost of Nutcracker-CBS is reduced by 90.37% compared to SOTA algorithm. Time consumption of goal allocation module is reduced by 86.76% compared to the original algorithm. MAPF module completes the path construction of 100 unmanned ground vehicles within 6 seconds, with the reduction of average path length by 6.058%. Filed test shows that the total path length is reduced by 55.26% and the total time consumption of the system is reduced by 61.29%, which boost the efficiency of multi-robot system and decrease path length.

Keywords: cooperative path finding; MAPF; conflict-based search; goal allocation

0 引 言

多智能体路径搜索(multi-agent path finding, MAPF)是人工智能和机器人领域的研究热点之一,旨在同时找到多个机器人的无碰撞路径,确保多机器人协同系统的高效运作,在水下潜航器环境勘察[1],无人机集群[2],机械臂协同[3]等领域得到广泛应用,如何提升求解效率和路径质量是当前的研究热点。

早期多机器人系统主要采用单机器人路径规划方法,辅以 ε 协作避碰(ε-cooperative collision avoidance, εCCA)[4],缓冲维诺细胞(buffered voronoi cell, BVC)[5]

收稿日期:2023-12-28 Received Date: 2023-12-28

等分布式避障思想初步实现多机器人路径构建。虽然避免了机器人之间碰撞,但由于缺乏全局信息,分布式算法易出现死锁,扩展性较差。因此现阶段研究偏向于集中式算法。现有集中式 MAPF 求解方法中,图搜索算法基于最佳优先思想搜索路径,扩展性良好,逐渐成为多智能体路径联合搜索的主流算法,如多路斯坦纳算法(multi Steiner*,MS*)[6],安全间隔路径规划算法(safe-interval path planning,SIPP)[7],基于冲突的搜索算法(conflict-based search,CBS)[8]等。其中 CBS 算法采用两层算法框架,上层构建一个二叉数,用于查找机器人之间的冲突并构建约束,解决冲突,下层采用改进的 A * 算法,查找满足上层所构建约束的路径,上下层迭代执行,确保所构建的路径无碰撞。CBS 算法总能获得最优解,对 MAPF 问题具有完备性[8],近年来涌现出大量的改进方案。智能体合并策略[9]根据随机准则,冲突次数准则,关键冲突准则将机器人合并为小组,然后逐级构建路径并解决冲突,当机器人较少时在复杂环境中效率较高。然而解决不同小组之间的冲突计算量仍较大,因此该策略对于大规模 MAPF 问题的求解效率仍然低下。文献[10]将轻量化 D*(D*-lite)作为下层路径规划器,增量式求解 MAPF 问题,从而避免了在环境改变时重新规划所有机器人路径,减轻了计算负担。针对连续时间空间的 MAPF 问题,文献[11]受 SIPP 算法启发,提出连续时间 CBS(continuous-time CBS, CCBS)和 SAT 模理论 CCBS(SAT modulo theory CCBS,SMT-CCBS)算法,实现在连续空间的多智能体路径联合寻优。启发式算法[12]通过将冲突构成图,得到上层最佳优先搜索的可纳启发值,虽然提升了路径质量,减少了长度总和,但时效性较差。为提升算法效率,有学者[13]采用次优思想求解改进 CBS 算法,以牺牲求解质量为代价,迅速得到可行解,从而实现实时性应用。然而所得结果路径总和较长,避障路径弯曲多变,对机器人执行器件性能要求较高。文献[14]采用次优求解 MAPF 的同时,引入绕道和优先级策略,提升了避让策略的平顺性,降低了路径长度,但是没有保证完备性,在机器人数量庞大时,不合理的求解顺序反而带来了更多的冲突和负担。综上,现有文献解决多机器人路径构建问题,无法兼顾高效率和高质量。

对于多机器人系统,为提升路径规划算法效率,缩减路径长度,除了优化各机器人路径之外,还可以在目标点分配方面进行优化。比如物流分拣[15],无人机群搜救[16]等场景向多机器人系统指派了若干导航目标点,而各目标点具体由哪一个机器人前往则由系统求解决定。合理的目标点分配可以减少机器人之间的冲突,缩减路径长度,提高求解效率,减少执行过程的用时。这种无需人为指定各机器人导航目标点的 MAPF 算法称为非指定式 MAPF,它能从目标点分配和路径构建两方面优化计算效率,减少冲突并缩减路径长度总和。文献[17]提出了一种非指定式多智能体路径规划算法,通过穷举不同分配和避让路径的方案,并排序得到最优。然而搜索空间的爆炸式增长导致其效率低下。元启发式算法由于效率高,寻优能力强,成为当前研究热点。近年来涌现出大量新算法[18-19],并被广泛应用于路径规划问题[20-21]。目标点分配作为一种整数规划问题,也有望采用元启发算法解决,并融入 MAPF 框架中,随着搜索过程增量式优化分配方案,提高求解效率和目标点分配合理性。

本文针对现有多智能体路径规划无法兼顾效率和质量的问题,提出了一种面向多无人车的目标点分配和协同路径规划算法,主要贡献有:1)提出一种紧耦合目标点分配 MAPF 框架,实现目标点分配和路径构建的联合寻优。2)提出改进的星鸦优化算法(nutcracker optimization algorithm, NOA),增量式地高效求解软约束构型的目标点分配问题,减少计算开销;3) 提出回退式约束构建机制,引入避碰路径估计的绕道机制和数据共享底层路径规划机制,提升路径构建模块求解效率和路径质量。

1 紧耦合目标点分配 MAPF 框架

本章构建一种紧耦合的目标点分配 MAPF 框架 Nutcracker-CBS,在多车路径构建过程中及时检测到方案的不合理性,调整目标点分配,提升算法求解效率和质量。

如图 1 所示,Nutcracker-CBS 算法框架由结点,二叉搜索树和搜索森林三个层面构成。结点代表一种多智能体路径规划方案,主要记录了其受到的约束(constraints),各机器人路径(Ps),目标点分配(Ta),各机器人路径代价总和(Sc),以及机器人规划顺序(agentorder)。每棵二叉搜索树代表一种目标点分配方案,由结点分裂扩展而成。算法在分裂结点时,总是继承其根结点的目标点分配方案,并调整约束和路径,从而求解在该目标点分配方案下的无碰撞路径。搜索森林包括多棵搜索树,以此存储不同的目标点分配方案,并对不同目标点分配方案下的所有已发现但未被处理的结点统一排序,实现目标点分配和路径规划的联合寻优。当智能体与目标点对数为 N 时,根据全排列原则,有 A_N^N 种目标点分配方案,因此森林中最多 A_N^N 棵搜索树。

算法伪代码如算法 1 所示,输入为目标点集合 goal 和当前时刻各机器人位姿 pos,输出为最优方案结点 p。算法包括初始化(1~9 行)和主循环(10~38 行)两个阶段。

在初始化阶段,算法构造一个不包括任何约束的空结点(1 行),并以欧氏距离作为各机器人与各目标点之

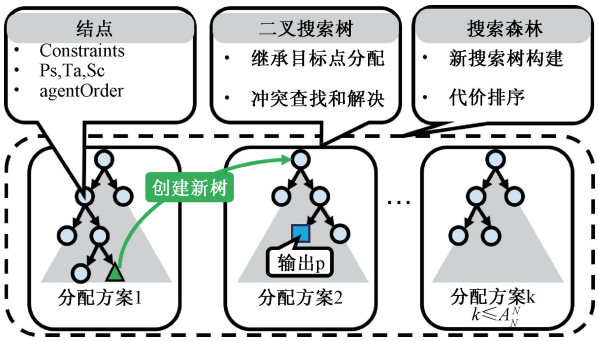


图 1　Nutcracker-CBS 模块组成

Fig. 1　Nutcracker-CBS components

间的距离估计(2 行),以此为依据分配各机器人目标点(3 行),并按顺序完成各机器人路径构建(4~7 行),求出各机器人路径代价总和(8 行),并将结点加入到搜索森林 Forest 中(9 行),以其作为第一棵搜索树的根结点。其中,结点包含数据内容如图 1 所示;第 3 行 NOA_Allocation 采用第 2 章所提改进的星鸦优化算法求解邻接矩阵 E 下的目标点分配;第 4 行 OrderList 将机器人路径构建顺序指定为 0~N-1,以便采用 3.3 节所述数据共享机制构建各机器人路径;第 6 行路径构建算法是 CBS 的下层模块,本文采用引入避碰路径估计的绕道机制和数据共享底层路径规划机制,在路径构建阶段减少机器人之间冲突,减轻上层算法的负担;第 9 行 Forest 由小顶堆实现,以结点中的路径代价总和(Sc)为指标,对结点排序。

在主循环阶段,算法将 Sc 最小的结点作为当前处理结点(11 行)。路径中的避碰和绕道导致长度相较于欧氏距离显著提升,因此算法在初始化阶段构建的邻接矩阵与真实数值差距悬殊。为此,算法在主循环中,根据当前结点中的路径长度更新邻接矩阵(12 行),从而得到各机器人与不同目标点之间距离的准确值,并重构目标点分配方案(13 行)。之后,算法分别检测当前结点的路径是否存在冲突(flag1),目标点分配方案是否需要调整(flag2),并在本搜索树中扩展结点(18~27 行),或构建新的搜索树根结点(28~37 行)。对于存在冲突的结点,算法构建约束并将其分裂为两个子结点(19 行)。子结点继承当前结点的目标点分配方案和约束,并按照 3.1 节所述方法加入新的约束,按照 3.3 节所述方式确定路径规划顺序并完成该结点中各机器人路径构建,最后将子结点加入 Forest 中。若目标点分配方案与结点中记录的方案不同,说明需要调整方案并构建新的搜索树(28~37 行)。算法以更新的邻接矩阵构建新根结点的目标点分配,并将该根结点加入到 Forest 中,生成新搜索树,如图 1 三角形结点所示。对于目标点分配方案无需调整且不存在冲突的结点 p,将被算法输出,如算法 15~17 行和图 1 中方形结点所示。

如上所述,本文后续章节中,第 2 章对应算法 1 中的 NOA_Allocation,采用改进的星鸦优化算法求解目标点分配;第三章对传统 CBS 的上下层做出改进,其中 3.1 节对应 19 行 ConstructConstraints,采用路径回退式约束构建机制,提升冲突解决能力;3.2 和 3.3 节对应算法 1 中的 PathPlanning,采用引入避碰路径估计的绕道机制和数据共享底层路径规划机制,构建高效的路径搜索模块,在路径构建阶段减少潜在的冲突,从而提升搜索效率,降低路径长度。

算法 1:Nutcracker-CBS;输入:goal, pos;输出:p

```
1: p=emptyNode( ) //空结点
2: E=EuclideanDis(goal, pos) //欧氏距离构建邻接矩阵
3: p. Ta=NOA_Allocation(E) //第 2 章目标点分配模块
4: p. agentOrder=OrderList(0, 1, agentN) //路径构建顺序
5: For a in p. agentOrder
6:     PathPlanning(Env, goal, pos, E, p, a) //3.2,3.3 节底层路径规划
7: EndFor
8: p. Sc=sumOfCosts(p) //路径代价总和
9: Forest. insert(p) //第一棵树的根结点
10: While not Forest. empty( )
11:     p=Forest. top( ); Forest. pop( ) //取出搜索森林中代价最小结点
12:     E=UpdateE(E, p. Ps) //根据路径更新邻接矩阵
13:     Ta=NOA_Allocation(E) //第 2 章目标点分配模块
14:     flag1=FindConflict(p. Ps); flag2=(Ta ! =p. Ta)
15:     If (! flag1 && ! flag2)
16:         return p //路径不存在冲突,目标点分配无需调整
17:     EndIf
18:     If(flag1) //路径存在冲突
19:         plist=ConstructConstraints(p. Ps) //3.1 节约束构建和分裂
20:         For subp in plist //遍历分裂出的若干子结点
21:             For a in subp. agentOrder
22:                 PathPlanning(Env, goal, pos, E, subp, a)//3.2,3.3 节
23:             EndFor
24:             subp. Sc=sumOfCosts(subp) //路径代价总和
25:             Forest. push(subp) //子结点加入搜索森林
26:         EndFor
27:     EndIf
28:     If(flag2) //目标点分配不合理
29:         p=emptyNode( ) //新结点
30:         p. Ta=Ta //重构目标点分配
```

```
31:          p. agentOrder = OrderList(0, 1, agentN) //路径构建顺序
32:          For a in p. agentOrder
33:              PathPlanning(Env, goal, pos, E, p, a) // 3.2, 3.3 节
34:          EndFor
35:          p. Sc = sumOfCosts(p) //路径代价总和
36:          Forest. insert(p) //新树的根结点
37:      EndIf
38: EndWhile
```

2 目标点分配模块构建

传统目标点分配采用匈牙利算法(Hungarian algorithm),通过矩阵变换得到最佳一对一分配,由于无法随 Nutcracker-CBS 主循环增量式计算,当机器人数量的增长时,计算时耗显著增加,因此不适用于 Nutcracker-CBS 框架。为此,本章首先构建软约束形式的一对一目标点分配模型,然后基于改进星鸦优化实现增量式求解。本章对应算法 1 中 NOA_Allocation 模块,以临界矩阵 E 为输入。

2.1 软约束目标点分配目标函数构建

本文假定无人车与目标点数量均为 N,构成一对一目标点分配。所述紧耦合 Nutcracker-CBS 目标函数为:

$$\min J = \sum_{i}^{N}\sum_{j}^{N} E_{ij}X_{ij} + J_{c1} + J_{c2}$$

$$J_{c1} = \sum_{i}^{N}\left\| \sum_{j}^{N} X_{ij} - 1 \right\| \tag{1}$$

$$J_{c2} = \sum_{j}^{N}\left\| \sum_{i}^{N} X_{ij} - 1 \right\|$$

其中 E_{ij} 表示无人车 i 与目标点 j 之间距离,算法 1 中首先以欧氏距离作为其估计值,并随主循环(10 ~ 38 行)的后续路径搜索不断更新并调整分配。X_{ij} = 0/1 为决策变量,决定是否由无人车 i 执行任务 j。目标函数 J 的第一项表示在目标点分配方案 X 下各无人车路径长度总和,后两项 $J_{c1,2}$ 作为模型软约束被加入到目标函数中,表示无人车与目标点一一对应。通过使优化目标的最小化,能获得路径长度总和最小的一对一目标点分配方案。

2.2 基于改进星鸦优化的目标点分配模块

星鸦优化算法(nutcracker optimization algorithm, NOA)于 2023 年被 Mohamed 等人提出[19],在多个标准测试模型中取得良好的成绩,有巨大的应用潜力。然而星鸦算法适用于连续问题的数值优化,上述目标函数属于整数规划离散问题,虽然解的每个分量取值为 0 或 1,较连续问题稀疏,但解空间维度很高,为 N^2,限制了算法的收敛速度。因此本文对 NOA 的觅食与存储,找回与恢复两组策略进行改进。其中每组策略各包含勘探和开发两个阶段,分别记为勘探 1,开发 1,勘探 2 和开发 2。

1)勘探 1:该阶段主要用于确保种群更加均匀地分布在搜索空间中,提升算法对全局环境的搜索能力

$$\vec{\boldsymbol{X}}_i^{t+1} = \begin{cases} \vec{\boldsymbol{X}}_{i,j}^{t}, & \tau_1 < \tau_2 \\ rand(N^2) \geqslant 0.5, & \text{其他} \end{cases} \tag{2}$$

式中:$\vec{\boldsymbol{X}}_i^{t+1}$ 为星鸦个体 i 在 $t+1$ 代的首个坐标,也即优化变量的更新值;$\vec{\boldsymbol{X}}_{i,j}^{t}$ 为在当前迭代轮次 t 下第 i 个星鸦个体的第 j 个坐标;第二行针对离散空间,按照均匀分布产生包含 N^2 个 0 或 1 的序列,作为下一代种群位置。$\tau_1, \tau_2 \sim U(0,1)$ 为随机数,决定采用当前代的解或随机采样产生下一代。

2)开发 1:该阶段将种群的位置朝当前最优解更新,以加速算法收敛。

$$\vec{\boldsymbol{X}}_i^{t+1(new)} =$$

$$\begin{cases} \vec{\boldsymbol{X}}_i^{t} \oplus ((rand(N^2) < \mu) \& (\vec{\boldsymbol{X}}_{best}^{t} \oplus \vec{\boldsymbol{X}}_i^{t})), & \tau_1 < \tau_2 \\ \vec{\boldsymbol{X}}_{best}^{t} \oplus (rand(N^2) < l), & \text{其他} \end{cases} \tag{3}$$

式中:$\vec{\boldsymbol{X}}_i^{t+1(new)}$ 表示在当前代数 t 下星鸦在存储区域中新的位置,$\vec{\boldsymbol{X}}_{best}^{t}$ 为截止当前代数 t 的最优解,$\vec{\boldsymbol{X}}_i^{t}$ 为当前代数 t 下星鸦种群位置。$\oplus$ 表示逻辑异或,μ 为随机数,l 随着 NOA 迭代线性递减至 0。式(3)按概率选择两种向当前代最优解靠近的方式,第一种在当前个体的基础上按照概率 μ 修改向量的某几维,使得当前个体在下一代更加接近当前代的最优解 $\vec{\boldsymbol{X}}_{best}^{t}$;第二种则直接基于当前最优解产生下一代的个体,它以概率 l 沿用了当前代最优解的某些维度。

3)勘探 2:该阶段用于检测前几代有可能存在最优解的区域,并将其作为当前代的参考点 $\overrightarrow{\boldsymbol{RP}}_i^{t}$。

$$\overrightarrow{\boldsymbol{RP}}_i^{t} =$$

$$\begin{cases} \vec{\boldsymbol{X}}_i^{t} \oplus ((rand(N^2) < \alpha) \& (\overrightarrow{\boldsymbol{RP}}_i^{t-1} \oplus \vec{\boldsymbol{X}}_i^{t})), & \theta = \pi/2 \\ \vec{\boldsymbol{X}}_i^{t} \oplus ((rand(N^2) < \alpha \cdot |\cos(\theta)|) \& (\vec{\boldsymbol{X}}_A^{t} \oplus \vec{\boldsymbol{X}}_B^{t})), & \text{其他} \end{cases} \tag{4}$$

$$\alpha = \begin{cases} (1 - t/T_{\max})^{2t/T_{\max}}, & r_1 > r_2 \\ (t/T_{\max})^{2/t}, & \text{其他} \end{cases}$$

式中:$T_{\max}$ 为最大迭代轮数,$r_1, r_2 \sim U(0,1)$,$\theta \sim U(0,\pi)$ 为随机数,$\vec{\boldsymbol{X}}_A^{t}$ 和 $\vec{\boldsymbol{X}}_B^{t}$ 为在迭代次数 t 下,随机选取的两个不同于 i 的星鸦个体的位置,$\overrightarrow{\boldsymbol{RP}}_i^{0}$ 被初始化为 $\vec{\boldsymbol{X}}_i^{0}$。采用两种策略产生本轮迭代的参考点 $\overrightarrow{\boldsymbol{RP}}_i^{t}$。第一种策略参考点基于以上一轮参考点为基准,按照概率 α 修改若干个元素,使当前参考点接近上一轮参考点;第二种则比较种群中其

余两个体 A 和 B,按照概率叠加两个体之间的差异元素产生参考点$\overrightarrow{\boldsymbol{RP}}_i^t$。勘探 2 阶段则按概率决定是否继续当前方向的搜索,或将搜索重心放到新的参考点$\overrightarrow{\boldsymbol{RP}}_i^t$ 处,这两个阶段的引入,可以有效帮助算法跳出局部最优。根据参考点,构建勘探 2 更新策略:

$$\vec{\boldsymbol{X}}_{i,j}^{t+1} = \begin{cases} \vec{\boldsymbol{X}}_{i,j}^t, & \tau_3 < \tau_4 \\ \vec{\boldsymbol{X}}_{i,j}^t \oplus ((rand(N^2) < r_1) \oplus (\overrightarrow{\boldsymbol{RP}}_t^t \oplus \vec{\boldsymbol{X}}_{C,j}^t)), & \text{其他} \end{cases} \tag{5}$$

式中:$\tau_3,\tau_4 \sim U(0,1)$ 为随机数,$\vec{\boldsymbol{X}}_C^t$ 为在迭代次数 t 下,随机选取的两个不同于$\vec{\boldsymbol{X}}_i^t$,$\vec{\boldsymbol{X}}_A^t$ 和$\vec{\boldsymbol{X}}_B^t$ 的星鸦个体的位置。

4)开发 2:该阶段通过比较参考点与当前个体的优劣决定下一轮个体:

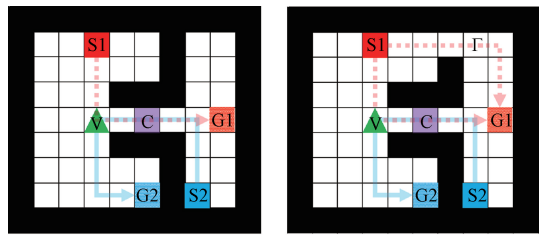
$$\vec{\boldsymbol{X}}_i^{t+1} = \begin{cases} \vec{\boldsymbol{X}}_i^t, & J(\vec{\boldsymbol{X}}_i^t) < J(\overrightarrow{RP}_i^t) \\ \overrightarrow{\boldsymbol{RP}}_t^i, & \text{其他} \end{cases} \tag{6}$$

其中 J 为目标函数(1)。由于逻辑运算在二值状态情景更新速度快,且能确保更新的解落在解的取值空间中,因此更加适用于目标点分配问题求解。另一方面,基于优化的思想求解目标点分配,当初始值接近全局最优时,算法能够很快到达最优解。所提 Nutcracker-CBS 算法框架每次循环求解目标分配时,采用上一轮迭代作为初值,在更新的 $\boldsymbol{E}$ 矩阵下求解目标分配问题(1)。由于 $\boldsymbol{E}$ 矩阵由 MAPF 模块构建的路径长度更新,其元素在后续循环中变化相对较小,因此下一轮目标点分配只需要对方案进行微调,其初值接近最优解。除了首次求解目标分配需要从远离最优的位置开始寻优,之后的每一次的初值都接近最优,因此相比每次都需要大量矩阵变换的匈牙利算法,求解时间得到有效缩短。

3　多智能体路径规划算法模块构建

经典 CBS 算法存在约束构建缺乏系统性,路径构建无法估计避碰代价,不同智能体路径构建互不考虑的问题,在狭窄空间中方案构建效率低下,路径冗长。为此本章分别构建回退式约束构建机制,引入避碰路径估计的绕道机制和数据共享底层路径规划机制,提升搜索效率和质量。

冲突分类处理有助于 MAPF 算法的分析和改进。文献[9]将 MAPF 中的冲突分为关键冲突,半关键冲突和非关键冲突,并以此作为分组依据,分解原 MAPF 问题。如图 2 所示,S_i 为无人车 i 起点,G_i 为终点,无人车 1 最短路径如细点画线箭头所示,无人车 2 最短路径则如实线箭头所示,C 为两车冲突的位置。关键冲突如图 2(a)所示,必须某一无人车选择较远路径绕道或等待一段时间才可避免;半关键冲突则如图 2(b)所示,可以选择路径长度相同的其他路径 Γ 绕开。非关键冲突能够被传统 CBS 算法高效处理,因此不再讨论。文献[9]按冲突类别分组完成路径构建,但未针对冲突类型给出优化方案。如果分别针对两种冲突优化避让策略,能够提升算法效率,降低路径长度。



(a) 关键冲突　(b) 半关键冲突
(a) cardinal conflict　(b) semi-cardinal conflict

图 2　关键冲突和半关键冲突

Fig. 2　Cardinal conflict and semi-cardinal conflict

3.1　路径回退式约束构建机制

关键冲突会产生大量的结点分裂和时耗。传统 CBS 所构建的避碰机制路径冗余,难以执行,原因在于算法总是仅针对当前冲突位置构建约束,而大多数冲突是由于之前某个位置策略不当所导致。因此若提前若干位置产生避让分支路径,一方面避免了在狭窄区域构建绕道策略的难题,另一方面增加避碰分支路径与冲突位置的距离,便于底层算法得到符合车辆运动学约束的路径。因此本节提出一种新型约束构建策略,沿车辆路径回退若干路径点构建约束,提升 CBS 算法避碰策略的搜索效率。本节对应算法 1 的 19 行 ConstructConstraints 模块,首先构建两个新的结点,继承输入结点 p 的目标点分配方案以及约束,然后按照下列方法构建新的约束。

当检测到智能体之间冲突时,算法沿着出现冲突的两个智能体路径回退 λ 个航迹点,其中 λ 为空间可行且无人车机动能力可行的数值:

$$\lambda = \max(\lambda_1,\lambda_2) \tag{7}$$

式中:λ_1 为从冲突点出发,沿路径回退并到达首个度大于 2 的路径点所需移动的点数,后文将该路径点称为可交换点。该值保障了避让分支的空间可行性。λ_2 根据车辆尺寸和机动性能确定,使得无人车能够完成避让机动,确保了无人车机动能力可行性。对于可原地旋转的差速驱动无人车,λ_2 被构建为车辆直径 D 与航迹点间隔长度 Δx 之比并取整:

$$\lambda_2 = ceil(D/\Delta x) \tag{8}$$

若出现碰撞的航迹点下标为 i,则约束为:

$$\boldsymbol{C} = \{x \mid x = x_{i-\lambda}, \cdots, x_i\} \times \{t \mid t = t_{i-\lambda}, \cdots, t_i\} \tag{9}$$

式中：$\{x \mid x = x_{i-\lambda}, \cdots, x_i\}$ 为从第 $i-\lambda$ 到第 i 个航迹点的位置，$\{t \mid t = t_{i-\lambda}, \cdots, t_i\}$ 为对应的时间戳。以两者的笛卡尔积构造约束，确保在该时间范围内智能体不进入该区域，以等待另一智能体走出该区域。如图 2(a)所示，两无人车按照先构建的路径将会在栅格 C 出现碰撞，因此在上层算法分裂的两个子结点分别沿两个无人车的路径回退。对于沿无人车 1 回退的子结点，找到首个可交换点如图中三角形 V 所示。因此该结点所构建的方案要求无人车 1 在到达 V 之前等待，直到无人车 2 通过狭窄区域，从而只需一次结点分裂和约束构建即可完成冲突避让，提升算法效率，避免了不当的避让策略导致的冗余路径。

3.2 引入避碰路径估计的绕道机制

半关键冲突虽然存在绕道路线，但传统 CBS 仅通过碰撞位置处的约束强制下层路径规划避让某一位置，导致底层算法无法正确理解约束并从根本上构造无碰撞路径。如果在下层路径搜索即将到达碰撞位置时估计出由于避碰带来的代价值并及时调整路径，能够带来更高的求解效率和更短的路径总和。

当下层路径规划搜索到上层算法标记的可交换点 $x_{i-\lambda}$ 时，避碰代价被构建为：

$$p = \begin{cases} \lambda - (t_{i-\lambda} - t), & 0 \leqslant t_{i-\lambda} - t \leqslant \lambda \\ 0, & \text{其他} \end{cases} \tag{10}$$

式中：$t_{i-\lambda}$ 为按照原路径到达可交换点的时间戳，t 为修正路径到达可交换点时间戳。通过避让代价估计值 p 获知修正路径在该点需要等待的时间，从而决定调整方案或继续等待。如图 2(b)所示，上一轮路径搜索出现冲突并得到了三角形 V 所示可交换点，在本轮智能体 1 路径通过该点时，算法比较该点代价总和，并决定重新选择路径 Γ 绕道，从而以更小的代价避免冲突，减少多智能体系统的路径总和，提升算法效率。

3.3 数据共享底层路径规划机制

在底层路径搜索中也考虑到其余智能体的影响，可以显著减轻上层模块的负担，提高效率。为此，本节在底层路径构建算法中加入无人车之间的距离惩罚项，在栅格点选择时降低多机器人冲突的概率，从而尽可能构建出无碰撞路径。

无人车之间距离惩罚项被构建为：

$$c_{ij}(t_k) = \begin{cases} \tan\left(\dfrac{s - d_{ij}}{s} \cdot \dfrac{\pi}{2}\right), & 0 < d_{ij} \leqslant s \\ 0, & \text{其他} \end{cases} \tag{11}$$

式中：$d_{ij} = \| x_i - x_j \|$，$j \neq i$ 为无人车 j 与当前车辆 i 之间的距离，s 为安全阈值。无人车路径为时间的函数，因此惩罚项与时间戳有关。对于某一距离小于阈值 s 的路径点，该时刻距离惩罚项将显著增长，从而使算法放弃该路径点，避免无人车之间的碰撞。

综合 3.2 和 3.3 节，在底层路径构建算法中，引入避碰路径代价和机器人间距离惩罚项，改进后的 A* 扩展路径栅格点的启发值为：

$$f_i(t_k) = h_i + g_i + p + \sum_{j \neq i} c_{ij}(t_k) \tag{12}$$

式中：i 为当前机器人编号，k 为路径扩展的时间戳，h 和 g 为原 A* 算法的距离启发项和累计代价项。利用改进的启发值计算方式，求解路径规划问题，作为算法 1 的 PathPlanning 模块。

上述方案将导致先构建路径的无人车将具有更高的优先级，而之后的无人车将以先形成的路径为参考，构建自己的路径。只有按照合理的顺序完成多车路径构建，才能为整个多车系统找到最优解。为此，上层的 CBS 搜索树的结点应包含路径构建顺序信息，并确保分裂的两个子结点的路径规划顺序与约束构建的优先级一致。如某结点的路径中，t_i 时刻，在 x_i 处，机器人 a_j 和 a_k 存在冲突：

$$Conflict(t_i, a_j, a_k, x_i) \tag{13}$$

则分裂的左右两个子结点分别对机器人 a_j 和 a_k 产生约束。在左子结点中，路径构建顺序沿用当前结点，但将机器人 j 顺序调整到 k 之前；右子结点则将机器人 j 调整到 k 之后。改变顺序以得到不同避让策略，从而找到最佳方案。

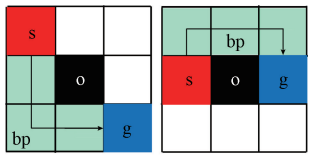
4 时间复杂度分析

本章分析算法 1 的时间复杂度，它主要受 NOA_Allocation 模块，PathPlanning 和主循环(10～38 行)的影响。

对于 NOA_Allocation 模块，根据文献[19]的算法 3，其时间复杂度可以表为 $\boldsymbol{O}(I_{\max} N_{\text{noa}} d_{\text{noa}})$，其中 $I_{\max}$ 为指定 NOA 的迭代次数，N_{noa} 是 NOA 种群个体，d_{noa} 是指自变量维度。第二章所述改进 NOA 针对整数规划问题，将自变量各维度由连续数值变为离散值 0 和 1，其收敛速度得到显著提升，因此可以指定较小的迭代次数 $I_{\max}$，从而降低目标点分配模块的时间复杂度。

PathPlanning 模块时间复杂度主要与障碍物密度，机器人与目标点对数量 N 有关。障碍物密度 ρ 为障碍物数量与总栅格数量之比，衡量了遇到障碍物的概率。路径搜索时耗与访问的栅格点数呈正比，因此障碍物带来的绕道将增加计算时长。在栅格地图中，绕道存在如图 3(a)、(b)的两种情况，当不存在障碍物 o 时，路径从 s 通过 o 所在位置连接到 g，其长度为 3；存在障碍物 o 时，绕道路径 bp 总长度为 5。因此由于绕过 o 增加的路

径点数为 2。假设障碍物在地图中均匀分布,且每个障碍物仅占据一个栅格,智能体 i 与其目标点之间栅格欧氏距离为 d_i。则机器人 i 路径中遇到障碍物数量的期望为 ρd_i,绕道带来的路径增长为 $2\rho d_i$,实际路径长度的期望为 $2\rho d_i + d_i$。因此 PathPlanning 模块时间复杂度为 $O(\rho d_i)$,则在算法 1 的 5~7 行,21~23 行和 32~34 行,时间复杂度均为 $O(N\rho d)$,其中 $d = \sum_{i=1}^{N} d_i/N$ 为各机器人与目标点之间欧氏距离的平均值。合理的目标点分配能够减少平均路径长度 d,从而降低 PathPlanning 的时间复杂度。



(a) 绕道情况1　(b) 绕道情况2
(a) detour case 1　(b) detour case 2

图 3　绕道带来的路径长度增加

Fig. 3　Path length increased by detour

算法 1 的主循环包括了 NOA _ Allocation 和 PathPlanning 两个耗时模块,而循环次数为 Forest 包含的结点数量 N_{forest},因此算法 1 时间复杂度为:

$$O((I_{\max}N_{\text{noa}}d_{\text{noa}} + N\rho d)N_{\text{forest}}) \tag{14}$$

由于本文算法采用了路径回退式约束构建机制,减少了为解决冲突而带来的结点数量;引入避碰路径估计的绕道机制以及数据共享底层路径规划机制,在路径规划层面减少了冲突数量,从而减少了搜索树分裂的结点数量 N_{forest}。综上,本文所述算法 1 通过减少 NOA _ Allocation 模块迭代次数 $I_{\max}$, PathPlanning 模块的平均路径长度 d 和主循环搜索树的结点数量 N_{forest},降低了求解非指定式 MAPF 问题的时间复杂度,提升求解效率。

5　实验与分析

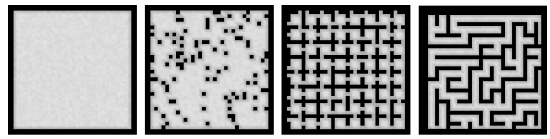
为验证所提出算法在目标点分配和轨迹规划的质量和效率,本章使用 C++完成算法编排,分别采用数据集实验和实际实验验证算法性能。

5.1　数据集实验

本节在 MAPF 标准数据集[22]上测试 Nutcracker-CBS 的效率和路径长度。首先与当前非指定 MAPF 的 SOTA 算法对比,验证所提紧耦合 MAPF 框架在求解非指定 MAPF 问题中求解效率和路径质量的提升。由于非指定 MAPF 算法的效率和路径质量取决于所提出的目标点分配和路径规划模块,因此之后进行两个模块替换实验,分别将所提算法 1 中的目标点分配模块、MAPF 模块替换为已有算法,对比验证所提模块的有效性。

(1)数据集说明

MAPF 的标准数据集[22]包含多种尺寸的栅格地图,本节选其中的空地图(empty32),随机地图(random32),室内场景(room32),迷宫场景(maze32)作为算法验证环境,依次如图 4(a)到(d)所示。文献[22]认为,图 4 四幅地图对于 MAPF 算法的挑战性从左到右逐渐增加。



(a) 空地图　(b) 随机地图　(c) 室内场景　(d) 迷宫场景
(a) empty32　(b) random32　(c) room32　(d) maze32

图 4　数据集地图

Fig. 4　Dataset map

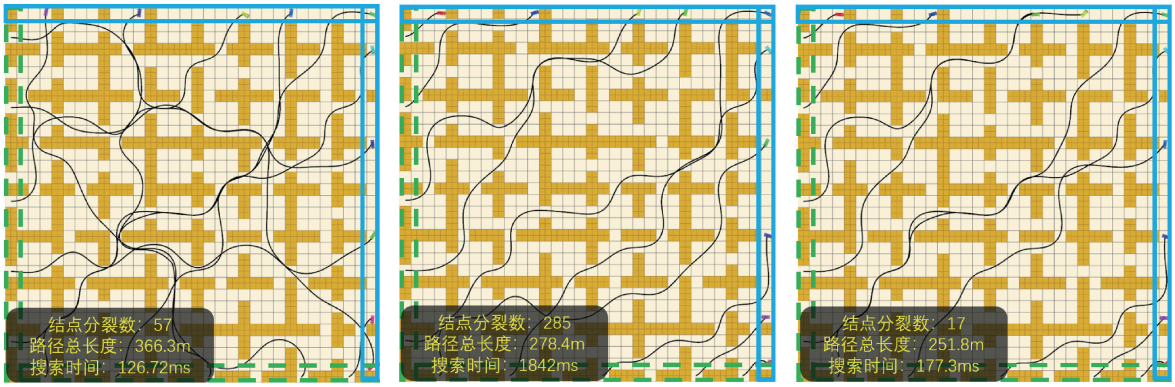
(2)非指定 MAPF 算法实验

为测试 Nutcracker-CBS 算法在处理非指定 MAPF 问题的求解能力,本节在数据集上分别测试 ECBS[8], TAPF[17]和 Nutcracker-CBS。ECBS 算法随机指派各无人车目标点,不具备目标点分配能力;TAPF 穷举不同目标点分配和路径,是当前非指定 MAPF 的 SOTA 算法。如图 5 所示,在 Room32 地图中,10 个机器人分布于左侧和下方虚线框区域,指派的任务位于右侧和上方的实线框区域,分别采用上述 3 种算法完成目标点分配和路径构建,结果依次如图 5(a)到(c)所示。

由图 5(a)知,ECBS 算法随机分配路径长度,由于没有求解目标点分配问题,算法耗时最小,但随机分配目标点产生了大量路径开销,增加了智能体之间的冲突,路径总长度最长。图 5(b)中 TAPF 和图 5(c)中的 Nutcracker-CBS 在该案例中构建路径总长度相近,但由于 TAPF 实质上按照穷举的方式搜索不同分配方案和避让策略,构建智能体路径,虽然得到了非指定 MAPF 的最优解,但是产生了大量结点分裂,算法效率低下,无法满足实时性应用。Nutcracker-CBS 路径长度与 TAPF 相当,同时算法求解效率较高,时效性良好(177.3 ms),相比 TAPF(1 842 ms)求解时间缩减了 90.37%,适用于真实机器人的应用,能够根据实时更新的环境地图在线重规划,从而说明所提出紧耦合目标点分配 MAPF 框架的有效性。

(3)目标点分配模块收敛速度实验

为验证目标点分配模块(NOA _ Allocation)的有效性,本实验在 room32 地图 N = 100 时,将 NOA _ Allocation 替换为匈牙利算法(Hungarian)和原 NOA 算法,对比模块求解速度。Hungarian 利用矩阵变换得到最佳一对一分配,是经典的目标点分配算法,然而由于每次都需要重新计算,无法增量式求解,不适用于 Nutcracker-CBS 框架。



(a) ECBS规划结果
(a) results of ECBS

(b) TAPF规划结果
(b) results of TAPF

(c) 本文算法规划结果
(c) results of proposed method

图 5 不同算法在 room32 地图 $N=10$ 非指定 MAPF 实验中的结果

Fig. 5 Results of different algorithms in unlabeled MAPF experiment in room32 when $N=10$

在算法 1 主循环的第一、二轮中，目标点分配模块用时 t1，t2 如表 1 所示。在首轮循环中，由于初值远离最优解，两种基于优化的目标点分配算法（NOA 和改进 NOA）均消耗了大量的求解时间，NOA 与 Hungarian 相当，而改进 NOA 则显著提高了效率；而在第二轮和之后的循环中，以上一轮目标点分配解作为初值，由于接近最优解，基于优化的算法求解时长明显小于 Hungarian 算法。而 Hungarian 算法每轮循环都需要重新利用矩阵变换求解，无法在初值的基础上优化，因此在之后的循环中，效率仍然低下。改进的 NOA 算法状态更新策略更适用于整数规划问题，求解时耗（17.01 ms）相比原算法（128.5 ms）减少了 86.76%，在之后的循环中具有极高的求解效率。

表 1 $N=100$，room32 地图下目标点分配算法耗时

Table 1 Time of different goal allocation algorithms in room32 when $N=100$ ms

算法	t1	t2
Hungarian	284.7	287.2
NOA	296.6	128.5
改进 NOA	45.39	17.01

第一轮循环中 NOA 和改进 NOA 的代价值（路径长度总和）随迭代次数变化曲线如图 6 所示。改进 NOA 随着迭代次数增长，代价逐渐减少，最终代价约为 2 450，与 NOA 算法最终代价值相近，说明基于逻辑运算作为状态更新的改进策略能够确保算法求得最优解。另一方面，原 NOA 在连续空间求解高维的整数规划问题时，算法收敛速度明显较慢，需迭代 491 代才能达到全局最优，而改进算法只需 73 代即可到达，可见采用逻辑运算改进状态更新策略的有效提升了该问题的求解速度。综上，本文所提目标点分配模块有效提升了目标点分配算法求解效率。

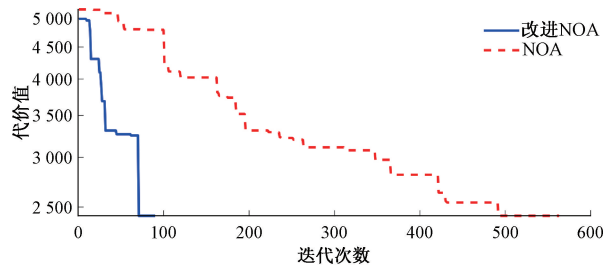


图 6 room32，N=100 首轮循环分配模块代价值收敛曲线

Fig. 6 Cost convergence curve of goal allocation module in the first round in room32 when N=100

（4）MAPF 模块对比实验

为验证所提 MAPF 模块在求解速度和路径长度的提升，本实验将其替换为平行分层组合 CBS 算法（parallel hierarchical composition CBS，PHC-CBS）[9]，动态自适应次优分配 CBS（dynamic adaptive sub-optimal bound assignment CBS，DASB-ECBS）[13]，显示估计 CBS（explicit estimation CBS，EECBS）[14]，它们分别采用分层策略，次优策略，绕道和优先级策略改进 CBS 算法，是当前最佳多机器人路径规划算法。

首先在 4 张地图中都进行 $N=10,20,\cdots,100$，共 10 个组别的实验，统计各组别中每个机器人路径长度平均值，以验证所提优化机制的有效性。由于目标点分配模块会随路径构建过程调整分配方案，在该实验中不便于比较 MAPF 算法的性能，因此去除目标点分配模块，并向各算法指定相同的任务，仅采用 3.1 节 Construct-Constraints，3.2 和 3.3 节的 PathPlanning 模块作为本节验证模块。

4 张地图对应的 10 组实验中，各机器人平均路径长度随 N 的变化曲线如图 7（a）~（d）所示。平均路径长度反映了 MAPF 算法路径构建的质量。在相同环境和目标

点分配下,较小的平均路径长度说明避碰机制高效合理。如图 7 所示,在同一地图环境中,平均路径长度随机器人数量增加呈现上升趋势;而不同环境中,平均路径长度随着从(a)~(d)环境复杂程度的增加而增长。可见,机器人数量和环境复杂程度的增长会带来大量机器人之间冲突,为 MAPF 算法带来极大地挑战性。

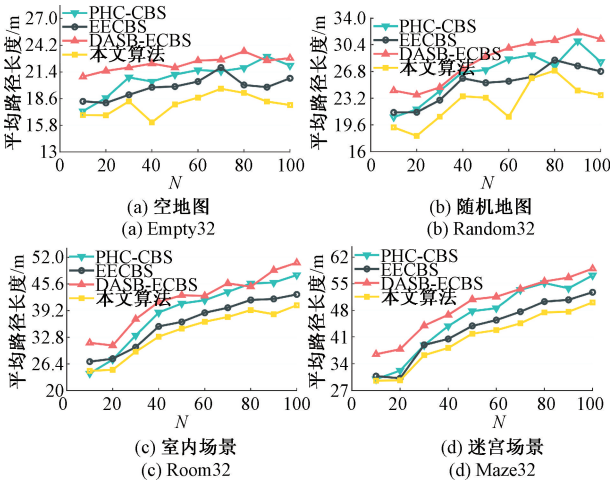


图 7　不同地图路径平均长度随 N 变化曲线

Fig. 7　Average path length w. r. t N in different map

图 7(a)~(d)所示的 4 张地图中,PHC-CBS 主要按照合并思想构建路径,虽然该策略在机器人数量较小时,提升了复杂环境中路径构建的质量,然而随着机器人数量的增长,不同子问题之间的冲突反而为算法带来很多棘手的关键冲突,而算法没有对关键、半关键冲突做出优化策略,导致路径质量的显著下降。

EECBS 算法主要采用绕道和优先级策略,有效地解决了关键、半关键冲突。然而算法并未确保优先级构建的合理性和最优性,当机器人数量继续增加后,不合理的求解顺序造成了死锁,从而触发了绕道。因此算法虽然保证了质量,却也产生了巨大的时间开销。

次优算法 DASB-ECBS 根据每个智能体遇到冲突数量分配各自权重,并据此求解次优上界。虽然尽快地找到可行解作为输出,却牺牲了最优性,因此所构建的路径质量相对较低。

本文所述模块在 4 张地图中均保持了最小的平均路径长度。通过路径回退式约束构建机制构建更可靠的避让策略,有效地解决了关键冲突;引入避碰路径估计的绕道机制预估了半关键冲突的等待代价,及时调整路线,减少避让等待造成的路径长度增加;在底层路径构建中共享机器人之间位置实现对冲突的预判,减少所构建路径中的冲突数量,减小 CBS 搜索树工作负担。在 room32 地图中,$N=100$ 时,所提算法模块平均路径长度(40.47 m)相比当前最优算法 EECBS(43.08 m)减少了 6.058%,提升了路径质量。此外在本实验中,由于没有目标点分配模块的介入,导致 $N=100$,room32 下所构建路径长度总和(4 047)相比 5.1.2 节代价总和收敛值(2 450)更大,也充分说明目标点分配模块能够降低多机器人系统路径总代价。

算法运行时间上,对 4 张地图都进行 $N=10,20,\cdots,100$ 共 10 个组别的实验,每组进行 50 次随机选取机器人初始位置和目标点位置的验证。4 张地图对应的 10 组实验中,每组的 50 次求解平均时间随组别智能体-目标点对数量 N 的变化曲线如图 8(a)到(d)所示。

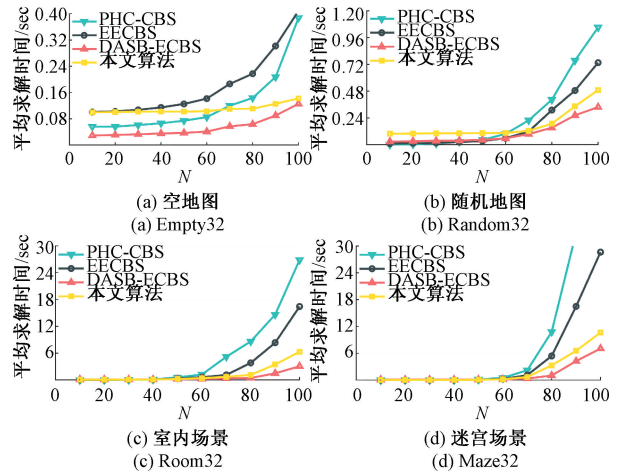


图 8　不同地图平均求解时间随 N 变化曲线

Fig. 8　Average runtime trajectory w. r. t N in different map

由图 8 知,PHC-CBS 仅在机器人数量小时,具有较高效率,随着机器人数量的增加,处理子问题之间的避碰带来了大量的时间开销,因此运算时间迅速增长。EECBS 无法保障所构建优先级的合理性,绕道策略也无法根据冲突灵活设计,因此在智能体数量继续增加后,不合理的求解顺序反而带来了更大的搜索负担。DASB-ECBS 通过牺牲最优性,确保了算法求解效率,是目前 MAPF 问题求解速度最快的算法。

本文算法主要通过底层共享机器人位置构建机制减少 CBS 算法的冲突数量,以提高搜索速度。虽然算法效率略低于次优思想 DASB-ECBS,room32 地图中 6 s 完成 100 辆无人车路径构建,但保障了解的最优性。综上,所提多智能体路径规划算法模块相比最快算法 DASB-ECBS 效率降低可接受的前提下,所构建路径质量超越了最优算法 EECBS,说明了所提方法突破了当前 MAPF 无法兼顾质量与效率的问题。

5.2　实际实验

本文采用 3 辆塔克机器人 AKMX2 型四轮小车完成实际实验测试。如图 9 所示,小车采用 28 cm * 18 cm 阿克曼的底盘,搭载思岚单线激光雷达和 Nvidia Jetson

Nano 车载计算机,采用 ROS Melodic 作为机器人操作系统。每辆小车通过 WiFi 接入局域网,基于 ROS 分布式通讯连接一台搭载 Intel Core-i5 处理器的控制主机。

图 9 实验用机器人

Fig. 9 The robot used for field test

采用 2.5 m * 3.0 m 的室内作为实验场地,如图 10 和图 11 所示。采用 Cartographer 算法完成环境地图构建,AMCL 算法实现激光重定位。各车的路径由笔记本主机构建,通过局域网发送到各小车上,最终被车辆的跟踪控制算法执行。

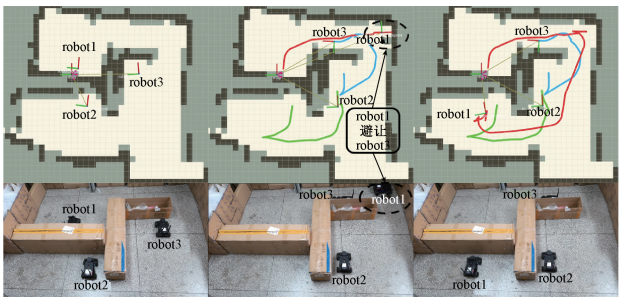


图 10 两阶段算法实际实验

Fig. 10 Field test with two-stage algorithm

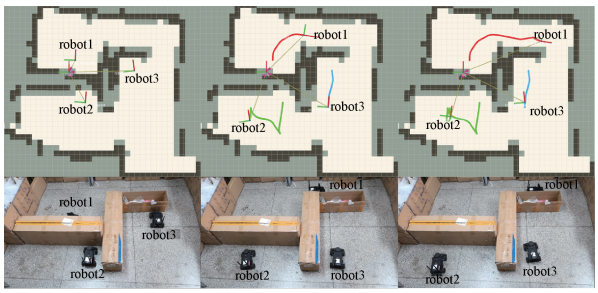


图 11 本文算法实际实验

Fig. 11 Field test with proposed algorithm

初始时刻,随机放置 3 辆小车并指定 3 个目标点,分别采用分配规划两阶段算法和 Nutcracker-CBS 完成目标点分配和各车路径。接下来,分别采用纯跟踪和 PID 作为控制方向和车速,确保各车准确跟踪规划的轨迹,减小控制误差对实验造成的影响。采用两阶段算法构建路径如图 10 所示。它首先根据各车初始位置与目标点之间欧氏距离计算目标点分配,然后采用 ECBS 构建路径。由于环境的复杂性,robot1 前往所分配目标点的过程中,一方面绕过障碍物造成路径长度增加,另一方面在地图左上角处避让 robot3 造成运行时间的进一步延长,对系统带来了显著的延时。在该实验中,3 个机器人路径长度分别为 5.7 m,3.1 m 和 2.6 m,系统运行总时长为 31 s。

本文所述算法 Nutcracker-CBS 构建方案如图 11 所示。由于在路径构建和结点分裂过程中综合评判各方案的合理性,逐渐筛选出图 11 的分配方案和各车路径。该方案一方面减少了机器人之间的冲突,另一方面缩短了系统的总体路径长度和运行时长。在该实验中,3 个机器人路径长度分别为 2.1 m,1.9 m 和 1.1 m,系统运行总时长为 12 s。路径总和(5.1 m)相比两阶段算法(11.4 m)缩减 55.26%,系统运行总时长缩减 61.29%,显著提升多机器人系统在复杂场景中非指定式多智能体路径规划的能力。

6 结 论

本文提出一种面向多无人车的目标分配和协同路径规划算法,求解了非指定式 MAPF 问题。它采用紧耦合的目标点分配路径规划框架,并对目标点分配模块和 MAPF 模块进行改进。采用逻辑运算改进 NOA 的状态更新策略,提升整数规划问题求解效率;针对关键冲突和半关键冲突,分别采取路径回退式约束构建机制和引入避碰路径估计的绕道机制,提升解决冲突的求解效率和质量;采用数据共享底层路径规划机制,减少智能体之间碰撞,分担了上层算法的多车路径搜索负担。

实验证明,本文算法提升了非指定 MAPF 的求解效率,减小了路径长度。在数据集实验中,构建路径长度与 SOTA 算法 TAPF 相当,但求解时耗缩减 90.37%;改进的 NOA 算法求解耗时相比标准 NOA 减少了 86.76%;MAPF 模块 6 s 内完成 100 辆无人车路径构建,效率上与次优算法 DASB-ECBS 持平,但平均路径长度相比最优算法 EECBS 缩短了 6.058%。实际实验中,路径总和与系统运行时长分别相对两阶段算法缩减了 55.26% 和 61.29%,从而提升了多机器人系统的效率,降低了冲突数量和路径长度总和。

参考文献

[1] 王宏健,鄂鑫,张凯,等. 改进蚁群算法解决 UUV 集群任务规划问题[J]. 仪器仪表学报, 2022, 43(9):

238-254.
WANG H J, E X, ZHANG K, et al. Improved ant colony algorithm to solve UUV cluster task planning problem[J]. Chinese Journal of Scientific Instrument, 2022, 43(9): 238-254.

[2] PARK J, KIM J, JANG I, et al. Efficient multi-agent trajectory planning with feasibility guarantee using relative bernstein polynomial[C]. IEEE International Conference on Robotics and Automation, 2020: 434-440.

[3] SOLIS V, MOTES J, SANDSTROM R, et al. Representation-optimal multi-robot motion planning using conflict-based search[J]. IEEE Robotics and Automation Letters, 2021, 6(3): 4608-4615.

[4] ALONSO-MORA J, BEARDSLEY P, SIEGWART R, et al. Cooperative collision avoidance for nonholonomic robots[J]. IEEE Transactions on Robotics, 2018, 34(2): 404-420.

[5] ZHOU D, WANG Z, BANDYOPADHYAY S, et al. Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells[J]. IEEE Robotics and Automation Letters, 2017, 2(2): 1047-1054.

[6] REN Z, RATHINAM S, CHOSET H. MS*: A new exact algorithm for multi-agent simultaneous multi-goal sequencing and path finding[C]. IEEE International Conference on Robotics and Automation, 2021: 11560-11565.

[7] REN Z, 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.

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

[9] LEE H, MOTES J, MORALES M, et al. Parallel hierarchical composition conflict-based search for optimal multi-agent pathfinding[J]. IEEE Robotics and Automation Letters, 2021, 6(4): 7001-7008.

[10] SEMIZ F, POLAT F. Incremental multi-agent path finding[J]. Future Generation Computer Systems, 2021, 116: 220-233.

[11] ANDREYCHUK A, YAKOVLEV K, SURYNEK P, et al. Multi-agent pathfinding with continuous time[J]. Artificial Intelligence, 2022, 305: 103662.

[12] BOYARSKI E, FELNER A, LE BODIC P, et al. F-aware conflict prioritization & improved heuristics for conflict-based search[C]. Proceedings of the AAAI Conference on Artificial Intelligence, 2021, 35(14): 12241-12248.

[13] RAHMAN M, ALAM Md A, ISLAM Md M, et al. An adaptive agent-specific sub-optimal bounding approach for multi-agent path finding[J]. IEEE Access, 2022, 10: 22226-22237.

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

[15] HOENIG W, KIESEL S, TINKA A, et al. Persistent and robust execution of mapf schedules in warehouses[J]. IEEE Robotics & Automation Letters, 2019, 4(2): 1125-1131.

[16] QAMAR RA, SARFRAZ M, RAHMAN A, et al. Multi-criterion multi-UAV task allocation under dynamic conditions[J]. Journal of King Saud University-Computer and Information Sciences, 2023, 35(9): 101734-101744.

[17] HENKEL C, ABBENSETH J, TOUSSAINT M. An optimal algorithm to solve the combined task allocation and path finding problem[C]. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2019: 4140-4146.

[18] ABDEL-BASSET M, MOHAMED R, JAMEEL M, et al. Spider wasp optimizer: A novel meta-heuristic optimization algorithm[J]. Artificial Intelligence Review, 2023, 56(10): 11675-11738.

[19] ABDEL-BASSET M, MOHAMED R, JAMEEL M, et al. Nutcracker optimizer: A novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems[J]. Knowledge-Based Systems, 2023, 262: 110248.

[20] 张宏宏, 甘旭升, 李双峰, 等. 复杂低空环境下考虑区域风险评估的无人机航路规划[J]. 仪器仪表学报, 2021, 42(1): 257-266.
ZHANG H H, GAN X SH, LI SH F, et al. UAV route planning considering regional risk assessment under

complex low altitude environment[J]. Chinese Journal of Scientific Instrument, 2021, 42(1): 257-266.

[21] 李艳生, 万勇, 张毅, 等. 基于人工蜂群-自适应遗传算法的仓储机器人路径规划[J]. 仪器仪表学报, 2022, 43(4): 282-290.
LI Y SH, WAN Y, ZHANG Y, et al. Path planning for warehouse robot based on the artificial bee colony-adaptive genetic algorithm [J]. Chinese Journal of Scientific Instrument, 2022, 43(4): 282-290.

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

作者简介

谷依田,2021 年于东南大学取得学士学位,现为东南大学硕士研究生,主要研究方向为移动机器人路径规划和多机器人协同控制。
E-mail: 220213652@seu.edu.cn

Gu Yitian received his B. Sc. degree from Southeast University in 2021. He is currently a master student at Southeast University. His main research interests include path planning of mobile robots and multi-robot cooperative control.

张涛(通信作者),2008 年毕业于东南大学获得博士学位,现为东南大学教授,主要研究方向为机器人导航制导与控制。
Email: zhangtao22@seu.edu.cn

Zhang Tao (Corresponding author) received his Ph. D. degree from Southeast University in 2008. He is currently a professor at Southeast University. His main research interests include navigation guidance and control of robots.