计算机科学
COMPUTER SCIENCE

基于冲突搜索的多智能体路径规划研究进展

王子晗, 童向荣

引用本文

王子晗, 童向荣. 基于冲突搜索的多智能体路径规划研究进展[J]. 计算机科学, 2023, 50(6): 358-368.

WANG Zihan, TONG Xiangrong. Research Progress of Multi-agent Path Finding Based on Conflict-based Search Algorithms [J]. Computer Science, 2023, 50(6): 358-368.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220800151

基于冲突搜索的多智能体路径规划研究进展

王子晗　童向荣
烟台大学计算机与控制工程学院　山东 烟台 264005
(zihan_won@163.com)

摘　要　多智能体路径规划是人工智能领域一个经典的搜索问题,基于冲突的搜索算法是当前解决该问题的最优算法之一。文中讨论了多智能体路径规划的基础研究,对国内外近年来基于冲突搜索算法及其变体的研究成果进行了分类,根据改进方式将其变体分为4类,包括分割策略的改进、启发式算法、对典型冲突的处理和次优算法。同时介绍了基于冲突的搜索算法在多智能体路径规划的扩展问题中的应用。最后根据当前算法的优缺点,指出了目前面临的挑战,并针对这些挑战给出了未来可能的研究方向。

关键词:人工智能;多智能体路径规划;基于冲突的搜索算法;启发式搜索算法;A* 算法
中图法分类号　TP18

Research Progress of Multi-agent Path Finding Based on Conflict-based Search Algorithms

WANG Zihan and TONG Xiangrong
School of Computer and Control Engineering, Yantai University, Yantai, Shandong 264005, China

Abstract　Multi-agent path finding is a classic search problem in the field of artificial intelligence. Conflict-based search algorithm is one of the best algorithms to solve this problem. This paper discusses the basic research of multi-agent path finding, and classifies the research results based on conflict search algorithms and their variants in recent years. According to the improved ways, the variants are divided into four categories, including segmentation strategy improvement, heuristic algorithm, bounded suboptimal algorithm and typical conflict processing. It also introduces the application of the conflict-based search algorithm to the extended problem of multi-agent path finding. Finally, according to the advantages and disadvantages of the current algorithm, the existing challenges are pointed out. In view of these challenges, the possible research directions in the future are given.

Keywords　Artificial intelligence, Multi-agent path finding, Conflict-based-search, Heuristic search algorithm, A* algorithm

1　引言

多 Agent 路径搜索问题(Multi-Agent Path Finding, MAPF)的任务是为多个 Agent 规划路径,其关键是使多个 Agent 能够同时沿着各自的路径行动而不发生冲突。MAPF 在电子游戏[1-2]、机器人[3-5]、自动交叉路口管理[6-7]、无人机[8-9]和信任路径搜索[10-13]等领域有很多应用,尤其是在物流行业[14-15]。随着电子商务的蓬勃发展,物流行业在最近十年内取得了长足的进步,越来越多的物流集散中心使用自动引导小车系统[16-17](Automated Guided Vehicle, AGV)分拣货物,这些系统本质上就是解决 MAPF 问题。

MAPF 求解方法可分为集中式方法和分布式方法。在分布式方法中,每个 Agent 都有独立的计算能力,Agent 之间互相通信,共享信息。在集中式方法中,只有一个中央处理器为所有 Agent 规划路径。其中,具有分布式计算能力,并且使用中央求解器控制所有 Agent 的方法也属于集中式方法。本文所综述的算法基本上都基于集中式方法。

集中式方法又可以分为解耦算法、耦合算法和混合算法。解耦算法将问题分解为单 Agent 问题,在较小的搜索空间内寻找可行解,但不能保证解的最优性和完备性。耦合方法使用一个联合空间来表示所有 Agent 的状态,目标是找到最优解,通常计算开销较大。混合算法结合了解耦和耦合优势,既考虑全局搜索空间,在搜索层面上又是面向单 Agent。

在早期的研究工作中,MAPF 的最优算法一般采用基于 A* 的耦合算法。A* 算法[18]是一种最优的单 Agent 路径搜索算法,然而用 A* 算法解决多 Agent 路径搜索问题时,状态空间会随着 Agent 的数量增加呈指数增长,当 Agent 数量较多、场景较大时,通常无法在有限的时间内找到最优解。也有

到稿日期:2022-08-15　返修日期:2022-12-02
基金项目:国家自然科学基金(62072392, 61972360);山东省重大科技创新工程项目(2019522Y020131);烟台市重点实验室:高端海洋工程装备智能技术
This work was supported by the National Natural Science Foundation of China(62072392, 61972360), Major Innovation Project of Science and Technology of Shandong Province(2019522Y020131) and Yantai Key Laboratory: Intelligent Technology of High-end Marine Engineering Equipment.
通信作者:童向荣(xr_tong@163.com)

研究者将 MAPF 问题规约成其他领域中成熟的问题，如布尔可满足性问题[19-21](Boolean Satisfiability，SAT)、整数线性规划问题[22-24](Integer Linear Programming，ILP)和约束满足问题(Constraint Satisfaction Problem，CSP)[25]等，但这些方法通常只能解决小型问题，如果问题规模过大，则会因规则设计的复杂性而导致效率低下，效果并不理想。

以往算法存在的问题促使各种新的 MAPF 求解算法不断涌现，例如 Sharon 等[26]提出了递增成本树搜索算法(Increasing Cost Tree Search，ICTS)。首先提出了一种“两层搜索”的概念，ICTS 的高层为所有的 Agent 分配一个成本值，低层在高层指定的成本下搜索路径。随后又进一步改进了这种“两层搜索”策略，提出了基于冲突的搜索算法(Conflict-Based-Search，CBS)[27]。

CBS 算法结合了解耦算法和耦合算法的优势，将多 Agent路径搜索分解为多个单 Agent路径搜索问题，而单 Agent 的路径规划能在较短的时间内完成，搜索空间比联合 A* 算法小得多；同时又考虑到了 Agent 之间的耦合关系，通过添加约束的方式解决路径之间的冲突，保证了解的完备性和最优性。此外，任何单 Agent 搜索算法都适用于 CBS 的低级搜索，有利于实现不同算法之间的融合，因此 CBS 算法成为近年来 MAPF 研究领域的热点。

目前国内外虽有一些总结 MAPF 问题的研究综述[28-30]，但对 CBS 算法的总结尚不多见，本文分析了近年来 CBS 算法的最新研究成果，并对其进行了分类叙述。

2 MAPF 问题的研究基础

MAPF 是人工智能领域的一个经典搜索问题，存在许多变体，不同的 MAPF 研究论文有不同的假设和目标函数。MAPF 问题的经典定义如下：

给定一个图 $G=(V,E)$，和一组 k 大小的 Agent，用 $A=\{a_1,a_2,\cdots,a_k\}$ 表示。每个 Agent a_i 都有一个起始顶点 s_i 和目标顶点 g_i。假设时间是离散的，在每个时间步内，Agent 可以移动到相邻顶点，也可以停留在同一顶点。MAPF 的目标是在满足预先给定的约束条件下，为 k 个 Agent 找到一组从 s_i 到 g_i 且互不冲突的路径。

当两个 Agent a_i 和 a_j 试图在相同的时间步 t 占用相同的顶点 v 或相同的边 e 时发生冲突。如果冲突发生在顶点，则冲突被表示为 (a_i,a_j,v,t)；如果冲突发生在边上，则冲突被表示为 (a_i,a_j,e,t)。设 l_i 表示 Agent a_i 的路径，MAPF 的可行解是 k 个 Agent k 条路径的集合 $L=\{l_1,l_2,\cdots,l_k\}$。

图 1 给出了 4 邻域网格中边冲突和顶点冲突的示意图，a_1 和 a_2 的路径如实线所示，箭头表示 Agent 移动的方向，图 1(a)中 a_1 和 a_2 试图在同一时间步沿相反的方向交换位置，图 1(b)中 a_1 和 a_2 试图在同一时间步经过相同的位置。

(a)边冲突 (b)顶点冲突

图 1 边冲突和顶点冲突示意图

Fig. 1 Diagram of edge conflict and vertex conflict

对于 Agent 到达目标点后的行为，有两种常见的假设：一种是停留在目标顶点；另一种是在目标顶点消失。大多数的研究都假设 Agent 停留在目标位置。

评价 MAPF 求解算法有两个重要指标，即完备性和最优性。完备性指当 MAPF 问题有解时算法一定可以返回一个解；最优性指 MAPF 问题有解时算法一定返回最优解。在某些情况下，适当牺牲最优性和完备性能较快得到次优解。

MAPF 问题有两种不同的优化目标：最大完工时间(Makespan)和个体成本总和(Sum of Individual Costs，SIC)。一般追求的优化目标为 SIC，即所有 Agent 的成本总和。

A* 算法是经典的单 Agent 路径搜索算法，其将启发式信息运用到了图形搜索中，如式(1)所示：

$$f(n)=g(n)+h(n) \tag{1}$$

其中，$g(n)$ 表示从起始节点到当前节点的耗散值，$h(n)$ 表示从当前节点到目标点的预估耗散值。A* 算法同样适用于 MAPF，但使用 A* 算法求解 MAPF 问题时，存在分支因子过大的问题。考虑一个具有分支因子 b 的图，每个 Agent 在下一个时间步都有 $b+1$ 个可能状态(移动或静止)，在包含 k 个 Agent 的耦合全局搜索中，基于 A* 的算法每次搜索过程的分支因子为 $(b+1)^k$，并随着 Agent 数量的增加呈指数级增长。基于 A* 的算法虽然一定能返回最优解，但运行时间、内存占用量往往是不可接受的，因此许多研究者对其进行了改进。

Standley[31]使用所有单 Agent 的成本总和 SIC 作为启发式函数，并对 A* 算法进行两个改进：算子分解(Operator Decomposition，OD)和独立检测(Independent Detection，ID)。OD 通过引入中间状态来减少 A* 生成的节点数；ID 对 Agent 进行分组，把一个 MAPF 问题分解为多个小的 MAPF 问题，然后不断合并互相冲突的 Agent 组。EPEA*[32]算法运用先验知识，只将一部分后继节点加入 OPEN 列表中，大大减少了分支因子，展现出了为多 Agent 规划路径的能力。Silver 提出了 HCA*[33]算法，按照预定义的顺序一次规划一个 Agent，之后的 Agent 不能遍历以前 Agent 经过的位置，这种算法求解效率很高，但并不能保证最优性和完备性，并且次优解的范围不可界定。Sharon 等首次提出了一种“两层搜索”算法 ICTS，通过改进文献[34]所提出的多值决策图(Multi-Value Decision Diagram，MDD)来搜索路径。

Sharon 等随后提出了 CBS 两级搜索算法，该算法是目前使用最广泛的 MAPF 求解算法之一。图 2 给出了当前主流的 MAPF 问题研究方法以及 CBS 算法在该领域的定位。

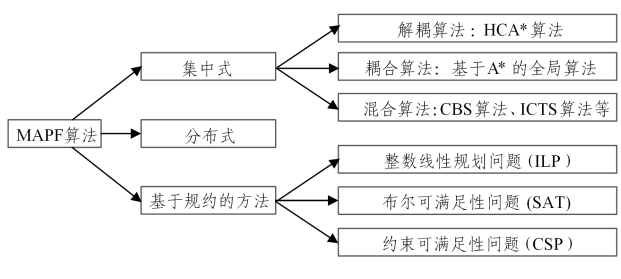


图 2 MAPF 问题的研究方法分类

Fig. 2 Classification of research methods for MAPF

3 基于冲突搜索及其变体

本节主要介绍 CBS 的基本概念和其变体的最新研究

成果，包括 CBS 分割策略的改进、启发式算法、几种典型冲突的处理和次优算法。

3.1 基于冲突的搜索算法

基于冲突的搜索是一种两级搜索算法，分为高级搜索和低级搜索。

高级搜索：CBS 的高级搜索构建一棵二叉约束树（Constraint Tree，CT），并通过添加约束的方式解决 Agent 之间的冲突。约束树的每一个节点 N 都存储了以下信息：

（1）一组约束（Constraints）。约束分为顶点约束和边约束，顶点约束用元组$\langle a_i, v, t\rangle$表示，表示a_i在 t 时间不能出现在位置 v；边约束用一个元组$\langle a_i, u, v, t\rangle$表示，表示$a_i$不能在时间步 $t-1$ 从位置 u 向位置 v 移动。

（2）一组方案（Solution）。方案包括 k 条路径，每个 Agent 一条，这些方案应满足当前节点的约束。

（3）成本（Cost）。成本为方案中的 k 条路径的成本总和。CBS 的高级搜索对二叉树中的叶子节点按照成本进行排序，每次扩展成本最小的叶子节点。

低级搜索：在节点 N 的约束下，忽略其他的 Agent，为每个 Agent 寻找路径，因此，低级搜索得到的 Agent 的路径之间可能存在冲突。

CBS 搜索过程可以描述为：首先建立根节点，根节点不存在任何约束。低级搜索为各个 Agent 寻找最优路径。如果两个 Agent 的路径发生冲突，则调用高级搜索，为保证最优性和完备性，生成两个子节点，分别对两个 Agent 施加约束以避免冲突。之后在满足新添加约束的条件下再次调用低级搜索，直到找到一个节点，节点中的路径不存在任何冲突。

3.2 CBS 分割策略的改进

最初的 CBS 算法在实践中有很大的随机性，首先体现在当节点存在多个冲突时，CBS 随机选择一对冲突进行分割，而解决冲突的顺序会影响算法的效率。许多研究者对 CBS 的随机分割策略进行了改进。

3.2.1 分割策略的改进

Boyaski 等[35]提出了增强的 CBS（Improved Conflict-Based Search，ICBS）算法，根据分割后子节点成本是否增加将冲突分为 3 类：基数冲突、半基数冲突和非基数冲突。当节点内有多个冲突时，ICBS 首先解决基数冲突。

ICBS 使用 MDD 识别冲突类型。每个 Agent 都维护一个 MDD，MDD_i是由a_i所有最短路径组成的有向无环图，图 3 给出了使用 MDD 识别基数冲突的一个实例。

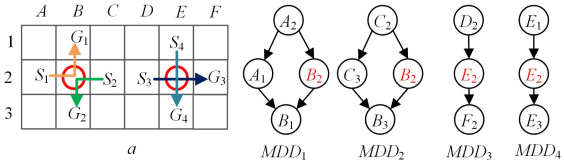


图 3 MAPF 实例和所有 Agent 的 MDD

Fig. 3 Example of MAPF and MDD of all Agents

图 3(a)给出了一个 MAPF 实例，有 4 个 Agent a_i，起始顶点和目标顶点分别位于S_i和G_i。在时间步 1，a_1和a_2在B_2发生冲突，a_3和a_4在E_2发生冲突。MDD_i是 Agent 各自的 MDD，a_1和a_2在时间步 1 都有不经过B_2的替代方案，因此a_1和a_2的冲突是非基数的；a_3和a_4在时间步 1 只有一种方案，因此a_3和a_4的冲突是基数冲突。

ICBS 还增加了绕过冲突的策略，如果a_i存在一条与 N 中成本相同的路径，且这条路径与其他的 Agent 不发生冲突，则用该路径替换 N 中a_i路径。

Li 等[36]提出了不相交分割的 CBS 算法（CBS with Disjoint Splitting，CBS-DS），添加约束时将问题分解为两个不相交的子问题，避免了搜索工作的重复，并引入正约束和负约束：

正约束：$\langle a_i, v, t\rangle$，强制a_i在时间步 t 占据节点 v。

负约束：$\overline{\langle a_j, v, t\rangle}$，禁止$a_j$在时间步 t 占据节点 v。

CBS-DS 算法选择冲突$\langle a_i, a_j\ v, t\rangle$中两个 Agent 中的一个$a_k$进行分裂，创建两个子节点，并分别添加约束，一个约束强制a_k在时间步 t 占据节点 v，另一个禁止a_k在时间步 t 占据节点 v，这样得到的两个节点是不相交的，即任何最优解只能属于一个 CT 节点。

3.2.2 分割前的合并策略

当一组 Agent 强耦合时，即该组中 Agent 之间的内部冲突较多时，在分割之前对这些 Agent 进行合并是一种有效的策略。Sharon 等[37]提出了 Meta-Agent CBS 算法，与基本的 CBS 相比，MA-CBS 在分割节点之前增加了将多个 Agent 合并为元 Agent 的操作，设置了一个阈值 B，如果发现两个 Agent 之间的冲突数超过 B，则执行合并操作，合并后的 Agent被视为一个单 Agent 参与 CBS 过程，并调用耦合算法解决组内 Agent 的冲突。

Lee 等[38]又提出了层次合成的 CBS 算法（Hierarchical Composition CBS，HC-CBS），该算法将完整的 MAPF 问题分解为较小的子问题，之后将这些子问题分层地组合成更大的子问题，直到合并为原始的 MAPF 问题，并提出了随机合并、冲突合并和基数冲突合并 3 种合并原则。随机合并将 Agent 随机分组合并；冲突合并依据每个子问题对之间的内部冲突数量；基数冲突合并仅考虑了基数冲突。与冲突合并和基数冲突合并相比，随机合并的开销最小，适用于结构和特征不明显的环境。在密集场景下，冲突合并有更好的表现。

Lee 等同时提出了 HC-CBS 的两种并行算法，即并行 HC-CBS 算法（Parallel Hierarchical Composition CBS，PHC-CBS）和动态并行 HC-CBS 算法（Dynamic Parallel Hierarchical Composition CBS，DPHC-CBS）。PHC-CBS 每一层的子问题并行执行，运行时间以最慢的子问题为界；DPHC-CBS 子问题在作业完成时动态合并，率先完成的子问题可以提前合并进入下一层，DPHC-CBS 的动态特性最大限度地减少了空闲等待时间。

3.3 CBS 的启发式算法

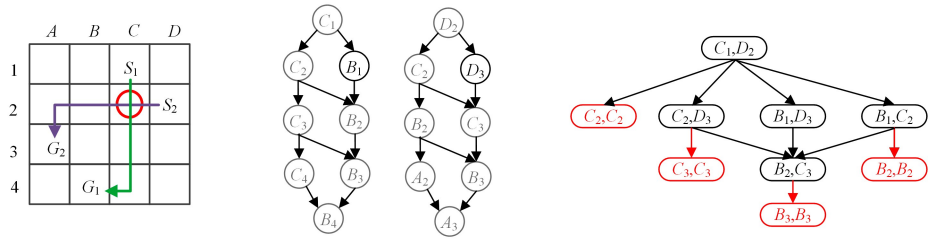
CBS 的搜索空间是一个二叉 CT 树，高级搜索只使用 CT 节点的成本进行优先级排序。该值可以被认为是节点的 g 值，许多研究者把一个非高估的 h 值加到它的优先级上，用来预估后续的节点分割数，从而得到具有启发式信息的 CBS 算法。

（1）基数冲突图（Conflict Graph，CG）启发式算法：Felner 等[39]构建了节点的基数冲突图G_{CF}，把 h 值的求解转换为求解图的顶点覆盖[40]问题，将基数冲突图的最小顶点覆盖的

大小(Minimum Vertex Cover，MVC)作为 h 值。

(2)依赖图(Dependency Graph，DG)启发式算法[41]：DG 在 CG 的基础上，将相互依赖的 Agent 添加到基数冲突图中，构建了成对依赖图G_D(Pairwise Dependency Graph)。DG 启发式的 h 值为成对冲突图 MVC 的大小。相互依赖表示 Agent 之间没有互不冲突的最短路径。

图 4 给出了 DG 如何识别依赖关系。图 4(a)是一个 MAPF 示例，某个节点 N 中的方案如图所示，在时间步 1，a_1 和a_2在C_2发生冲突。图 4(b)是a_1和a_2各自的 MDD，DG 启发式将合并两个 MDD，联合图图 4(c)的深度代表在相应的时间步两个 Agent 可达的位置，红色节点代表发生了冲突。联合 MDD 表明两个 Agent 没有互不冲突的最短路径。



(a) (b) (c)

图 4 一个 MAPF 示例、两个 Agent 的 MDD 和两个 MDD 的联合图(电子版为彩图)

Fig. 4 MAPF example, MDD of two Agents and Joint graph of two MDDs

(3)加权依赖图[41](Weighted Pairwise Dependency Graph，WDG)启发式算法：对于两个冲突的 Agent a_i 和a_j，将它们最小无冲突路径的成本之和与它们在 N 中的路径的成本之和的差称为Δ_{ij}($\Delta_{ij}\geqslant 0$)。DG 启发式算法只能提供$\Delta_{ij}\geqslant 0$ 的信息，并不能提供Δ_{ij}到底有多大的信息。WDG 启发式使用与 DG 启发式相同的图，但为每条边添加了权重，权重的值为Δ_{ij}。将 MVC 问题推广为边加权最小顶点覆盖(Edge-Weighted MVC，EWMVC)问题，为图中的每一个顶点赋值一个x_i，x_i是满足式(2)的最小值：

$$x_i+x_j\geqslant w_{ij} \tag{2}$$

其中，w_{ij}的为 v_i 和 v_j 之间边的权重，WDG 计算的启发式的值为：

$$h=\sum_i x_i \tag{3}$$

WDG 启发式算法和 DG 启发式算法相比，主要的区别在于 WDG 需要额外计算Δ_{ij}。

表 1 引用了文献[41]中 3 种启发式算法在不同场景下根节点内的 h 值(50 组实验的平均值)对比，h^* 是最优解成本减去根节点的成本，h 与 h^* 越接近，说明启发式越准确。密集图上 Agent 之间的耦合度较高，因此仅考虑依赖关系和基数冲突是不够的，WDG 算法提供了更准确的 h 值。启发式越复杂需要的额外开销就越大，但对于密集图来说，这些额外的开销是值得的，场景越复杂，高级启发式算法的表现就越好。

表 1 不同实例上根 CT 节点的平均 h 值和平均 h^* 值

Table 1 Average h-values and average h^*-values of root CT node on different instances

Type	Agents	CG	DG	WDG	h^*
Empty	30	0.2	1.0	1.2	1.7
Empty	40	0.5	1.6	2.0	3.3
Empty	50	0.5	2.2	2.6	4.7
Dense	16	3.9	3.9	11.5	18.6
Dense	20	4.7	4.7	14.0	23.2
Dense	24	6.5	6.5	18.9	28.5

上述的 3 种启发式算法只考虑了后续节点中 g 值的变化，但由于搜索过程中的 h 值也会减小，因此后续节点中的 f 值可能并不会随着 g 值的增大而增大。Boyarski 等[42]提出了一种新的冲突类型分类方式，在 ICBS 算法的基础上增加了两种冲突类型：f-基数冲突和半 f-基数冲突。按照新的冲突优先级顺序进一步减小 CT 树的大小。

表 2 列出了文献[42]中的冲突类型。其中的 g-cardinal 冲突和 semi-g-cardinal 冲突相当于 ICBS 算法中的基数冲突和半基数冲突。

表 2 文献[42]中的冲突类型(优先级从上往下递减)

Table 2 Types of conflicts in reference [42](priority decreases from top to bottom)

名称	定义	识别方式
f-cardinal	子节点的 f 值都增加	模拟子节点的 MVC，MVC 不变，且冲突位置发生在目标顶点
semi-f-cardinal	一个子节点的 f 值增加	模拟子节点的 MVC，MVC 大小不变且无目标冲突
g-cardinal	子节点的 g 值都增加	MDD 在冲突时间步没有替代方案
semi-g-cardinal	一个子节点的 g 值增加	只有一个 Agent 的 MDD 存在旁路
non-g-cardinal	子节点的成本不变	MDD 在冲突时间步有替代方案

(4)近顶点权重启发式算法：文献[42]在 3 种冲突图的每一条边的两侧添加了近顶点权重，分别增强了 CG，DG，WDG 启发式的信息，得到了近顶点加权基数冲突图(Near Vertex-Weighted Cardinal Conflict Graph，NVWCG)、近顶点加权依赖图(Near Vertex-Weighted DG，NVWDG)和近顶点加权的边加权依赖图(Near Vertex-Weighted Edge-Weighted Dependency Graph，NVWEWDG)启发式算法。这些近权重表示重新规划图中的对应顶点 Agent 的路径来解决两个 Agent 之间的当前冲突所需的最小附加成本。在构建近顶点权重的冲突图之后，设边 $e=(v_i,v_j)$的权重分别为c_{e_i}和c_{e_j}，为顶点v_i赋值x_i，x_i满足：

$$x_i\geqslant c_{e_i}\vee x_j\geqslant c_{e_j} \tag{4}$$

近顶点权重启发式的 h 值为满足顶点覆盖的x_i最小和，如果放宽时间限制，近顶点权重启发式算法将有更优秀的表现。因为如果限制的时间较短，计算的开销不会得到回报。

表 3 列出本节所提到的 CBS 启发式算法。

表 3 CBS 启发式算法的对比

Table 3 Comparison of CBS heuristic algorithms

名称	h 值	求 h 值所需的额外计算量	适用场景
CG	基数冲突图 MVC 大小	构建基数冲突图、求解图 MVC 大小	基数冲突较多的场景
DG	依赖图的 MVC 大小	识别冲突对存在的依赖关系，构建依赖图，求解图 MVC 的大小	小型密集地图
WDG	依赖图的 EVMVC	构建依赖图，解决一个双 Agent 问题，计算 Δ_{ij}，求解 EVMVC	大型密集地图
NVWCG	满足顶点覆盖的 x_i 最小和	在 CG 的基础上计算近顶点权重	限制时间较长的场景
NVWDG	满足顶点覆盖的 x_i 最小和	在 DG 的基础上计算近顶点权重	限制时间较长的场景
NVWEWDG	成本变化最小的顶点覆盖的 x_i 最小和	在 WDG 的基础上计算近顶点权重，确定重新规划哪些 Agent 成本的变化值最小	限制时间较长的场景

3.4 几种典型的对称冲突

4 连通网格图中存在一种对称性，即一组路径或路径段之间的等价性，这些路径或路径段具有相同的起始顶点和目标顶点以及相同的成本，仅动作出现的顺序不同。

在某些情况下，两个 Agent 所有的最短路径是互相冲突的，这表示在当前成本下无法为两个 Agent 找到有效方案。由于 CBS 的特性，如果两个 Agent 发生冲突，那么就执行最佳优先搜索，首先寻找相同成本下的替代方案，但这些方案是互相冲突的，CBS 算法必须列举这些冲突路径的组合，从而导致无效节点的扩展。

文献[43-45]识别了 3 种具有对称性特点的冲突：矩形冲突、目标冲突、走廊冲突。对于每种类型的冲突，文中改进了高级搜索中添加约束的策略。新的推理技术根据最优解的特征，在一个节点中同时添加多个约束或添加范围约束(在某个时间范围内禁止某 Agent 通过某节点)，能在单个分支步骤中解决冲突，减小了约束树的大小。图 5 给出了 3 种冲突的示例，其中 S_i 和 G_i 分别表示 Agent a_i 的起始顶点和目标顶点。文献[43-45]提到的几种推理方式具体如下。

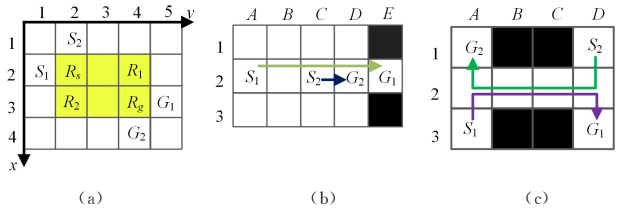


图 5 矩形冲突、目标冲突、走廊冲突的示例(电子版为彩图)

Fig. 5 Examples of rectangle conflict, target conflict and corridor conflict

(1)矩形冲突。如图 5(a)所示，a_1 和 a_2 的所有最短路径都会在黄色矩形区域内的某个位置产生冲突，Li 等将这类冲突称为矩形冲突。如果两个 Agent 存在矩形冲突，则满足以下等式：

$$|S_1 \cdot x - G_1 \cdot x| + |S_1 \cdot y - G_1 \cdot y| = G_1 \cdot t - S_1 \cdot t > 0 \tag{5}$$

$$|S_2 \cdot x - G_2 \cdot x| + |S_2 \cdot y - G_2 \cdot y| = G_2 \cdot t - S_2 \cdot t > 0 \tag{6}$$

$$(S_1 \cdot x - G_1 \cdot x)(S_2 \cdot x - G_2 \cdot x) \geqslant 0 \tag{7}$$

$$(S_1 \cdot y - G_1 \cdot y)(S_2 \cdot y - G_2 \cdot y) \geqslant 0 \tag{8}$$

其中，$S_i \cdot t$ 和 $G_i \cdot t$ 表示 Agent a_i 最短路径开始和结束的时间步。

文献[34]定义了出口边界，从 R_1 到 R_g 的边界和从 R_2 到 R_g 的边界分别称为 a_1 和 a_2 的出口边界，分别用 R_1R_g 和 R_2R_g 表示。Li 等发现，经过 Agent 相应出口边界的所有最短路径的组合都是冲突的，通过在矩形区域内给予一个 Agent 优先权，迫使另一个 Agent 等待一个时间步或绕道来解决矩形冲突。定义障碍约束(Barrier constraint)如下：

$$B(a_i, R_i, R_g) = \{\langle a_i, (x,y), t\rangle | ((x,y), t)\} \tag{9}$$

其中，$((x,y),t) \in R_iR_g$，i 的值为 1 或 2。它是一组顶点约束，表示 Agent 禁止出现在出口边界 R_iR_g 上的任何位置。在用 CBS 解决矩形冲突时，生成两个子节点，将 $B(a_1, R_1, R_g)$ 添加到其中一个节点，将 $B(a_2, R_2, R_g)$ 添加到另一个节点。

(2)目标冲突。如图 5(b)所示，当一个 Agent 已经到达目标点并停留在那里之后，另一个 Agent 经过第一个 Agent 的目标顶点会产生目标冲突。解决目标冲突的方法是对 Agent 的路径长度进行限制。假设 a_2 在时间步 t' 到达其目标顶点 g_2 并永远停留在那里，然后，a_1 在时间步 $t(t \geqslant t')$ 访问顶点 g_2。文献[40]使用以下两个长度约束，限制 a_2 的路径长度 L_2 来解决目标冲突。

$L_2 > t$，表示 a_2 仅在时间步 t 之后才能完成其路径。该约束仅影响 a_2 的路径。

$L_2 \leqslant t$，a_2 必须在时间步 t 或 t 之后到达顶点 g_2，并永远停留在那里，这要求任何其他 Agent 都不能在时间步 t 或之后访问顶点 g_2。该约束可能影响所有 Agent 的路径。

(3)走廊冲突。当两个 Agent 试图同时沿相反方向通过狭窄的通道时，就会出现走廊冲突。如图 5(c)所示，CBS 算法每次只添加一个约束，限制一个 Agent 等待一个时间步，随着走廊深度的增加，扩展 CT 节点的数量呈指数增长。

解决走廊冲突的最优策略分为两种情况，一种是两个 Agent到达目标顶点都没有不穿过该走廊的替代路径，此时最好的策略是在一个 Agent 通过该走廊时禁止另一个 Agent 通过。另一种情况是存在不穿过走廊的旁路，此时需要对比旁路与等待哪一种策略更优，因此，考虑具有端点 e_1 和 e_2 的长度为 k 的走廊，文献[44]给两个子节点分别添加范围约束：

$$\langle a_1, e_1, [0, \min(t_1'(e_1)-1, t_2(e_1)+k)]\rangle \tag{10}$$

$$\langle a_2, e_2, [0, \min(t_2'(e_2)-1, t_1(e_2)+k)]\rangle \tag{11}$$

其中，$t_1(e_i)$ 是 a_i 穿过走廊达到 e_i 的最早时间步，$t'_i(e_i)$ 是 a_i 存在旁路的情况下到达 e_i 的最早时间步，$\langle a_i, v, [t_{\min}, t_{\max}]\rangle$ 是一个范围约束(Range Constraint)，表示在 $t_{\min}$ 到 $t_{\max}$ 这个时间段内，a_i 不能经过顶点 v。

以上几种冲突的推理技术都是在单分支内解决冲突，保证了解的最优性和完备性，并可与其他改进相结合。文献[46]用走廊推理的技术增强了 NVWCG 和 NVWDG 启发式。对于 NVWCG，将所有走廊冲突视为基数冲突，并将其边添加到冲突图中；对于 NVWDG，将解决走廊冲突时收到的额外范围约束下的最短路径成本和当前成本的差值作为近顶点权重。

表 4 列出了 3 种推理方式的识别方式和添加的约束类型。

表 4 典型冲突推理总结

Table 4 Summary of special types of conflicts

冲突类型	识别方法	约束类型	CBS 的节点分支方法
矩形冲突	两个 Agent 都遵循它们的曼哈顿最优路径，冲突区域内每个单元到两个起始节点的距离相等	障碍约束	分别限制两个 Agent 通过各自的出口边界，两个子节点分别添加一个障碍约束
目标冲突	将冲突时间步与 Agent 的路径长度进行比较，若路径长度大于冲突时间步，则发生了目标冲突	长度约束	约束到目标顶点的 Agent 的路径长度
走廊冲突	检查冲突顶点（或冲突边）是否为 2°，检查相邻两个顶点中的每个顶点的度数，然后重复这个过程，直到找到一个度数不为 2 的顶点，即为冲突走廊的起始顶点	范围约束	在某段时间内限制其中一个 Agent 进入走廊，两个子节点分别添加一个范围约束

3.5 CBS 的有界次优算法

最优求解 MAPF 被证明是 NP 难的[47-48]，因此任何最优算法能求解的 Agent 数量是有限的，适当牺牲解的质量可以缩短算法的运行时间，CBS 的低级搜索使用了 A* 算法，A* 算法的一些有界次优改进也可以用于改进 CBS 算法。

WA* 算法[49]对启发式进行加权处理，通过引入固定的权值，将与最优路径相比差异较小的次优路径也作为候选路径，A_ε^*[50]也是有界次优算法，其增加了 FOCAL 列表，FOCAL⊆OPEN，用来存储一些与最优节点相差较小的节点。

增强的次优 CBS 算法（Enhanced CBS，ECBS）[51]改进了 A_ε^*，在高级搜索和低级搜索时使用相同的次优因子 w。低级 ECBS 为 a_i 找到一条有界次优路径，高级搜索使用冲突的数量对 FOCAL 进行排序。ECBS 算法存在一定的局限性，首先，权重 w 是人工设置的，如果 w 过大，则允许 Agent 经更长的路径绕过当前的冲突，导致 Agent 的扩散并造成额外的冲突；另外 ECBS 对所有的 Agent 都指定同一个权重 w，不考虑每个 Agent 的实际需求，可能会导致求解路径较长，增大了搜索空间。

文献[52]提出了弹性的 ECBS 算法（Flexible ECBS，FECBS），根据有界次优解与路径成本之和的差值，动态调整 Agent 的路径，放宽对某些 Agent 的限制，允许它们选择成本大于有界次优解的路径，但总成本（控制）在次优界的范围内。文献[53]提出了自适应特定于 Agent 的次优 CBS 算法（Adaptive Agent-Specific Sub-Optimal Bounding Approach，ASB-ECBS）和它的动态版本 DASB-ECBS。算法根据每个 Agent 的冲突数量分配不同的权重 w_{a_i}：

$$w_{a_i}=1+\left(n*\frac{W-1}{count}\right) \tag{12}$$

其中，$count$ 为冲突的最大数量，W 为预先指定的权重值，n 为 a_i 的冲突总计数。因此，每个 Agent 得到一个不同的次优界，该次优界与该 Agent 和其他 Agent 冲突的数量成正比。

ASB-ECBS 只将 Agent 特定的次优界分配给 Agent 一次，适用于 Agent 较少且空旷的环境，而 DASB-ECBS 动态地为 Agent 分配特定的次优界，并在整个搜索过程中随时调整，适用于 Agent 较多且道路狭窄的环境。

在 ECBS 的高级搜索中，仅仅根据冲突数量排序，搜索可能会陷入局部最优。文献[54]提出了显示估计的 CBS 算法（Explicit Estimation CBS，EECBS）。EECBS 包含 3 个启发式值，即冲突数量、MVC 大小和成本的预估增加量，并将 CBS 的绕过冲突策略、WDG 启发式算法、矩形推理加入到 EECBS 算法中。

4 MAPF 的扩展及应用

经典的 MAPF 模型把时间离散为时间步，假设图上的每条边长度都是相同的，并忽略 Agent 的大小和形状。但这些假设并不能满足实际应用的需要，当前已经有许多研究者针对一些现实世界的场景，对 MAPF 问题进行重新定义。

（1）I-MAPF：在货物配送问题中，由于交通堵塞、临时维修等原因使某些节点堵塞，导致规划好的路径不可用，文献[55]引入了增量式多 Agent 路径规划（Incremental Multi-Agent Path Finding，I-MAPF）问题，提出了两种改进的解决 I-MAPF 的算法：CBS-Replanner 和 CBS-D*-lite，图 6 给出了两种算法的基本流程。

二者的主要区别在于低级搜索使用了不同的算法。每次环境发生变化，CBS-Replanner 算法都会从头开始执行常规 CBS 算法，重新建立约束树，低级搜索使用 A* 算法，总能返回最优解（如果存在），但计算量过于庞大，无法满足实时性的要求。CBS-D*-lite 低级搜索使用 D*-lite 算法[56]，该算法只有在当前约束下无法找到解的情况下才重新建立约束树，重新规划速度比 CBS-Replanner 快得多，适用于动态环境，但并不能保证解的最优性，且次优解的范围无法界定。

（2）在物流分拣中心这样的动态环境中，Agent 到达目标点会被赋予新的任务，环境中也可能随时加入新的 Agent，这被称为 Lifelong MAPF 问题[57-58]。在这些情况下，最好的策略是着眼于当前的冲突。文献[57]提出 LPCBS 算法，根据时间的顺序和新加入的任务，不断调整约束树，剪枝过时约束（当前时间步之前的约束），为新加入的 Agent 添加新的约束。

（3）连续时间下的 MAPF：经典的 MAPF 问题可被视为在同质图上对 Agent 进行路径规划，这种规划处理起来比较容易，但并不符合实际场景的要求，异质图上的 MAPF 也受到了研究者的广泛关注。

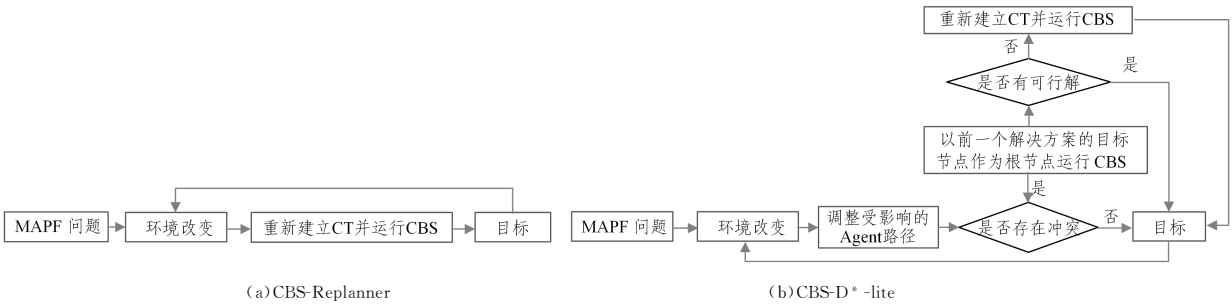


图 6 CBS-Replanner 和 CBS-D* -lite 的算法流程图

Fig. 6 Flow chart of CBS-Replanner and CBS-D* -litea algorithm

文献[59]定义了$MAPF_R$问题，在图的每条边上添加了权重，权重表示 Agent 通过这条边所需的时间。文献[60]提出了连续时间下的 CBS 算法(Continuous-Time CBS，CCBS)，将 CBS 的约束改为 Agent a_i 不能在某个时间段内执行某个动作，CCBS 将约束视为动态障碍，通过施加在 v 处的约束计算每个顶点 v 的安全间隔，低级搜索使用 SIPP 算法[61]在有效时间间隔内搜索路径。文献[62]将 CBS 算法中的冲突优先排序 、不相交分割和启发式算法结合到了 CCBS 算法中。文献[63]提出一种$MAPF_R$的有界次优算法 ECBS-CT，将 ECBS 的低级搜索更换为一种改进的 SIPP 算法，ECBS-CT 输入一个次优界 $w\geqslant 1$，生成的解的成本不超过最优成本的 w 倍。文献[64]考虑到在无人机在作业中常常执行往返任务，补充了$MAPF_R$问题，假设 Agent 到达目标点后立即返回起始顶点，并提出了 ICBS-HB 算法。

文献[65]提出了时间不确定的多 Agent 规划(MAPF with Time Uncertainty，MAPF-TU)，MAPF-TU 中每个动作的持续时间是不确定的。MAPF-TU 实例由元组$\langle G,s,t,w^-,w^+\rangle$定义，$G$，$s$ 和 t 与经典 MAPF 一致，w^- 和 w^+ 分别表示边持续时间的下限和上限。当 Agent 遵循某条路径 l_i 时，Agent 可能在 v 处的时间间隔称为潜在存在(Potential Presence)，表示为 $\tau(l_i,v)$，图 7 给出了潜在存在的示例。

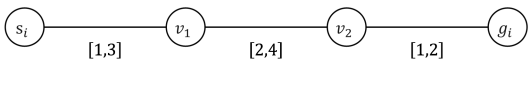


图 7 从s_i到g_i的某条路径中的潜在存在示意图

Fig. 7 Schematic diagram of potential presence in a path from s_i to g_i

设$l_i=((v_0,v_1)\cdots(v_{n-1},v_n))$为某 Agent 的路径，由 l_i 引起的顶点潜在存在如式(13)所示：

$$\tau(l,v)=\bigcup_{\substack{0\leqslant i\leqslant n\\ v_i=v}}\left[\sum_{0\leqslant j\leqslant i}w^-(v_j,v_{j+1}),\sum_{0\leqslant j\leqslant i}w^+(v_j,v_{j+1})\right] \tag{13}$$

边潜在存在与顶点潜在存在类似。

$$\tau(l,u,v)=\bigcup_{\substack{0\leqslant i\leqslant n\\ v_i=u\\ v_{i+1}=v}}\left[\sum_{0\leqslant j\leqslant i}w^-(v_j,v_{j+1}),\sum_{0\leqslant j\leqslant i}w^+(v_j,v_{j+1})\right] \tag{14}$$

如果两个 Agent 的路径在 v 的潜在存在相交，则它们在 v 处存在潜在顶点冲突。类似地，如果两个 Agent 的路径在同一条边上的潜在存在相交，则它们在一对顶点(u,v)处具有潜在边冲突。MAPF-TU 的目标是为 Agent 规划不存在潜在冲突的路径。针对 MAPF-TU 所提出的 CBS-TU 算法，对边添加范围约束，间接解决顶点潜在冲突。

此外，文献[66]将环境扩展到连续空间，其可行解为某时间区间内从起始位置到目标位置的一条路线，并将 CBS 应用于基于采样的运动规划，提出了 CBS-MP 算法，将路线离散成段，用以检测冲突。

(4)MAPF-DL：Hang 等[67]在 MAPF 问题中补充了截止时间的概念，给出一个截止时间 T，MAPF-DL 的优化目的是尽可能让更多的 Agent 在 T 之前到达目标，因此提出了 CBS-DL 算法。CBS-DL 算法根据 Agent 能否在 T 之前到达目标顶点将所有 Agent 分为成功 Agent 和未成功 Agent。Hang 等还将 CBS-DL 算法扩展到 DBS 算法，将 CBS 和 CBS-DL 相结合，DBS 以未成功的 Agent 为代价，即如果未成功的 Agent 增加 1，则成本增加 1。

(5)kR-MAPF：在很多实时环境中，可能出现意外事件延误一些 Agent，导致原计划不可行。Atzmon 等[68]定义了 k-鲁棒 MAPF(k-Robust MAPF，kR-MAPF)，并提出了 KR-CBS 算法[69]，确保 Agent 即使发生 k 个时间步以内的延迟也能遵循规划好的路径。KR-CBS 算法与 CBS 的不同之处在于它们识别和解决冲突的方式。当存在 $\Delta\in\{(0\cdots k)\}$时，出现 k 延迟冲突，其中a_i和a_j在 t 和 $t+\Delta$ 占据相同的顶点。为了解决冲突，KR-CBS 会在a_i或a_j上增加一个约束，a_i的约束为$\langle a_i,v,t\rangle$，a_j的约束为$\langle a_j,v,t+\Delta\rangle$。Chen 等[70]用 CBS 矩形推理、目标推理和走廊推理解决 kR-MAPF 中的成对对称现象，进一步提高了 KR-CBS 算法的成功率。

(6)考虑运动学的多 Agent 路径规划，经典的 MAPF 求解器往往忽略了 Agent 的大小，这将导致生成的解不能在真实世界的多 Agent 系统上执行，许多工作将运动学的约束加入 CBS 算法中。

文献[71]将类车机器人 MAPF 问题形式化为 CL-MAPF 问题，使用阿克曼转向对类车机器人进行状态表示，将 Agent 之间的冲突定义为实体冲突，并提出了 CL-CBS 算法，高级搜索构建实体冲突树，低级搜索使用时空混合状态 A* 为单 Agent 规划路径。添加约束时，允许 Agent 出现 k 秒以内的延迟，以避免因延迟而发生进一步的冲突。CL-CBS 还用顺序规划的方法缩短高级搜索的时间，将 Agent 分批，按次序为每一批 Agent 规划路径，并将规划好的路径视为动态障碍，作为约束加入到下一批次的根节点中。

文献[72]提出了一种基于双层走廊的模型预测优先轮廓

线控制方法，分为两个阶段。阶段一使用改进的 CBS 算法生成参考路径，在 8 邻域网格的环境下建模，并考虑了 Agent 之间的运动学约束；阶段二构建双层走廊以保证安全，静态走廊避免 Agent 与环境之间的碰撞，动态走廊避免 Agent 之间的碰撞。该模型也采用了与 CL-CBS 类似的分批策略，根据每个 Agent 的任务完成百分比来分配优先级。文献[73]根据工业仓库和自动化物流系统的实际情况，结合集中式方法和分散式方法的优势，提出了一种两层协作的终身系统，其中集中层负责交通流量预测与动态扇区路径规划，分散层则使用 CBS 算法和分层协作的 A* 算法进行局部路径规划。

5 现存挑战和未来的展望

本节总结了 CBS 算法目前的研究方向和挑战，并针对现有算法存在的问题对未来的研究进行了展望。

5.1 现有的主要研究方向

CBS 算法是目前解决 MAPF 问题最有效的方法之一，自提出以来就受到了学术界的广泛关注。根据前文的分析，当前对 CBS 算法的研究主要分为两个方向。

(1)完善基础的 CBS 算法，包括分割策略的改进、启发式算法研究和解决典型的对称冲突。CBS 算法的本质是构建一棵约束树，检测到冲突后，通过生成含新约束的子节点来解决冲突，节点的数量是衡量算法效率的重要指标，上述 3 种改进方式都是通过减少生成节点的数量来加速算法。分割策略的改进包括对冲突进行优先级排序、剪枝策略和合并策略。对冲突进行优先级排序旨在区分出影响成本的基数冲突，对于非基数冲突，采用绕过冲突的策略，能够在不分割节点的情况下解决冲突。CBS 的剪枝策略使解集分布到两个节点中，防止相同的冲突再次出现，生成了两个解集不相交的节点来避免搜索工作的重复，合并策略旨在在问题变得大而复杂之前解决高耦合的 Agent 组。启发式算法则通过启发式函数指导高级搜索的搜索方向，替代了随机选择的策略。针对典型的对称冲突，采取的解决思路是根据最优解的特征添加特殊类型的约束，在单个分支内解决冲突，杜绝了冗余节点的生成。

(2)用 CBS 算法解决实际问题。当前，对经典 MAPF 问题的变形研究越来越受到关注。在现实问题中，路径规划需要考虑速度、故障、Agent 的形状等因素，而经典的 MAPF 模型并没有考虑这些因素。许多研究者针对 MAPF 的变体问题，修改 CBS 算法的约束类型，提出了 CBS 变体算法。CBS 算法在 MAPF 的扩展问题上表现出了良好的适用性。

5.2 未来的研究方向

虽然 CBS 算法提出的时间并不长，但其在 MAPF 领域已经取得了不俗的表现，目前还处于发展阶段。结合以上的讨论分析，未来可能的研究方向主要分为以下几点。

(1)探索更多的冲突类型。当前 CBS 算法在解决冲突时往往考虑的是两个 Agent 之间的一对冲突，当两个 Agent 的冲突被解决后，新规划的路径也可能会与另外的 Agent 发生冲突。在某些 Agent 较多的密集场景里，可能会出现两个以上 Agent 之间的复杂交互，在这些场景下，只考虑两个 Agent 之间的冲突显然是不够的，未来可以考虑将多个冲突合在一起同时解决。此外，2.4 节中提到的几种对称型冲突的推理，仅适用于 4 连通网格下的标准 MAPF 问题，未来可以在 MAPF 的扩展问题上开发新的对称性冲突推理方法，比如：在 kR-MAPF 中，Agent 之间需要保持彼此之间的安全时间。因此，在不同时间步长处于同一顶点的 Agent 可能会出现潜在的冲突。假设 Agent 之间的速度并不相同，速度快的 Agent 可能会追上速度慢的 Agent，产生“追逐冲突”，以上情况都引入了更多的对称冲突。

(2)完善 CBS 的启发式算法。CBS 的启发式算法是最强大的 MAPF 算法之一。2.3 节提到的启发式算法都是将启发式信息转化为求解图的 MVC 问题，然而求解 MVC 也是 NP 难的，这在一定程度上限制了启发式算法的表现，启发式越复杂，就需要更多的额外开销。此外，有界次优算法已被广泛应用到 CBS 实践中，放宽最优解的条件可以提高 CBS 算法的上限。下一步可将现有的启发式算法运用到 CBS 的次优解法中，进一步提高 CBS 的次优算法求解问题的能力。

(3)研究分布式的 MAPF 算法。以往的工作通常采用集中式的方法，每个 Agent 执行由中央求解器规划的路径，Agent 之间具有相同的优先级。集中式的方法虽提供了建模上的便利，但最近一些研究表明，集中控制的模型在某些领域中并不可取。例如，文献[74]指出，在车辆路径问题中每台车辆都是独立自主的，如果将集中式方法用在城市车辆自动驾驶问题中显然是不切实际的；文献[75]表明，在无人机领域中每个参与者都是独立的实体，具有自己的业务目标，如果为避免冲突而采取更长的路径可能会引起客户的不满。为解决这类问题，可以在 CBS 算法中考虑公平性原则，对 Agent 和任务添加优先级；或采用分布式的方法。目前，由于分布式设计的复杂性，MAPF 分布式求解方法[76-77]并不常见。与集中式方法不同，分布式的解决方案由多个 Agent 规划，这些 Agent 之间相互协调，特别是在发生干扰的情况下有较好的鲁棒性，因此，开发分布式的 MAPF 算法也是未来研究的热点。

(4)此外，算法的表现与环境的大小、障碍密集程度、结构等因素息息相关，不同的环境下算法的表现有所不同。例如对于启发式算法，设计复杂的启发式函数通常会提供更准确的信息，但会增加额外的计算量，如果环境过于简单，其表现反而不如较为简单的启发式函数。因此，对自适应算法的研究也是必要的。

(5)文献[78]用机器学习的策略替换了 CBS 算法原有的冲突排序策略，除此之外，还没有 CBS 算法与机器学习相结合的工作。文献[79]表明 ICTS 和 ICBS 等基于搜索的求解器的概念同样也适用于基于 SAT 的方法，多种算法的融合也受到了越来越多学者的关注。

参 考 文 献

[1] WANG K H C, BOTEA A. Fast and memory-efficient multi-agent pathfinding[C]// Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling. 2008: 380-387.

[2] MA H, YANG J, COHEN L, et al. Feasibility study: Moving non-homogeneous teams in congested video game environments [C]// Proceedings of the AAAI Conference on Artificial Intelli-

gence and Interactive Digital Entertainment. 2017.
[3] VEDDER K, BISWAS J. X*: Anytime multi-agent path finding for sparse domains using window-based iterative repairs[J]. Artificial Intelligence, 2021, 291: 103417.
[4] HAN S D, YU J. Ddm: Fast near-optimal multi-robot path planning using diversified-path and optimal sub-problem solution database heuristics[J]. IEEE Robotics and Automation Letters, 2020, 5(2): 1350-1357.
[5] BARTÁK R, ŠVANCARA J, ŠKOPKOVÁ V, et al. Multi-agent path finding on real robots[J]. AI Communications, 2019, 32(3): 175-189.
[6] DRESNER K, STONE P. A multiagent approach to autonomous intersection management[J]. Journal of Artificial Intelligence Research, 2008, 31: 591-656.
[7] BUZACHIS A, CELESTI A, GALLETTA A, et al. A multi-agent autonomous intersection management (MA-AIM) system for smart cities leveraging edge-of-things and Blockchain[J]. Information Sciences, 2020, 522: 148-163.
[8] WALKER T T, CHAN D M, STURTEVANT N R. Using hierarchical constraints to avoid conflicts in multi-agent pathfinding [C]// Twenty-Seventh International Conference on Automated Planning and Scheduling. 2017.
[9] HÖNIG W, PREISS J A, KUMAR T K S, et al. Trajectory planning for quadrotor swarms[J]. IEEE Transactions on Robotics, 2018, 34(4): 856-869.
[10] CHE J Y, TONG X R. Dynamic Weighted Anytime Bounded Cost Search Algorithm[C]// 2021 7th International Conference on Systems and Informatics (ICSAI). IEEE, 2021: 1-6.
[11] KONG R, TONG X R. Anytime dynamic heuristic search for suboptimal solution on path search[C]// 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI). IEEE, 2020: 1070-1074.
[12] KONG R, TONG X R. Dynamic weighted heuristic trust path search algorithm[J]. IEEE Access, 2020, 8: 157382-157390.
[13] WEI T, TONG X R. Robust trust path generation based on weighted heuristic search[J]. Journal of Nanjing University, 2018, 54(6): 1161-1170.
[14] LIU Y, CHEN M, HUANG H. Multi-agent pathfinding based on improved cooperative A* in Kiva system[C]// 2019 5th International conference on control, automation and robotics (ICCAR). IEEE, 2019: 633-638.
[15] LI J, TINKA A, KIESEL S, et al. Lifelong multi-agent path finding in large-scale warehouses[C]// Association for the Advancement of Artificial Intelligence. 2020: 1898-1900.
[16] SUN S, GU C, WAN Q, et al. CROTPN based collision-free and deadlock-free path planning of AGVs in logistic center[C]// 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV). IEEE, 2018: 1685-1691.
[17] ZHANG Z, WU L, ZHANG W, et al. Energy-efficient path planning for a single-load automated guided vehicle in a manufacturing workshop[J]. Computers & Industrial Engineering, 2021, 158: 107397.
[18] HART P E, NILSSON N J, RAPHAEL B. A formal basis for the heuristic determination of minimum cost paths[J]. IEEE transactions on Systems Science and Cybernetics, 1968, 4(2): 100-107.
[19] SURYNEK P. Towards optimal cooperative path planning in hard setups through satisfiability solving[C]// Pacific Rim International Conference on Artificial Intelligence. Berlin/Heidelberg: Springer, 2012: 564-576.
[20] SURYNEK P, FELNER A, STERN R, et al. Efficient SAT approach to multi-agent path finding under the sum of costs objective[C]// Proceedings of the Twenty-second European Conference on Artificial Intelligence. 2016: 810-818.
[21] SURYNEK P, STERN R, BOYARSKI E, et al. Migrating Techniques from Search-Based Multi-Agent Path Finding Solvers to SAT-Based Approach[J]. Journal of Artificial Intelligence Research, 2022, 73: 553-618.
[22] YU J, LAVALLE S M. Planning optimal paths for multiple robots on graphs[C]// 2013 IEEE International Conference on Robotics and Automation. IEEE, 2013: 3612-3617.
[23] YU 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.
[24] WANG H, CHEN W. Multi-Robot Path Planning With Due Times[J]. IEEE Robotics and Automation Letters, 2022, 7(2): 4829-4836.
[25] RYAN M. Constraint-based multi-robot path planning [C]// 2010 IEEE International Conference on Robotics and Automation. IEEE, 2010: 922-928.
[26] SHARON G, STERN R, GOLDENBERG M, et al. The increasing cost tree search for optimal multi-agent pathfinding[J]. Artificial Intelligence, 2013, 195: 470-495.
[27] SHARON G, STERN R, FELNER A, et al. Conflict-based search for optimal multi-agent pathfinding[J]. Artificial Intelligence, 2015, 219: 40-66.
[28] STERN R, STURTEVANT N R, FELNER A, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks[C]// Twelfth Annual Symposium on Combinatorial Search. 2019.
[29] LIU Q Z, WU F. Research progress of multi-agent path planning [J]. Computer Engineering, 2020, 46(4): 1-10.
[30] CHE J, TONG X, KONG R. Intelligent trust path search[C]// 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI). IEEE, 2020: 1075-1080.
[31] STANDLEY T. Finding optimal solutions to cooperative pathfinding problems[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2010.
[32] FELNER A, GOLDENBERG M, SHARON G, et al. Partial-expansion A* with selective node generation[C]// Twenty-Sixth AAAI Conference on Artificial Intelligence. 2012.
[33] SILVER D. Cooperative Pathfinding[J]. Aiide, 2005, 1: 117-122.
[34] SRINIVASAN A, HAM T, MALIK S, et al. Algorithms for discrete function manipulation[C]// 1990 IEEE International Conference on Computer-Aided Design. IEEE Computer Society, 1990: 92-95.
[35] BOYARSKI E, FELNER A, STERN R, et al. ICBS: Improved

conflict-based search algorithm for multi-agent pathfinding [C]// Twenty-Fourth International Joint Conference on Artificial Intelligence. 2015.

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

[37] SHARON G, STERN R, FELNER A, et al. Meta-agent conflict-based search for optimal multi-agent path finding[C]// International Symposiumon Combinatorial Search. 2012.

[38] 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.

[39] FELNER A, LI J, BOYARSKI E, ET AL. Adding heuristics to conflict-based search for multi-agent path finding[C]// Proceedings of the International Conference on Automated Planning and Scheduling. 2018.

[40] FELNER A, KORF R E, HANAN S. Additive pattern database heuristics[J]. Journal of Artificial Intelligence Research, 2004, 22: 279-318.

[41] LI J, FELNER A, BOYARSKI E, et al. Improved heuristics for multi-agent path finding with conflict-based search[C]// Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence. 2019: 442-449.

[42] 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: 12241-12248.

[43] LI J, HARABOR D, STUCKEY P J, et al. Symmetry-breaking constraints for grid-based multi-agent path finding[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2019: 6087-6095.

[44] LI J, GANGE G, HARABOR D, et al. New techniques for pairwise symmetry breaking in multi-agent path finding[C]// Proceedings of the International Conference on Automated Planning and Scheduling. 2020, 30: 193-201.

[45] LI J, HARABOR D, STUCKEY P J, et al. Pairwise symmetry reasoning for multi-agent path finding search[J]. Artificial Intelligence, 2021, 301: 103574.

[46] BOYARSKI E, FELNER A, LE BODIC P, et al. Further improved heuristics for conflict-based search[C]// Proceedings of the International Symposium on Combinatorial Search. 2021: 213-215.

[47] YU J, LAVALLE S M. Structure and intractability of optimal multi-robot path planning on graphs [C]// Twenty-Seventh AAAI Conference on Artificial Intelligence. 2013.

[48] MA H, TOVEY C, SHARON G, et al. Multi-agent path finding with payload transfers and the package-exchange robot-routing problem[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2016.

[49] POHL I. Heuristic search viewed as path finding in a graph[J]. Artificial Intelligence, 1970, 1(3/4): 193-204.

[50] PEARL J, KIM J H. Studies in semi-admissible heuristics[J]. IEEE Transactionson Pattern Analysis and Machine Intelligence, 1982(4): 392-399.

[51] BARER M, SHARON G, STERN R, et al. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem[C]// Seventh Annual Symposium on Combinatorial Search. 2014.

[52] CHAN S H, LI J, GANGE G, et al. ECBS with flex distribution for bounded-suboptimal multi-agent path finding[C]// Proceedings of the International Symposium on Combinatorial Search. 2021: 159-161.

[53] RAHMAN M, ALAM M A, ISLAM M M, et al. An Adaptive Agent-Specific Sub-Optimal Bounding Approach for Multi-Agent Path Finding[J]. IEEE Access, 2022, 10: 22226-22237.

[54] 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(AAAI). 2021.

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

[56] KOENIG S, LIKHACHEV M. D* lite[C]// Eighteenth National Conference on Artificial Intelligence, American Association for Artificial Intelligence. Menlo Park, USA: CA, 2002: 476-483.

[57] MA H, LI J, KUMAR T K S, et al. Lifelong multi-agent path finding for online pickup and delivery tasks[C]// Proceedings of the International Conference on Autonomous Agents and Multiagent Systems(AAMAS). 2017: 837-845.

[58] WAN Q, GU C, SUN S, et al. Lifelong multi-agent path finding in a dynamic environment[C]// 2018 15th International Conference on Control, Automation, Robotics and Vision(ICARCV). IEEE, 2018: 875-882.

[59] WALKER T T, STURTEVANT N R, FELNER A. Extended increasing cost tree search for non-unit cost fomains[C]// Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence. 2018: 534-540.

[60] ANDREYCHUK A, YAKOVLEV K, ATZMON D, et al. Multi-agent pathfinding with continuous time[C]// IJCAI International Joint Conference on Artificial Intelligence. 2019: 39-45.

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

[62] ANDREYCHUK A, YAKOVLEV K, BOYARSKI E, et al. Improving continuous-time conflict based search[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2021: 11220-11227.

[63] COHEN L, URAS T, KUMAR T K S, et al. Optimal and bounded-suboptimal multi-agent motion planning[C]// Twelfth Annual Symposium on Combinatorial Search. 2019.

[64] AI B, JIANG J, YU S, et al. Multi-agent path finding with heterogeneous edges and roundtrips[J]. Knowledge-Based Systems, 2021, 234: 107554.

[65] SHAHAR T, SHEKHAR S, ATZMON D, et al. Safe Multi-Agent Pathfinding with Time Uncertainty[J]. Journal of Artificial Intelligence Research, 2021, 70: 923-954.

[66] SOLIS I, MOTES J, SANDSTRÖM R, et al. Representation-op-

timal multi-robot motion planning using conflict-based search [J]. IEEE Robotics and Automation Letters, 2021, 6(3): 4608-4615.

[67] MA H, WAGNER G, FELNER A, et al. Multi-agent path finding with deadlines[C]// Proceedings of the 27th International Joint Conference on Artificial Intelligence. 2018: 417-423.

[68] ATZMON D, STERN R, FELNER A, et al. Robust multi-agent path finding[C]// Eleventh Annual Symposium on Combinatorial Search. 2018.

[69] ATZMON D, STERN R, FELNER A, et al. Robust multi-agent path finding and executing[J]. Journal of Artificial Intelligence Research, 2020, 67: 549-579.

[70] CHEN Z, HARABOR D, LI J, et al. Symmetry breaking for k-robust multi-agent path finding[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2021: 12267-12274.

[71] WEN L, LIU Y, LI H. CL-MAPF: Multi-Agent Path Finding for Car-Like robots with kinematic and spatiotemporal constraints [J]. Robotics and Autonomous Systems, 2022, 150: 103997.

[72] YU L, WANG Z. Multi-Robot Robust Motion Planning based on Model Predictive Priority Contouring Control with Double-Layer Corridors[J]. Applied Sciences, 2022, 12(3): 1682.

[73] LIU Z, WANG H, WEI H, et al. Prediction, planning, and coordination of thousand-warehousing-robot networks with motion and communication uncertainties[J]. IEEE Transactions on Automation Science and Engineering, 2020, 18(4): 1705-1717.

[74] VARGA L Z. Solutions to the routing problem: towards trustworthy autonomous vehicles[J]. Artificial Intelligence Review, 2022, 55: 5445-5484.

[75] HO F, GERALDES R, GONÇALVES A, et al. Decentralized multi-agent path finding for UAV traffic management[J]. IEEE Transactions on Intelligent Transportation Systems, 2022, 23: 997-1008.

[76] PIANPAK P, SON T C, TOUPS Z O, et al. A distributed solver for multi-agent path finding problems[C]// Proceedings of the First International Conference on Distributed Artificial Intelligence. 2019: 1-7.

[77] FINES K, SHARPANSKYKH A, VERT M. Agent-based distributed planning and coordination for resilient airport surface movement operations[J]. Aerospace, 2020, 7(4): 48.

[78] HUANG T, KOENIG S, DILKINA B. Learning to resolve conflicts for multi-agent path finding with conflict-based search [C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2021: 11246-11253.

[79] SURYNEK P, STERN R, BOYARSKI E, et al. Migrating Techniques from Search-based Multi-Agent Path Finding Solvers to SAT-based Approach[J]. Journal of Artificial Intelligence Research, 2022, 73: 553-618.

WANG Zihan, born in 1998, master, is a member of China Computer Federation. His main research interests include multi-agent path finding and trust path search.

TONG Xiangrong, born in 1975, Ph.D, professor, master supervisor, is a member of China Computer Federation. His main research interests include computer science, intelligent information processing and social networks.

(责任编辑:杨雪敏)