Journal of System Simulation

Volume 34 | Issue 12 Article 15

12-21-2022

Multi-robot Path Planning Based on CBS Algorithm

Qiao Qiao
Engineering Research Center of Internet of Things Technology Applications Ministry of Education, Jiangnan University, Wuxi 214122, China;, 905523350@qq.com

Yan Wang
Engineering Research Center of Internet of Things Technology Applications Ministry of Education, Jiangnan University, Wuxi 214122, China;, wangyan@jiangnan.edu.cn

Zhicheng Ji
Engineering Research Center of Internet of Things Technology Applications Ministry of Education, Jiangnan University, Wuxi 214122, China;

Multi-robot Path Planning Based on CBS Algorithm

Abstract

Abstract: Aiming at the long multi-robot planning path and long one-way search running time of conflict-based search(CBS) in the multi-agent path finding(MAPF), an improved CBS algorithm is proposed, which in a two-way A^* focus search is used to optimize the search direction and search method. *The suboptimal factor$\omega$is introduced into the underlying search function of the CBS algorithm to improve the efficiency of path search. The one-way search in the conflict search algorithm is optimized to a two-way A^* search*. The experimental results show that the path cost of the improved CBS algorithm is shortened by 14.82%, and the total running time is shortened by 10.63%.

Keywords

multi-agent path finding(MAPF), two-way search, focus search, path planning, conflict-based search(CBS)

Recommended Citation

Qiao Qiao, Yan Wang, Zhicheng Ji. Multi-robot Path Planning Based on CBS Algorithm[J]. Journal of System Simulation, 2022, 34(12): 2659-2669.

第 34 卷第 12 期 系统仿真学报© Vol. 34 No. 12
2022 年 12 月 Journal of System Simulation Dec. 2022

基于冲突搜索算法的多机器人路径规划

乔乔，王艳*，纪志成
(江南大学 教育部物联网技术应用工程中心，江苏 无锡 214122)

摘要：针对冲突搜索法(conflict-based search，CBS)在多机器人路径规划(multi-agent path finding，MAPF)过程中规划路径过长、单向搜索运行时间长等缺陷，从搜索方向和搜索方式提出一种改进的双向A*焦点搜索来优化冲突搜索算法。*将次优因子ω引入冲突搜索算法的底层搜索函数中，以提高路径搜索的效率；将冲突搜索算法中的单向搜索优化为双向A*搜索。*实验结果表明：改进的冲突搜索算法的路径成本缩短了14.82%，总运行时间缩短了10.63%。

关键词：多机器人路径规划；双向搜索；焦点搜索；路径规划；冲突搜索算法
中图分类号：TP242 文献标志码：A 文章编号：1004-731X(2022)12-2659-11
DOI: 10.16182/j.issn1004731x.joss.22-FZ0926

Multi-robot Path Planning Based on CBS Algorithm

Qiao Qiao, *Wang Yan**, *Ji Zhicheng*
(Engineering Research Center of Internet of Things Technology Applications Ministry of Education, Jiangnan University, Wuxi 214122, China)

Abstract: Aiming at the long multi-robot planning path and long one-way search running time of conflict-based search(CBS) in the multi-agent path finding(MAPF), an improved CBS algorithm is proposed, which in a two-way A* focus search is used to optimize the search direction and search method. *The suboptimal factorωis introduced into the underlying search function of the CBS algorithm to improve the efficiency of path search. The one-way search in the conflict search algorithm is optimized to a two-way A* search*. The experimental results show that the path cost of the improved CBS algorithm is shortened by 14.82%, and the total running time is shortened by 10.63%.

Keywords: multi-agent path finding(MAPF); two-way search; focus search; path planning; conflict-based search(CBS)

引言

多机器人路径规划问题是一个NP-hard问题，通过给每个机器人规划出一条路径规避静态障碍物，保证机器人之间不会发生碰撞，并最小化总运行成本。传统方法是采用机器人单独规划方法，采用避障算法给每个机器人从起点到终点规划出最短路径。如果两个机器人发生碰撞，就采用一些简单的交通规则进行处理。这种方法适用于简单工作系统，一旦机器人数量增大，就会出现拥堵，从而导致整个系统的效率变低。MAPF(multi-agent path finding)任务是考虑各种碰撞的可能性，并计划每个机器人移动/等待的指令，将其从起点移动到终点避免与其他机器人发生碰撞(即不在同一时间出现在同一个位置)，同时最小化移动的总成本。

目前，MAPF的研究主要有两大难点，第一个难点是针对本身MAPF问题，如何提高现有算

收稿日期：2022-08-07 修回日期：2022-09-23
基金项目：国家自然科学基金(61973138)；国家重点研发计划(2018YFB1701903)
第一作者：乔乔(1996-)，男，硕士生，研究方向为机器人路径规划。E-mail：905523350@qq.com
通讯作者：王艳(1978-)，女，博士，教授，研究方向为制造系统能效优化。E-mail：wangyan@jiangnan.edu.cn

法的求解效率和解的质量[1]。传统的MAPF算法大多基于A*算法、Dijkstra算法等进行优化。MAPF在物流仓储，自动化仓库等场所也有着很频繁的应用，缩短算法运行时间，提高算法求解效率是一个很有意义的问题。

第二个难点是如何把MAPF应用到实际问题当中，并处理不同实际问题带来的约束。除了仓储系统，MAPF还可以应用于飞机调度，需要考虑如何使用MAPF方法控制飞机的起飞和降落。这个难点在于飞机的控制包含很多不确定性因素，例如，天气干扰、人为因素干扰，因此将MAPF模型和随机模型结合时要考虑到各种因素干扰，来控制整体的调度及运行。MAPF还可以应用到很多机器人系统中，如自动停车系统，无人车、无人机以及水下机器人，对这些系统一方面要规避静态和动态的碰撞，还要考虑各种机器人不同的动力学约束，如速度、加速度、转角，以及各种干扰的约束。

神经网络，蚁群算法和遗传算法陆续应用到路径规划问题中，但都有着计算量大，容易陷入停滞等问题，没有很高的求解效率。针对该问题，诸多学者展开了对MAPF算法的研究。最简单的MAPF算法有A*搜索算法和代价增长树搜索算法。这些算法实现比较简单，在机器人密集问题中表现良好。但是随着问题规模的增大，效率会降低，时间代价和空间代价较高。王洪斌等[2]通过采用自适应圆弧优化算法和加权障碍物步长调节算法来提升算法和路径规划的效率。强宁等[3]提出了基于PSO和三次样条插值路径规划的方法，通过以路径节点N为粒子编码，降低算法编码的维度，从而优化算法的性能来提高算法的运行效率。张云旺[4]针对大规模连续任务场景下多机器人路径规划问题，提出了拥塞控制的策略应用于跳点算法(jump point search，JPS)，将机器人的任务节点作为临时停靠点来缓解机器人堵塞情况，从而提高货物运输的效率。

冲突搜索法(conflict-based search，CBS)算法是用于解决多机器人路径规划最优解的算法，该算法分别在低层和高层对机器人路径进行规划来避免其发生冲突。杨伦[5]提出基于ICBS(improved CBS)算法，在CBS算法基础上添加不同优化算法消除对顶冲突。ECBS(enhanced CBS)算法采用了焦点搜索，选出发生冲突数较小的节点进行扩展，能够更快找到次优解。万千[6]基于CBS算法提出了一种EPCBS(enhanced parallel CBS)算法，该算法通过沿用ECBS算法的前一次搜索过程中产生的约束，对机器人路径进行快速调整，采用多线程方法来加快ECBS算法的运行效率。Li等[7]提出了EECBS(explicit estimation CBS)搜索算法，该算法包含了显式估计搜索和3个启发式算法来提高算法搜索的精度，从而得到更加精确的路径估计代价$h(n)$。综上所述，现有的CBS及其改进算法均存在运行效率低等问题，因此，本文提出双向A*焦点搜索，分别从正向和反向进行焦点搜索，在双端同时进行搜索来加快ECBS算法的运行时间。通过一个开放列表存入所有的节点，并且按照最小代价f的值去排列。最小的设为$f_{\min}$，其为最优解的一个低界，而有界次优解的要求只需要最终解的代价比$\omega \cdot f_{\min}$小。

1 问题描述

MAPF问题由无向图$G=(V,E)$和一组k个机器人$\{a_1, a_2, \cdots, a_k\}$组成，每个机器人$a_i$都有单独的起点$s_i \in V$和单独的终点$g_i \in V$。时间离散为$t$，在每个时间段，机器人可执行从当前顶点移动到相邻顶点的移动动作或停留在当前顶点的等待动作，这两个动作行动成本均为1[8]。任务是为每个机器人a_i计划一系列移动/等待动作，将其从s_i移动到g_i，同时避免与其他机器人发生碰撞(即不同时占用同一位置)，并最小化累积成本[9]。

机器人a_i的路径p_i是从起始顶点s_i开始移动到目标顶点g_i结束的顶点序列，p_i中每个顶点对应机器人在每个时间段t停留的顶点。p_i中两个相

DOI: 10.16182/j.issn1004731x.joss.22-FZ0926

邻顶点若为相邻顶点，则为机器人a_i移动，若为相同顶点，则为机器人a_i等待。

路径p_i的成本是机器人a_i从起始顶点s_i移动到目标顶点g_i的步数或前进时间，也是路径长度$|p_i|$。在计算路径p_i的成本时，排除了机器人a_i在目标顶点g_i等待的时间[10]。将路径的总成本定义为其组成路径成本的总和$\sum|p_i|$。

如果机器人在相同的顶点或在相同的时间段以相反的方向经过相同的边则视为碰撞。两条路径之间的碰撞是一个数组$<a_i, a_j, v, t>$，机器人a_i和机器人a_j将在时间段t占据同一个顶点v。解决方案是一组无碰撞路径$\{p_1, p_2, \cdots, p_m\}$，最优解是其总成本$\sum|p_i|$最小的解。用于寻找MAPF的有效解的算法可分为3类：最优解算法、次优解算法、有界次优算法[11]。

如图1所示，一个机器人a_1要从S_1移动到G_1，一个机器人a_2要从S_2移动到G_2。MAPF算法最开始先忽略另一个机器人，给每一个机器人规划一条最短路径。检查当前的最短路径有没有发生碰撞，算法根据其中一个机器人是否执行当前指令分别生成2条分支并且添加额外的一个约束。之后在各自的分支中重新规划路线，左边给机器人a_1重新规划一条路径，右边给机器人a_2重新规划一条路径。重复之前的过程，直到找到一个节点，里面的路径没有任何碰撞。

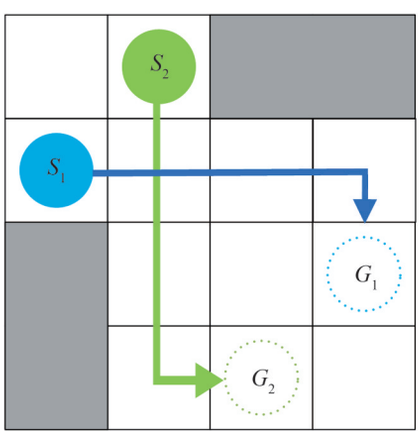


图1　机器人移动路径图
Fig. 1　Robot moving path diagram

2　冲突搜索算法

CBS算法为解决MAPF最优解算法。在CBS中，给机器人添加约束条件以解决2个机器人之间的碰撞。一个机器人a_i的约束$<a_i, a_j, v, t>$表示机器人a_i不能在时间t位于点v，或者机器人a_i从时刻t到时刻$t+1$不能经过边v。机器人a_i的连续路径是满足所有约束的路径，解决方案是仅由连续路径组成的解决方案。尽管机器人的各条路径都满足各自的约束条件，但机器人之间仍可能存在碰撞，满足约束条件的解决方案可能无效。

因此，为了解决机器人之间的碰撞问题，CBS分为2个层次的搜索过程。在低层，其搜索过程与普通的路径规划方法类似，如Dirkstra、A*等，为每个机器人搜索出一条有效路径。但其不同之处在于：① 搜索过程中需要考虑额外的约束，即高层次搜索中添加的冲突；② 在搜索过程中需要考虑原地等待的情况。

在高层为不同的机器人生成约束。如果这些路径发生碰撞，则添加新约束以解决其中一个碰撞，并再次调用低层对这个机器人的路径进行规划。

CBS高层搜索约束二叉树CT(constraint tree)，CT中的每个节点N都包含：

(1) 施加在机器人上的约束集(N_{con})；

(2) 满足约束条件的解决方案(N_{sol})，即每个机器人的路径都满足该节点N中约束集的约束；

(3) 当前节点N解决方案的总成本(N_{cost})。

CT的根节点包含一组空约束，CT中的节点的后继节点继承其父节点的约束并为单个机器人添加单个新约束。通过低层搜索寻找N_{sol}，当N_{sol}有效时，即所有机器人的路径没有发生碰撞，CT节点N是目标节点。CBS在高层的CT上执行最佳优先搜索，将CT中的节点按成本排序。

在二叉树中处理节点：给定一个二叉树CT节点N，对单个机器人调用低层搜索得到与其在N中的满足约束条件的最佳路径。任何最佳单机器

人寻路算法都可以针对低层CBS进行修改。通常使用A*和最短距离启发式算法，当每个机器人在低层找到满足约束条件的路径时，这些路径就通过模拟机器人沿其计划的路径移动(N_{sol})，并与其他机器人进行验证。如果所有机器人都顺利到达目的地而没有任何碰撞，则将此CT节点N声明为目标节点，并返回N_{sol}。若在执行算法时发现2个(或更多)机器人a_i和a_j的碰撞$<a_i, a_j, v, t>$，则算法停止并且该节点被声明为非目标节点。

处理碰撞：给定一个非目标CT节点N，其解N_{sol}包括碰撞$<a_i, a_j, v, t>$，在任何有效解中，至多有一个碰撞机器人a_i和a_j可能在时间t占据顶点v。因此，至少有一个约束，$<a_i, v, t>$或$<a_j, v, t>$成立。CBS生成2个新的CT节点作为节点N的子节点，每个子节点都将这些约束中的一个添加到先前的一组约束N_{sol}。除了根节点，对于每个CT节点低层搜索仅针对一个机器人为其添加新的约束[12]。

如图2所示，有2个机器人a_1、a_2，分别在各自的起点S_1和S_2上，需要到各自的终点G_1、G_2。如图2所示，根节点包含一组空约束。在开始阶段，低层返回每个机器人的最优解，$<S_1, A_1, C, G_1>$为机器人a_1的最优路径，$<S_2, B_1, C, G_2>$为机器人a_2的最优路径，因此该节点的总成本为6。所有的信息都存在这个节点中，然后将根节点插入已生成排序的OPEN列表中，并将其展开。

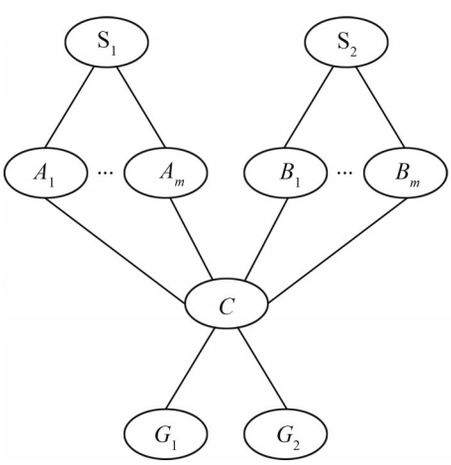


图2 MAPF事例
Fig. 2 MAPF case

验证2个机器人的解决方案，如图3所示，当2个机器人在第2步到达顶点C时会发生碰撞。这会产生碰撞$<a_1, a_2, C, 2>$。根节点则被声明为非目标，并生成2个子节点来解决碰撞。左边的子节点添加约束$<a_1, C, 2>$，而右边的子节点添加约束$<a_2, C, 2>$。然后调用低层算法重新搜索机器人a_1的路径，查找左子节点，以找到满足新约束的最优路径。为此，机器人a_1必须在S_1(或A_1)等待一个时间t，并为机器人a_1返回的路径$<S_1, A_1, A_1, C, G_1>$，机器人a_2返回的路径$<S_2, B_1, C, G_2>$在左子节点中保持不变。其中成本计算为单个机器人成本的总和，因此左子节点的成本为7。以类似的方式生成正确的右子节点，其成本也是7。2个子进程都被添加到OPEN列表中。在最后一步中，选择左子节点进行展开，并验证底层路径，若不存在碰撞，则左边的子节点被声明为目标节点，并返回其解决方案；若产生碰撞，则继续生成子节点来解决碰撞，直到生成一条没有发生碰撞的解决方案。

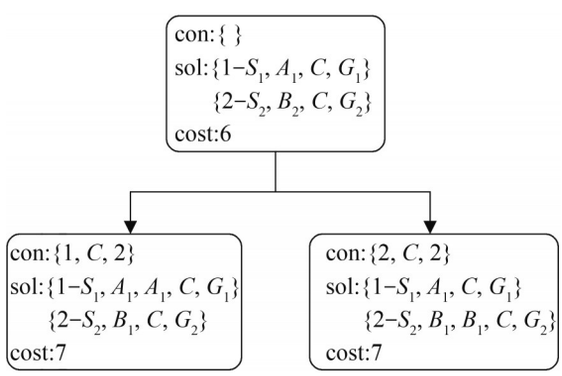


图3 二叉树CT
Fig. 3 Binary tree CT

3 增强型冲突搜索算法

CBS算法在检测到发生冲突的节点时，会根据冲突创建2个子节点，并对2个子节点添加其相应的约束条件，并在低层重新搜索单个机器人的路径，直到新的路径没有发生冲突。这种算法只能够处理已发现的冲突，重新搜索路径时还可能

DOI: 10.16182/j.issn1004731x.joss.22-FZ0926

会与其他节点发生冲突[13]。

ECBS 算法采用 A* 算法为每个机器人 a_i 从给定的起点 $s_i \in V$ 到终点 $g_i \in V$ 规划出最优路径，根据估计函数对每一个搜索的位置进行评估，得到中间节点，再从中间节点进行搜索直到到达目标节点。当采用焦点搜索后，A* 算法创建 FOCAL 列表存放符合要求的节点，该列表中的节点 $f(n)$ 值不大于最优节点 $f_{\min}(n)$ 的 ω 倍，即

$$\text{FOCAL}=\{n \in \text{OPEN} \mid f(n) \leqslant \omega \cdot f_{\min}(n)\} \tag{1}$$

式中：n 为路径节点；OPEN 列表为 A* 算法搜索当前节点的所有邻居节点的集合；$f(n)$ 为 OPEN 列表中节点 n 的从起点经由节点 n 到终点的代价估计；$f_{\min}(n)$ 为 OPEN 列表中代价估计最小值。FOCAL 列表中的节点按照每个节点机器人 a_i 发生碰撞的次数进行排序，来决定该列表中节点的优先级。

图 4 为 ECBS 算法的流程。初始化时候先计算出所有的机器人 a_i 的解和对应的总成本，令约束树 CT 为空；然后每一轮弹出约束树成本最小的节点 N，判断 N 中是否存在一个碰撞，生成 2 个子节点 N_1 和 N_2，并且都只传递继承节点 N 的内容；接着将碰撞分成 2 个约束节点，一个加入到 N_1 约束树中，一个加入到 N_2 约束树中，对于 N_1 而言，机器人 a_i 的约束发生了变化，因此需要对 a_i 进行重新规划，规划时需要在约束树下进行路径规划；最后更新机器人 a_i 的路径和成本并插入到优先队列 OPEN 列表中。接着对 N_2 进行同样的操作，直到取出的节点检测没有发生碰撞时返回节点 N。

该算法主要改进了高低层的搜索算法，CBS 算法在低层只需要找出一条满足约束树的路径即可，搜索路径时比较盲目，这种盲目的搜索容易导致规划出的路径再次出现新的碰撞。因此，ECBS 在 CBS 的框架基础上对高低层进行了搜索改进，采用了焦点搜索技术，来提高 CBS 的速度[14]。

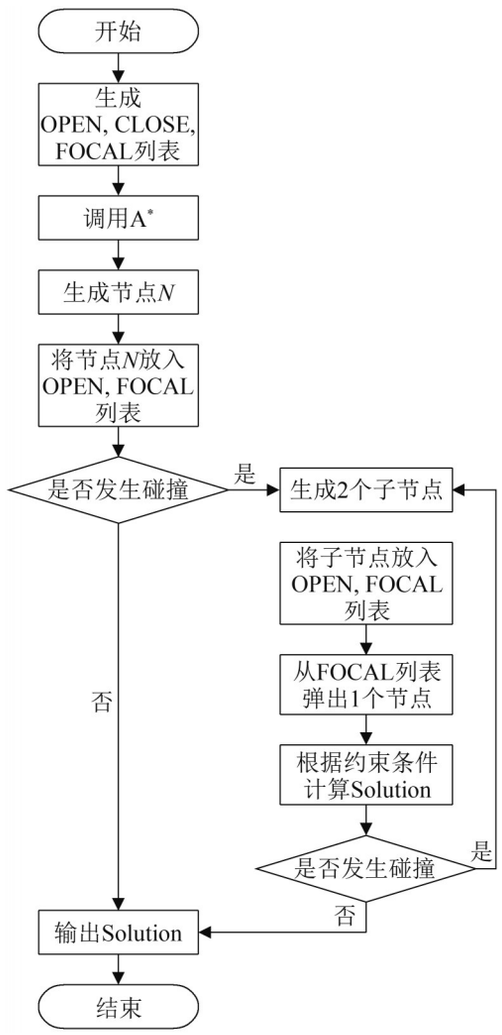


图 4 ECBS 运行流程
Fig. 4 ECBS operation process

3.1 低层焦点搜索

在低层，ECBS 为机器人 a_i 寻找路径，给高层输出路径的总成本和最优路径成本的下界 $lb(n)=f_{\min}(n)$。低层 OPEN_i 列表中将节点 n 按常规启发式 $f(n)=g(n)+h(n)$ 的值递增顺序进行排序。设 N^* 为低层 OPEN_i 列表中最小 $f(n)$ 节点，ω 为指定次优因子。在低层焦点搜索开始时，低层 OPEN_i 列表只有根节点 N_0，因此根节点 N_0 也为 N^*，函数最小值 $f_{\min}(n)$ 为 $f(N_0)$。低层焦点搜索还包含一个低层 FOCAL_i 列表，OPEN_i 列表中所有满足 $f(n) \leqslant \omega \cdot f_{\min}(n)$ 的节点 N 在该列表中。定义 $f_2^i(n)$ 为机器人 a_i 从起点 S_i 移动到顶点 v 与其他机器人发生碰撞的次数。将

低层 $\mathrm{FOCAL_i}$ 列表中的节点优先按 $f_2^i(n)$ 递增进行排序。

在进行节点拓展的时候，低层焦点搜索在低层 $\mathrm{FOCAL_i}$ 列表中取出 $f_2(n)$ 的最小值的节点进行拓展。证明可以得到 $lb(n) \leqslant \left| N_{\text{paths}} \right| \leqslant \omega \cdot lb(n)$。其中低层输出的解决方案的 $f_{\min}^*(n) \leqslant \omega \cdot f_{\min}(n)$。低层焦点搜索为机器人 a_i 返回一条路径，其成本最多为 $\omega \cdot c^*$：

$$lb(n) \leqslant \left| N_{\text{paths}} \right| \leqslant \omega \cdot lb(n) \leqslant \omega \cdot c^* \tag{2}$$

定义 $hb(n) = \omega \cdot lb(n)$ 为 CT 节点机器人 a_i 的上界，低层焦点搜索也返回机器人 a_i 最优路径的成本的下界 $lb(n)$。

3.2 高层焦点搜索

在高层，ECBS 也进行 CT 焦点搜索。给定 CT 节点 N，定义其下界 $LB(N) = \sum_{i=1}^{k} f_{\min}(i)$，上界 $UB(N) = \omega \cdot \sum_{i=1}^{k} f_{\min}(i)$。将 CT 节点 N 中所有路径对之间产生的碰撞数定义为 $f_2(N)$。令 $LB = \min(LB(N) | N \in \text{OPEN})$。由于 $LB(N)$ 是 CT 节点 N 以下解的最小成本的下界，因此 LB 是最优路径成本的下界，记为 C^*。高层焦点搜索根据 $LB(N)$ 对高层 OPEN 列表中的 CT 节点 N 进行排序，将成本最多为 $\omega \cdot LB$ 的 CT 节点加入到高层 FOCAL 列表中，其中 ω 为与低层相同的次优因子。ECBS 扩展高层 FOCAL 列表中 $f_2(N)$ 值最小的 CT 节点，即产生碰撞次数最少的 CT 节点。由于 ECBS 只选择成本最多为 $\omega \cdot LB$ 且 $LB \leqslant \omega \cdot C^*$ 的 CT 节点，因此可以找到一个成本最多为 $\omega \cdot C^*$ 的解，即：

$$LB \leqslant N_{\text{cost}} \leqslant \omega \cdot LB \leqslant \omega \cdot C^* \tag{3}$$

定义 $HB = \omega \cdot C^*$ 为高层焦点搜索的上界。

CBS 和 ECBS 都不会扩展成本高于最优解 ω 倍的节点。此外，所有有效解始终与 OPEN 列表中的 CT 节点一致。由于两者都是系统搜索，如果存在，该算法最终会找到合适的解。

4 优化 ECBS 算法

由于 ECBS 算法在低层进行路径搜索时，仍采用普通 $\mathrm{A^*}$ 算法，每次计算路径所耗费的时间非常多，本文就该问题对低层 $\mathrm{A^*}$ 算法进行优化，结合焦点搜索提出了双向 $\mathrm{A^*}$ 焦点搜索算法，来提高 ECBS 的运行速率。

4.1 搜索邻居点

ECBS 在低层包含一个 OPEN 列表和一个 FOCAL 列表。正向搜索时，从 FOCAL 列表中取出 1 个 CT 节点 N，该节点生成 8 个方向的邻居点和 1 个包含时间 t 的抽象邻居点，将节点 N 生成的 9 个邻居点判断其是否满足约束条件 $<v, edge, collision>$，若不满足则删除不满足的邻居点，若满足，则输出该邻居点放入 FOCAL 列表中，如图 5 所示。反向搜索同理，与正向搜索不同的是，反向 FOCAL 列表的节点只生成 8 个方向的邻居点，只判断其是否满足碰撞约束条件 $<collision>$，如图 6 所示。

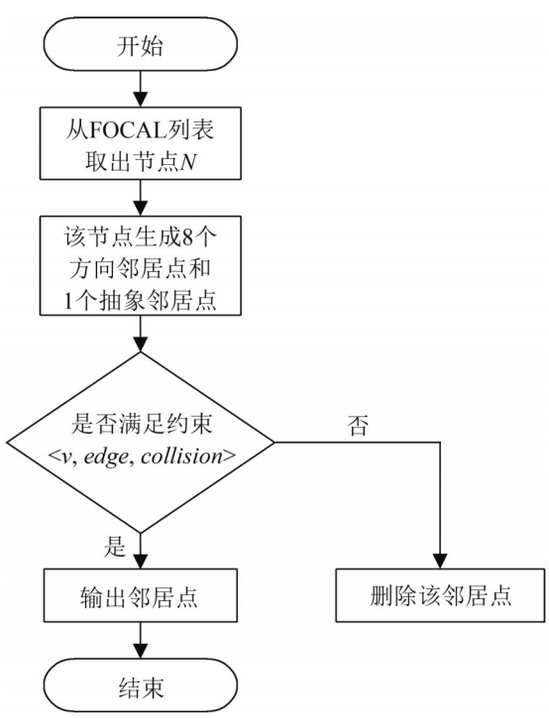


图 5 正向搜索邻居点
Fig. 5 Looking for neighbor points

DOI: 10.16182/j.issn1004731x.joss.22-FZ0926

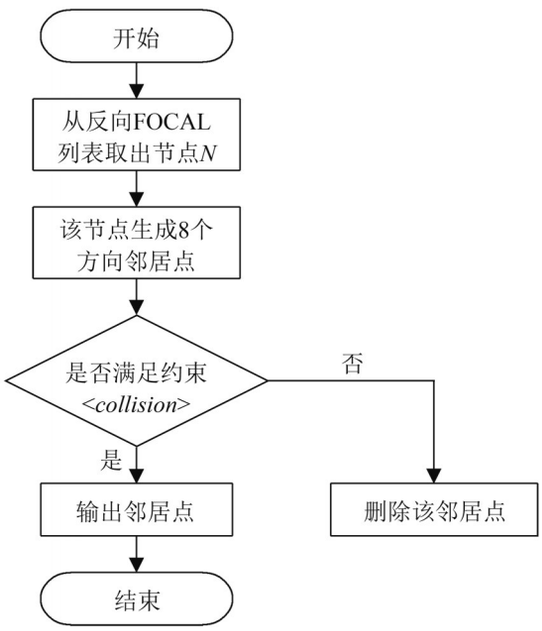


图6 反向搜索邻居点
Fig. 6 Finding neighbor points in reverse

4.2 双向A*搜索算法

算法1 双向A*算法流程

输入：起始格添加到“正向开启列表”，把终点添加到“反向开启列表”

将起点设置为正向开启列表中$f(n)$值最低的格子cur_grid_1，将终点设置为反向开启列表中$f(n)$最低的格子cur_grid_2

do
{
以cur_grid_1为当前格
将其放入正向CLOSE列表
对当前格相邻的8个格子
if(不可通过||已经在“正向关闭列表”)
{
什么也不做
}
if(不在“正向开启列表”)
{
把它添加进"正向开启列表"，把当前格作为这一格的父节点，并根据cur_grid_2，计算这一格的$f(n)$、$g(n)$和$h(n)$。
}
if(在“正向OPEN列表”)
{
if (用$g(n)$值为参考检查新的路径是否更好)
{
把这一格的父节点改成当前格，并重新计算这一格的$f(n)$、$g(n)$和$h(n)$。
}
}
if(该邻居点满足$f(n)\leqslant\omega\cdot f_{\min}(n)$)
{
将该邻居点放入正向FOCAL列表
}
if(cur_grid_2已经在正向OPEN列表)
找到路径；
if(发生碰撞)
{
进行碰撞处理
} break;
以cur_grid_2为当前格
将其放入反向CLOSE列表
对当前格相邻的8个格子
if(不可通过||已经在“反向CLOSE列表”)
{
什么也不做
}
if(不在“反向OPEN列表”)
{
把它添加进反向OPEN列表，把当前格作为这一格的父节点，并根据cur_grid_1，计算这一格的$f(n)$、$g(n)$和$h(n)$。
}
if(在“反向OPEN列表”)
{
if (用$g(n)$值为参考检查新的路径是否更好)

{

把这一格的父节点改成当前格，并重新计算这一格的$f(n)$、$g(n)$和$h(n)$。

}

}

if(该邻居点满足$f(n) \leqslant \omega \cdot f_{\min}(n)$)

{

将该邻居点放入反向FOCAL列表

}

if(cur_grid_1已经在反向OPEN列表)

找到路径；

if(发生碰撞)

{

进行碰撞处理

} break;

While(正向OPEN列表不为空&&反向OPEN列表不为空)

}

4.3 双向A*焦点搜索算法

图7为碰撞处理的整个流程。在低层正向搜索，取出一个CT节点N为起点生成8个方向的邻居点和一个包含时间t的抽象邻居点，反向搜索生成8个方向的邻居点。当正向搜索的点出现在反向OPEN列表中，说明2个方向搜索在某个节点相遇，停止搜索。正向回溯生成正向*solution*，反向添加约束时间t回溯，若反向回溯满足约束条件则和正向*solution*一起输出*solution*。若不满足约束条件，则以该点为起点开始重新双向搜索，直到反向回溯的*solution*全部满足约束条件。

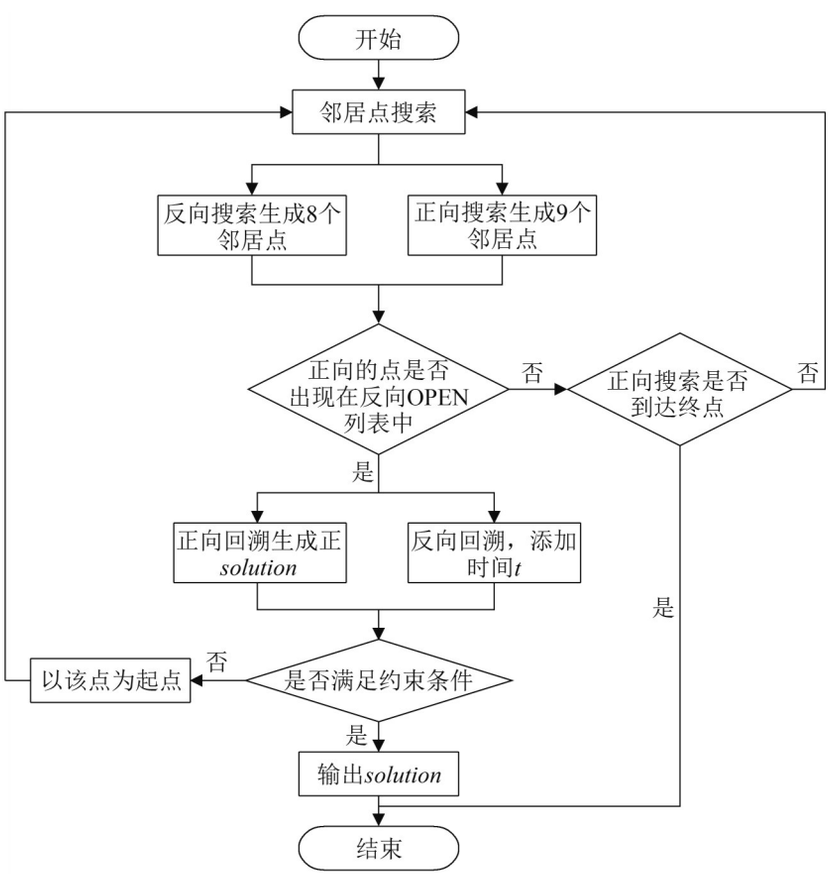


图7 碰撞处理流程

Fig. 7 Collision handling process

DOI: 10.16182/j.issn1004731x.joss.22-FZ0926

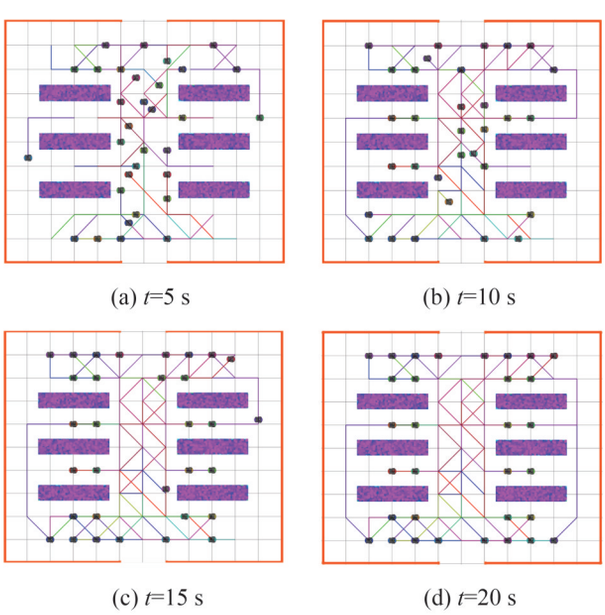
5 仿真

5.1 仿真环境

本文提出的算法是用C++实现的，并在一台运行 Ubuntu 20.04、Intel i5-11300HQ @3.10 GHz CPU 和 16 GB RAM 的笔记本电脑上执行。使用 OctoMap 表示环境占用图，在网格大小为 $D = 1$ m 的网格图中规划离散路径。在本例中，时间间隔 ΔT 设置为 1 s。

5.2 运行效率和解决方案的质量

本文开发方法主要应用在仓库中的物料搬运，本文构建了一个仓库环境来评估所提出方法的性能。环境大小为 10 m×12 m，包含 6 个 3 m×0.6 m 的架子。每个机器人的起点和目标位置随机分配在环境边界或货架附近的取物点。图 8 为 32 个机器人在仿真环境中的示例，平均运行时间为 2.7 s。表 1 为在仓库中 ECBS 运行时间对比，整个过程 ECBS 耗时约 0.01 s。与优化前 ECBS 相比，本文提出的优化方法计算效率更高。当机器人规模变大时，算法优化的效果更加显著。



(a) t=5 s　(b) t=10 s

(c) t=15 s　(d) t=20 s

图 8　32 个机器人在仓库中运行轨迹
Fig. 8　32 robots run track in warehouse

表 1　运行时间对比
Table 1　Running time comparison

机器人数量/个	ECBS 运行时间/s	改进 ECBS 运行时间/s	运行时间降低百分比/%
4	0.000 43	0.000 42	2.32
8	0.000 59	0.000 58	1.69
12	0.000 85	0.000 72	15.3
16	0.001 29	0.001 14	11.6
20	0.002 43	0.002 50	−2.88
24	0.005 93	0.004 14	30.2
28	0.017 94	0.017 39	3.07
32	0.071 10	0.018 10	74.5

文献[15]解释了次优系数随着地图的扩大而减小。如图 9 所示，本文评估了 6 个不同的 ω 值，在这些不同 ω 值下，随着机器人数量逐渐变多，运行时间的差异也逐渐变大。根据地图运行环境以及图 9 数据对比，当次优系数设为 1.5 时，机器人完成路径规划的总运行时间最低，而且能够更加稳定的完成实验。

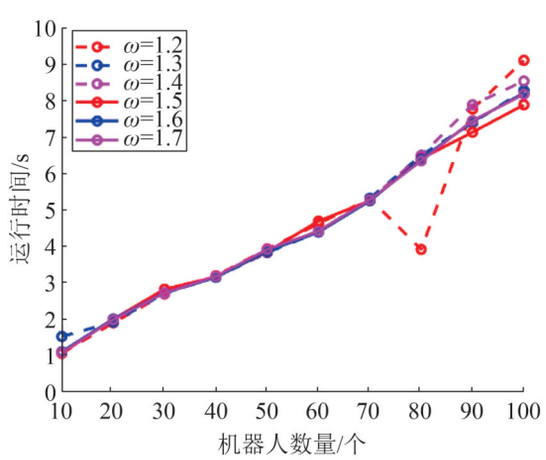


图 9　不同次优因子路径成本对比
Fig. 9　Comparison of path costs with different suboptimal factors

为了证明优化后算法解决方案的质量，扩大实验规模，根据文献[16]的方法，将仿真地图规模扩大至 50 m×50 m，机器人数量增加至 100 个，仿真结果如图 10 所示。该图的障碍物为 3 m×3 m 的正方形，呈九宫格环绕放置在地图中。100 个机器人在次优系数设为 1.5 的情况下完成路径规划的总运行时间大约为 8.2 s 左右。

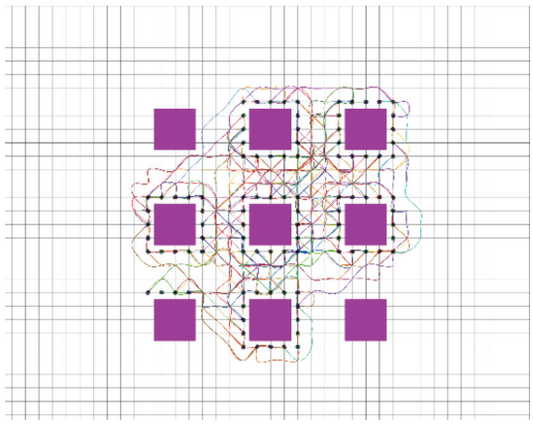


图 10　100 个机器人在仓库中运行轨迹
Fig. 10　100 robots run track in warehouse

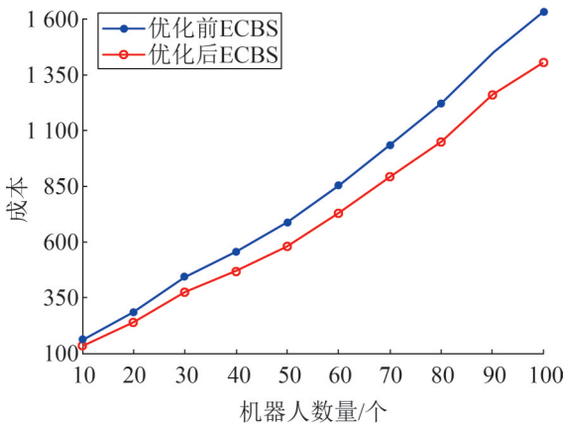


图 12　仓库中路径成本对比
Fig. 12　Route cost comparison in warehouse

ECBS 算法求解 100 个机器人情况的时间约为 2.63 s，优化后 ECBS 算法求解时间约为 1.81 s。图 11 和图 12 分别为把次优因子设为 1.5 的情况下，算法优化前后的运行时间和路径成本。可以看出，优化后算法的运行时间和路径成本随着机器人数量的增加线性增加。优化前算法的路径成本普遍高于优化后算法的路径成本。随着机器人数量的增加，发生碰撞的次数越多，相差的路径成本也越大。当机器人数量较少时，优化前算法的总运行时间略低于优化后算法，当机器人数量超过 30 个后，优化后算法的总运行时间略低于优化前算法的总运行时间。优化后算法路径成本平均比优化前低 14.82%，总运行时间比优化前低 10.63%。

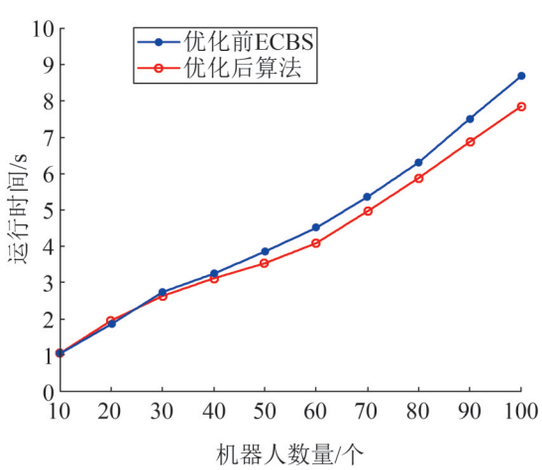


图 11　仓库中总运行时间对比
Fig. 11　Comparison of total running time in warehouse

6　结论

本文提出了一种多机器人在有障碍的地图上寻找一条无碰撞的有效路径的规划算法。该搜索方法寻找无碰撞，时间最优的路径，将路径规划问题分为低层和高层来处理。针对 ECBS 算法求解成本高、低层搜索单个机器人路径的过程太过繁琐的问题，提出了一种改进的 ECBS 算法，使用双向焦点搜索来改进 ECBS 算法低层中的 A* 算法，提高了 ECBS 在低层单机器人路径的搜索速度。实验结果表明，当机器人数量为 4 或更少时，总运行时间和路径成本相差不大，2 种方法的性能相当。但是随着机器人数量的增加，机器人之间发生碰撞的次数越来越多，运行时间和路径成本相差越大。使用这种方法可以加快对单个机器人路径的搜索速度，在大规模场景下 100 个机器人能够在 8.2 s 完成对路径的规划，优化后算法的总运行时间比优化前降低 10.63%，路径成本降低 14.82%。优化后的 ECBS 算法返回的解决方案接近最优解。

参考文献：

[1] 刘晶, 姚维, 章玮. 移动机器人全覆盖路径规划算法研究[J]. 工业控制计算机, 2019, 32(12): 52-54.
Liu Jing, Yao Wei, Zhang Wei. Research on Complete Coverage Path Planning Algorithm of Mobile Robots[J]. Industrial Control Computer, 2019, 32(12): 52-54.

DOI: 10.16182/j.issn1004731x.joss.22-FZ0926

[2] 王洪斌, 尹鹏衡, 郑维, 等. 基于改进的A*算法与动态窗口法的移动机器人路径规划[J]. 机器人, 2020, 42(3): 346-353.
Wang Hongbin, Yin Pengheng, Zheng Wei, et al. Mobile Robot Path Planning Based on Improved A* Algorithm and Dynamic Window Method[J]. Robot, 2020, 42(3): 346-353.

[3] 强宁, 高洁, 康凤举. 基于PSO和三次样条插值的多机器人全局路径规划[J]. 系统仿真学报, 2017, 29(7): 1397-1404.
Qiang Ning, Gao Jie, Kang Fengju. Multi-Robot Global Path Planning Based on PSO and Cubic Spline Interpolation[J]. Journal of System Simulation, 2017, 29 (7): 1397-1404.

[4] 张云旺. 物流中心多任务场景下多AGV的路径规划算法设计[D]. 哈尔滨: 哈尔滨工业大学, 2021.
Zhang Yunwang. Path Planning Algorithm Design of Multi-AGV in Multi-Task Scenario of Logistics Center [D]. Harbin: Harbin Institute of Technology, 2021.

[5] 杨伦. 消解基于冲突搜索的多智能体路径规划算法中的对顶冲突[D]. 长春: 吉林大学, 2020.
Yang Lun. Resolving the Top Conflict in the Multi-Agent Path Planning Algorithm Based on Conflict Search[D]. Changchun: Jilin University, 2020.

[6] 万千. 基于CBS算法的物流分拣多AGV路径规划的研究[D]. 哈尔滨: 哈尔滨工业大学, 2018.
Wan Qian. Research on Multi-AGV Path Planning for Logistics Sorting Based on CBS Algorithm[D]. Harbin: Harbin Institute of Technology, 2018.

[7] Li Jiaoyang, Tinka Andrew, Kiesel Scott, et al. EECBS: A Bounded-Suboptimal Search for Multi-Agent Path Finding [C]//35th AAAI Conference on Artificial Intelligence. California, SC: Association for the Advancement of Artificial Intelligence, 2021: 12353-12362.

[8] 朱宝艳. 移动机器人全覆盖路径规划算法研究[D]. 淄博: 山东理工大学, 2017.
Zhu Baoyan. Research on Complete Coverage Planning Algorithm for Mobile Robots[D]. Zibo: Shandong University of Technology, 2017.

[9] 赵萍, 雷新宇, 陈波芝, 等. 基于TI-A*的多机器人动态规划协调方法研究[J]. 系统仿真学报, 2019, 31(5): 925-935.
Zhao Ping, Lei Xinyu, Chen Bozhi, et al. Research on Multi-Robot Dynamic Programming Coordination Method Based on TI-A*[J]. Journal of System Simulation, 2019, 31(5): 925-935.

[10] 李颀, 汪伟. 多全向轮协同分拣平台的路径规划[J]. 系统仿真学报, 2021, 33(3): 698-709.
Li Qi, Wang Wei. Path Planning of Multi-omnidirectional Wheel Collaborative Sorting Platform [J]. Journal of System Simulation, 2021, 33(3): 698-709.

[11] 陈志梅, 李敏, 邵雪卷, 等. 基于改进RRT算法的桥式起重机避障路径规划[J]. 系统仿真学报, 2021, 33(8): 1832-1838.
Chen Zhimei, Li Min, Shao Xuejuan, et al. Obstacle Avoidance Path Planning of Bridge Crane Based on Improved RRT Algorithm[J]. Journal of System Simulation, 2021, 33(8): 1832-1838.

[12] 黄晓冬, 苑海涛, 毕敬, 等. 基于DQN的海战场舰船路径规划及仿真[J]. 系统仿真学报, 2021, 33(10): 2440-2448.
Huang Xiaodong, Yuan Haitao, Bi Jing, et al. Ship Path Planning and Simulation in Naval Battlefield Based on DQN[J]. Journal of System Simulation, 2021, 33(10): 2440-2448.

[13] 林彬, 韩光辉, 宋晨晨, 等. 基于辐射扫描算法的机器人路径规划与仿真[J]. 系统仿真学报, 2021, 33(1): 84-90.
Lin Bin, Han Guanghui, Song Chenchen, et al. Robot Path Planning and Simulation Based on Radiation Scanning Algorithm[J]. Journal of System Simulation, 2021, 33(1): 84-90.

[14] 刘恒麟. 多AGV路径规划算法的研究与实现[D]. 南京: 东南大学, 2020.
Liu Henglin. Research and Implementation of Multi-AGV Path Planning Algorithm[D]. Nanjing: Southeast University, 2020.

[15] Li Jiaoyang, Tinka Andrew, Kiesel Scott, et al. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses [C]//35th AAAI Conference on Artificial Intelligence. California, SC: Association for the Advancement of Artificial Intelligence, 2021: 11272-11281.

[16] Li J, Ran M, Xie L. Efficient Trajectory Planning for Multiple Non-holonomic Mobile Robots Via Prioritized Trajectory Optimization[C]//IEEE Robotics and Automation Letters, 2021, 6(2): 405-412. DOI: 10.1109/LRA.2020.3044834.