

Multi-Bug 全局路径规划算法研究

彭艳¹ 鲍凌志¹ 瞿栋¹ 解杨敏²

(1. 上海大学无人艇工程研究院, 上海 200444; 2. 上海大学上海市智能制造与机器人重点实验室, 上海 200444)

摘要: 提出一种基于 Multi-Bug 思想的非搜索全局路径规划算法。在 Multi-Bug 算法中, 借用传统 Bug 算法的寻路逻辑, 加入遇到障碍物时的爬虫分裂规则及爬虫死亡条件判断规则, 直至其中一只爬虫以相对最优路径抵达终点, 从而实现多路径并行运算的局部最优寻路策略。利用栅格法对多类障碍物、迷宫类地图等环境进行建模, 并与 Dist-Bug 算法、RRT* 和 A* 算法进行路径长度及运算时间的对比仿真实验, 结果表明, 采用 Multi-Bug 算法获得的路径长度和用时都表现得更加稳定; 与获得最短路径的 A* 算法相比, Multi-Bug 算法获得的平均路径长度仅增加了 16.8%, 平均用时减少了 86.5%。经理论分析及仿真验证, Multi-Bug 算法时间复杂度为 $O(n)$, 具有路径较短、时效性强、算法通用性和稳定性好的路径规划性能。

关键词: 路径规划; Multi-Bug 算法; Dist-Bug 算法; 栅格法

中图分类号: TP242 文献标识码: A 文章编号: 1000-1298(2020)06-0375-10

OSID:



Global Path Planning Algorithm of Multi-Bug

PENG Yan¹ BAO Lingzhi¹ QU Dong¹ XIE Yangmin²

(1. Research Institute of USV Engineering, Shanghai University, Shanghai 200444, China

2. Shanghai Key Laboratory of Intelligent Manufacturing and Robotics, Shanghai University, Shanghai 200444, China)

Abstract: Path planning is one of the key technologies to realize its ability of autonomous action in complex environment. A non-search global path planning algorithm called Multi-Bug was proposed. It combined the merits of short path length of global path planning and small computation cost of the Bug algorithms, which generated better overall performance compared with traditional path planning methods. This method had great advantages in the situation of high timeliness and general path requirements. In the Multi-Bug algorithm, the simple path-finding logic that followed the wall in the Bug algorithm was adopted, and bug splitting and death rules were added into the framework. The algorithm was ended when one of the bugs arrived at the target or all of the bugs met the death rules and dead, which permitted multi-path parallel computation for local optimal planning solution. Grid maps, including multiple types of obstacles and various mazes, were used to testify the Multi-Bug algorithm. In order to ensure the reliability of operation results, each algorithm in the same grid map was run 10 times, and its operation time was averaged. Compared with Dist-Bug algorithm, RRT* and A* algorithms, it showed much better performance stability in terms of the path length and the computational time cost. Especially when compared with the A* algorithm, which provided the shortest path length, it decreased the calculation time by more than four fifths with grid map size as 50×50 and only increased the path length by less than one fifth. In 500×500 grid map, the time was less than one ten-thousandth of that of A* algorithm. By the theoretical analysis and the simulation verification, Multi-Bug algorithm had a time complexity as $O(n)$, provided relatively short path, required less computational cost, and had good stability and versatility for various environments.

Key words: path planning; Multi-Bug algorithm; Dist-Bug algorithm; grid method

收稿日期: 2019-10-31 修回日期: 2020-02-07

基金项目: 国家自然科学基金项目(91648119)、上海市科委项目(17DZ1205001)和上海市科技创新行动计划启明星项目(19QA1403500)

作者简介: 彭艳(1982—),女,教授,博士生导师,主要从事软体机器人、智能控制和仿生控制研究,E-mail: pengyan@shu.edu.cn

通信作者: 解杨敏(1984—),女,副教授,博士生导师,主要从事无人移动系统环境建模和运动控制研究,E-mail: xieym@shu.edu.cn

0 引言

路径规划^[1]是移动机器人在复杂环境中实现自主行动能力的关键技术之一。路径长度、实时性以及工作代价值等是评估路径规划算法的性能指标^[2],不同的应用场合对于路径规划算法有着不同的特性需求。

以对环境信息的认知范围区分^[3],路径规划算法可分为完整地图信息已知的全局路径规划和部分地图信息实时更新的局部路径规划。全局路径规划按照优化算法可以分为随机采样类算法、群体智能优化算法、图搜索类算法等^[4]。随机采样类算法主要包括快速扩展随机树(RRT)算法^[5-6]和概率路线图(PRM)算法^[7],在多维状态空间应用广泛,其搜索能力强,但该类算法运算成本高、随机性强^[8]。群体智能优化算法主要包括遗传算法^[9]、蚁群算法^[10-11]、粒子群算法^[12-13]等。遗传算法和蚁群算法适用于复杂问题的求解和优化,能同时处理多个个体,易于实现并行性,但是运算效率低;相比较而言,粒子群算法收敛速度快,但易陷入局部最优解。图搜索类算法主要有 Dijkstra 算法^[14]、A* 算法^[15]、D* 算法等。Dijkstra 算法是一种典型的最短路径算法,但是存在遍历节点多、效率低下的缺点;A* 算法增加了启发式搜索,通过设定启发函数评价路径优劣,倒序找到起点,相比 Dijkstra 算法减少了搜索量,在计算效率略有提升的同时保证了路径的最优性,但是在环境复杂、地图规划较大时,耗时仍然过长。尽管如此,A* 算法依然是目前广泛使用的路径规划算法^[16]。对于上述的全局优化算法,爬行虫(Bug)系列算法具有明显的实时计算优势。Bug 系列算法是完全应激式算法的典型代表^[17-18],也是局部路径规划中常见的算法,包括 Bug1、Bug2^[19]、Dist-Bug^[20]和 Rev-Bug^[21]等。Bug2、Dist-Bug、Rev-Bug 在应用细节上对 Bug1 进行了不同程度的改进。其中,表现较为突出的是 Dist-Bug 方法。虽然 Bug 系列算法在计算实时性能上有非常突出的表现,但由于其完全应激的基本特性,得到的路径长度明显较长。

为此,本文提出基于多爬虫机制的 Mutli-Bug 避障寻路算法。其改进思路是:相对于单爬虫的寻路策略,当遇到障碍物时爬虫进行分裂,同时沿不同的避障方向进行探索,并设置一定的爬虫死亡条件。此分裂死亡过程不断进行,直到最快的一只爬虫抵达终点,得到一条最优路径。这样在保留 Bug 算法简单寻路规则的同时也保证最终路径的相对最优性,因此,能够形成以部分路径长度为代价、降低运

算成本的全局路径规划算法。

1 Dist-Bug 算法简介

Bug 系列算法最初是在二维未知的复杂障碍环境下通过传感器实时传输环境信息进行实时路径规划的算法。其基本原理包括 2 种基本运动模式:模式 1,向终点直线移动;模式 2,障碍物绕行。通过两个模式互相转换,Bug 算法可实时调整其路径规划策略直到抵达终点。在直线移动模式下,爬虫由当前点沿着指向终点方向移动,直到抵达终点,或是遇到障碍物进行模式切换;在障碍物绕行模式下,爬虫绕行障碍物,并同时判断是否满足转换为模式 1 的条件。不同的 Bug 算法主要区别在于其障碍物绕行策略及模式切换策略不同。为了便于理解本文的 Multi-Bug 算法,详细描述作为其基础架构的具有代表性的 Dist-Bug 算法,算法框架如图 1 所示。

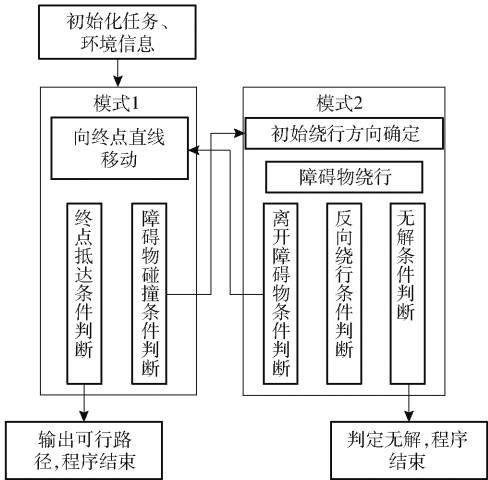


图 1 Dist-Bug 算法框架图
Fig. 1 Dist-Bug algorithm framework diagram

爬虫当前位置 x (视为 1 个质点),启用模式 1 时会发生 2 种情况(图 2):①直接到达终点 T ,路径规划算法停止。②碰到障碍物,将这一点表示为 H_i 点,并切换成模式 2。

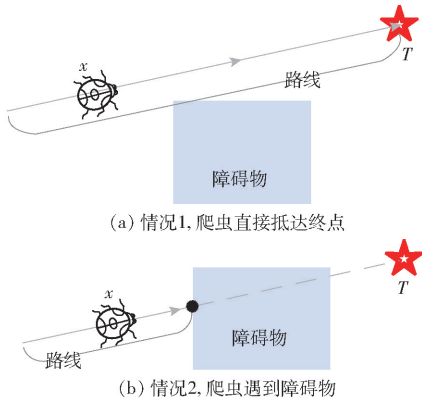


图 2 基于模式 1 的两种路径规划情况
Fig. 2 Two path planning cases based on mode 1

进入模式 2 后,爬虫会根据障碍物的边界和碰撞点 H_i (第 i 次从模式 1 转换到模式 2 时爬虫位置) 位置确定边界初始绕行方向(即在该碰撞点处选择绕行障碍物的方向),然后沿着障碍物边界移动。Dist-Bug 算法中边界绕行方向的选择策略如图 3 所示,分别比较顺、逆时针绕行方向与当前点指向终点方向对应的夹角 α_1 、 α_2 ,选择夹角小的绕行方向,当夹角 $\alpha_1 = \alpha_2$,则优先选择顺时针旋转方向。

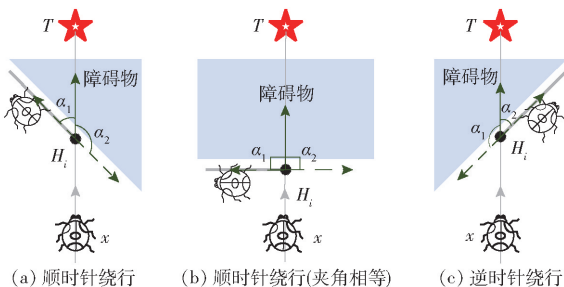


图 3 模式 2 中选择初始绕行方向策略

Fig. 3 Strategy for selecting initial detour direction in mode 2

在模式 2 的障碍物绕行过程中,需要持续判断是否切换到模式 1。Dist-Bug 算法的切换条件有 2 个,即

$$d(x, T) - F \leq 0 \quad (1)$$

$$d(x, T) - F \leq d_{\min}(T) - P \quad (P > 0) \quad (2)$$

式中 $d(x, T)$ ——机器人当前位置到终点距离
 F ——爬虫指向终点方向与第一个可见障碍物的距离

$d_{\min}(T)$ ——走过的路径点与终点最近的距离

P ——障碍物最小壁厚

满足其中任意一个条件,则切换到模式 1,否则继续绕行障碍物(图 4)。图 4a 中,机器人在点 L_i (第 i 次从模式 2 转换到模式 1 时爬虫位置)处, $d(x, T) = d(L_i, T)$, $F = d(L_i, T)$, 满足式(1);图 4b 中,机器人在点 L_i 处, $d(x, T) = d(L_i, T)$, $F = d(L_i, H_i)$, $d_{\min}(T) = d(L_i, T)$, 满足式(2)。

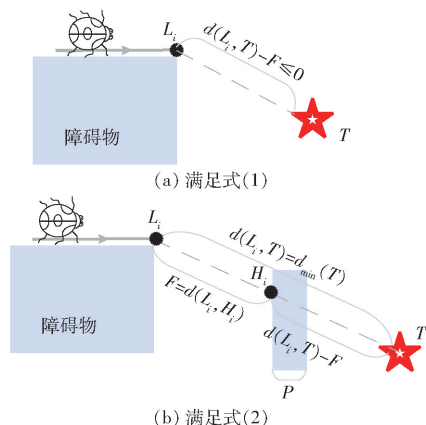


图 4 绕行过程中的切换条件情况

Fig. 4 Switching conditions during bypass

在 Dist-Bug 算法中,为避免初始绕行方向不合理还设置了反向绕行条件。在绕行过程中判断当前路径方向与指向障碍物方向夹角 α 是否大于 135° ,如果大于则进行最多一次的反向绕行操作,若沿障碍物绕行一周,抵达 H_i 或者 R_i (第 i 个碰撞点后机器人反向绕行的点),仍未满足模式切换条件,则判定终点不可达,结束路径规划。不可达情况判定如图 5 所示,爬虫由起点 S 出发,启用模式 1,至碰撞点 H_1 满足切换条件,启用模式 2,在 H_1 选择顺时针绕行至 R_1 点,此处当前路径方向与指向障碍物方向夹角 α 大于 135° ,且在碰撞点 H_1 后,没有执行过反向操作,满足反向条件,爬虫选择反向逆时针绕行,至 R'_1 点,由于执行过一次反向操作,所以此处不再反向,继续绕行至反向点 R_1 点,判断终点不可达,结束路径规划。

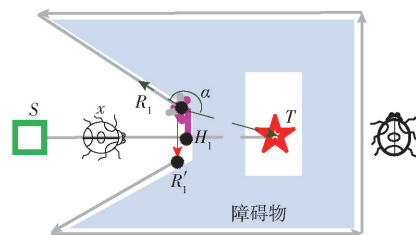


图 5 没有可行路径

Fig. 5 No feasible path

以上 Dist-Bug 的伪代码算法表示:

(1) 建立栅格地图。

(2) 初始化 $i = 0$, 绕行标志 $G = 1$ 并设置 P 为壁厚。

While (1)

If ($G = 1$) 进入模式 1

递增 i 并向终点移动,直到出现以下情况之一:

① 抵达终点,停止,退出 While 循环。② 遇到障碍物,将这一点坐标表示为 H_i , G 赋值为 2。

End

If ($G = 2$) 进入模式 2

判断转向并跟随障碍物边界,同时连续更新最小值 $d_{\min}(T)$ 。直到出现以下情况之一:

① 终点可见,即满足 $d(x, T) - F \leq 0$ 。将爬虫当前点表示为 L_i , G 赋值为 1。② 如果离开绕行障碍物条件成立,即满足 $d(x, T) - F \leq d_{\min}(T) - P$,将爬虫当前点表示为 L_i , G 赋值为 1。③ 判断当前路径方向与指向障碍物方向夹角是否大于 135° 。如果大于 135° ,满足反向第 1 个条件,再判断是否已经有反向点 R_i 。无,满足反向第 2 个条件,取反向处点为 R_i ,反向;有反向点 R_i ,则不再反向。④ 判断 R_i 是否有值。无,机器人完成了—个循环并达到了 H_i ,终点无法访问,停止,退出 While 循环;有,机器

人完成了一个循环并达到了 R_i , 终点无法访问, 停止, 退出 While 循环。

End
End

Dist-Bug 算法能够在未知环境信息的情况下, 根据自身位置信息、任务信息以及实时的传感器测量信息, 实时高效地调整自身姿态, 选择运动模式, 规划出到达终点的路径。但正因其算法设计基于有限的局部环境信息, 当将其应用于全局路径规划时, 其决策缺乏对路径长度优化策略的考虑, 通常会给出远超过最优路径长度的规划结果。

2 Multi-Bug 全局路径规划

2.1 算法介绍

Multi-Bug 算法属于全局路径规划, 环境信息均为已知, 使用 2 种同样的基本运动模式。模式 1, 向终点直线移动; 模式 2, 障碍物绕行。爬虫遇到障碍物时, 爬虫分裂生成 2 条路径, 由模式 1 切换至模式 2, 分别顺逆时针绕行障碍物各自前行; 如此循环, 直到其中一个爬虫率先抵达终点。Multi-Bug 算法框架图如图 6 所示。

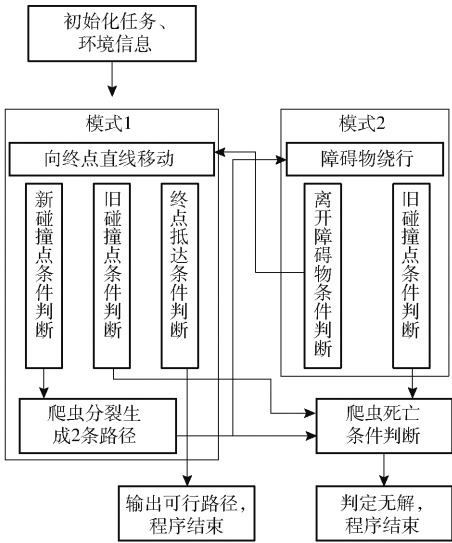


图 6 Multi-Bug 算法框架图
Fig. 6 Multi-Bug algorithm framework diagram

启用模式 1 时, 如图 7 所示。在此模式下需要持续判断是否发生以下 3 种情况: ①遇到新的碰撞点 H_i 点, 爬虫生成顺、逆时针两个绕行方向的路径, 如图 7a 所示, 并切换成模式 2。②遇到以前的碰撞点, 舍弃当前这条路径, 如图 7b 所示, 路径 L'_i 绕行一圈回到碰撞点 H_i , 该条路径舍弃。③满足抵达终点条件, 输出可行路径, 路径规划算法结束。

进入模式 2 以后, 需要持续判断是否发生以下 3 种情况: ①满足条件 $d(x, T) - F \leq 0$, 直接达到终点, 路径规划算法成功停止。②满足条件 $d(x, T) -$



图 7 基于模式 1 的第 1, 2 种情况
Fig. 7 Case 1 and case 2 based on mode 1

$F \leq d_{\min}(T) - P$, 切换成模式 1 离开障碍物, 这里与 Dist-Bug 算法相同。③遇到以前的碰撞点, 舍弃当前这条路径, 如图 8 所示。爬虫遇到碰撞点 H_i 后, 生成 L'_i 和 L''_i 2 条绕行路径, 随后路径 L'_i 遇到新的碰撞点 H_{i+1} , 生成 L'_{i+1} 和 L''_{i+1} 2 条路径, 此时共有 3 条同时循迹的路径 L'_i 、 L'_{i+1} 和 L''_{i+1} 。路径 L'_{i+1} 绕行时遇到以前的碰撞点 H_i , 舍弃。

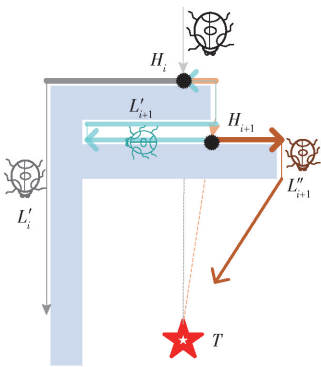


图 8 模式 2 中遇到以前碰撞点的情况
Fig. 8 Previous collision points encountered based on mode 2

当所有路径都停止循迹且未找到终点时, 说明终点不可达, 停止循迹, 如图 9 所示。爬虫由起点走到 H_1 点后, 分别顺逆时针方向绕行障碍物, 直到顺时针绕行路线 (红色) 和逆时针绕行路线 (紫色) 分别碰到以前的碰撞点 H_1 。说明没有可行解。

Multi-Bug 算法中基本操作为移动栅格, 时间复杂度为 $O(n)$, 各情况判断条件时间复杂度也为

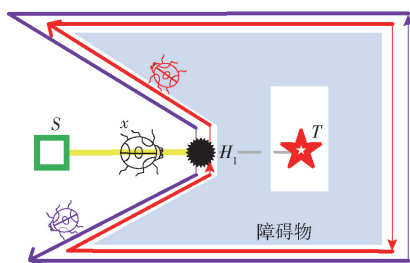


图9 没有可行路径的情况

Fig. 9 No feasible path

$O(n)$; 设操作中同时存在的最大爬虫个数为常数 M , 算法总时间复杂度将满足条件

$$T(n) \leq CMn \quad (3)$$

式中 C ——常数系数 n ——问题规模

因此 Multi-Bug 算法时间复杂度为 $O(n)$, 这一点将在之后的仿真中予以验证。

Multi-Bug 算法的伪代码表示为:

(1) 建立栅格地图。

(2) 输入起点 S 、终点 T 以及障碍物信息, 初始化一条爬虫路径, P 取 3, 以及绕行标志 $G=0$, 即默认启用模式 1。

While (1)

 计算出现有路径个数 M ;

 If ($M = 0$)

 输出无解; 退出 While 循环;

 End

 For ($i=0; i \leq M; i++$)

 If ($G = 0$) 进入模式 1

 向终点方向移动一个栅格; 更新 $d_{\min}(T)$;

 ①判断该点是否为新的碰撞点。是, 爬虫在新的碰撞点处分裂出 2 条路径, 更新路径 G 值; 否, 继续。②判断该点是否为以前的碰撞点 H_i 。是, 退出 For 循环, 舍弃该条路径; 否, 继续。③判断该点是否为终点; 是, 退出 While 循环, 输出路径; 否, 继续 For 循环。

 Else 进入模式 2

 绕行障碍物一个栅格; 更新 $d_{\min}(T)$ 。①判断该点是否满足离开障碍物条件 $d(x, T) - F \leq d_{\min}(T) - P$ 。满足, 更新路径 G 值; 不满足, 继续。②判断该点是否为以前的碰撞点 H_i 。是, 退出 For 循环, 舍弃该条路径; 否, 继续 For 循环。

 End

End

End

Multi-Bug 算法在已知环境信息的情况下, 根据环境信息以及任务信息, 选择运动模式, 生成多条路径, 实时高效规划出到达终点的路径。因 Multi-Bug

算法基于局部路径规划算法 Dist-Bug 设计, 不进行大范围搜索, 所以具有高时效性的特点; Multi-Bug 算法有着全局环境信息, 相当于多爬虫同时进行路径探索并且输出路径为所有爬虫可行解中最优, 路径长度能够实现局部最优。

2.2 算法证明分析

假定在已知的有限面积环境信息下, 起点 S 到终点 T 的直线距离为 E , 中间随机分布着个数为 K 的有限个不规则多边形的障碍物, 其中障碍物的边界周长最大值为 Q 。

(1) 路径有限性: 在有限的路径长度下停止 Multi-Bug 算法。

证明: Multi-Bug 算法生成的路径包括向终点直线运动模式下生成的路径和障碍物边界跟随模式下生成的路径 2 部分, 证明路径有限性只需证明在 2 种模式下算法路径长度有限, 且模式 1、2 切换次数有限。

模式 1 下每一段直线路径长度小于 E , 因为每一次由模式 2 切换到模式 1 均满足条件式(2)或是条件式(1)。模式 2 生成的每一段路径长度不会超过障碍物的周长 Q , 因此 2 种模式下算法路径长度均有限。

模式 2 切换到模式 1 条件式(1)在输出路径中只发生一次, 因为机器人在此条件成立后, 直接到达目标; 条件式(2)可以发生 K 次, $K \leq E/P^{[20]}$ 。机器人在第 $K+1$ 次遇到障碍物时, 有 2 种可能: 直接抵达终点或者路径形成环路, Multi-Bug 算法停止。因此 2 种模式切换次数有限, 定理得证。

(2) 有解可达性: 只要从起点到终点有可行路径, Multi-Bug 算法一定可以到达终点。

证明: 证明路径有解可达, 先证明舍弃的路径不会造成可行解的丢失, 接着证明有解时, 不会提前退出算法, 导致可行解无法输出。

路径舍弃条件 1: 遇到以前的碰撞点, 舍去, 因为继续绕行下去会与在该碰撞点处另一绕行方向的路径重复; 路径舍弃条件 2: 有可行解抵达路径。这两种均不会造成可行解丢失。

算法结束条件 1, 有爬虫路径抵达终点, 输出可行解。结束条件 2, 判断所有爬虫路径均无效舍弃, 此时无可行路径。因此结束条件不会导致有可行解时提前结束算法, 一定会有输出路径, 或是判定无解, 定理得证。

(3) 路径局域最优性: 利用 Multi-Bug 算法得到的最后输出路径, 一定是诸多可行路径中最短的。

证明: 证明路径最优性, 先证明得到的路径不会有重复路径出现, 接着证明输出路径是所有可行路

径中最短的。

如果其中一条路径遇到以前的碰撞点,该路径直接删除,因为在该碰撞点,顺、逆时针 2 个绕行方向已经经历过,从而保证了每一条路径里都不会有重复的相同路径段出现。

如果有可行路径已经达到终点,会直接输出可行路径,并且停止更新其他路径,不会出现重复输出第 2 条更长路径的现象。因此利用 Multi-Bug 算法最后的输出路径,一定是其诸多可行路径中最短的,定理得证。

3 算法仿真验证

Multi-Bug 算法是一种全局路径规划算法,该算法相比于搜索类路径规划算法是以有限的路径长度换取运算效率的很大提升。为了进一步验证 Multi-Bug 算法相较于传统路径规划算法在路径长度及运算时间成本方面的性能优劣,本文分别复现了代表性的路径规划方案 Dist-Bug 算法、A* 算法和 RRT* 算法,通过对比算法输出路径长度和运算时间来比较算法性能。运行算法计算机配置为 Xeon(R) E3-1230 v2

@ 3.30 GHz * 8 CPU; ubuntu 16.04 系统; Matlab R2017b 软件。为保证运算结果可靠性,对同样的栅格地图中每个算法运行 10 次,其运算时间取平均值。为了验证算法的通用性,本文测试了各算法在包括障碍物规则类图形、障碍物不规则类图形以及迷宫类图形的多类栅格地图下的路径规划性能。

在本文复现的算法中,RRT* 算法性能依赖于其参数设置。随机生成的点越多,得到可行路径的概率越大,但是运算时间也越长;搜索半径越大,得到的更短路径长度概率越大,运算时间越长;搜索步长通常小于搜索半径。综合考虑路径长度和运算时间,本文中选取参数为随机生成点共 2 000 个,搜索半径取 15(栅格数,下同),搜索步长 5。因为 RRT* 算法的随机性,导致其每次得到的路径长度不同,时间不同,这里给出的时间和长度为 10 次仿真的平均值,给出的仿真图选择路径长度最接近平均值的一组图。

各算法在多类地图中的路径规划结果数据对比见表 1。

表 1 算法运算时间和路径长度汇总
Tab. 1 Summary of algorithm operation time and path length

场景	Multi-Bug 算法		Dist-Bug 算法		A* 算法		RRT* 算法	
	运算时间/s	路径长度 (栅格数)	运算时间/s	路径长度 (栅格数)	运算时间/s	路径长度 (栅格数)	运算时间/s	路径长度 (栅格数)
1(图 10)	0.290 2	96	0.252 1	105	2.769 6	92	6.027 5	127.4
2(图 12)	0.231 5	58	0.217 9	60	1.130 0	45	1.081 0	59.2
3(图 13)	0.243 9	103	0.325 5	255	0.591 2	85	3.482 2	96.7
4(图 14)	0.276 0	79	0.272 3	109	2.019 3	66	11.466 8	70.6
5(图 15)	0.263 6	67	0.330 4	314	2.223 7	57	64.990 8	
6(图 16)	0.337 8		0.462 5		3.460 1		63.804 0	
时间平均值	0.273 8		0.310 1		2.032 3		25.142 1	
时间标准差	0.034 5		0.078 7		0.958 3		27.937 8	
已有路径长度平均值		80.6		168.6		69.0		88.5

3.1 多类地图路径规划性能比较

3.1.1 凸多边形障碍物地图

在如图 10 所示的规则凸多边形障碍物地图中,均匀密集地分布着正方形障碍物。各算法具体运算时间、路径长度见表 1。从路径长度上分析,Multi-Bug、Dist-Bug 和 A* 算法得到路径长度结果基本相似,与路径最短的 A* 算法相比路径长度分别增加了 4.3%、14.1%;RRT* 得到的路径最长,超过 A* 近 40%。从运算时间上分析,Multi-Bug 算法与 Dist-Bug 算法用时均小于 0.3 s,比 A* 算法用时减小了约 90%。Dist-Bug 算法时间最短,但在中间路径判断绕行方向时(见图 10b 红框内路径),出现了遇顶点多解情况。如图 11 所示,在碰撞点 H_i 处,最

佳绕行方向应该逆时针,图中顺时针绕行;在碰撞点 H_{i+1} 处,最佳绕行方向为顺时针,图中逆时针绕行。这两处问题的原因相同,因为碰撞点恰巧在障碍物顶点处,导致确定碰撞点 H_i 遇到障碍物的边界时出现问题。这一绕行算法的局限在文献[20]中并没有提及。

3.1.2 凹多边形障碍物地图

在图 12 存在凹多边形障碍物的地图中,随机分布形状各异的障碍物,Multi-Bug 算法、Dist-Bug 算法、A* 算法和 RRT* 算法具体规划路径长度、运算时间见表 1。从路径长度上分析,A* 得到的路径长度最短,而 Multi-Bug 算法和 Dist-Bug 算法、RRT* 算法得到路径长度基本相同,相差小于 3%;从时间上

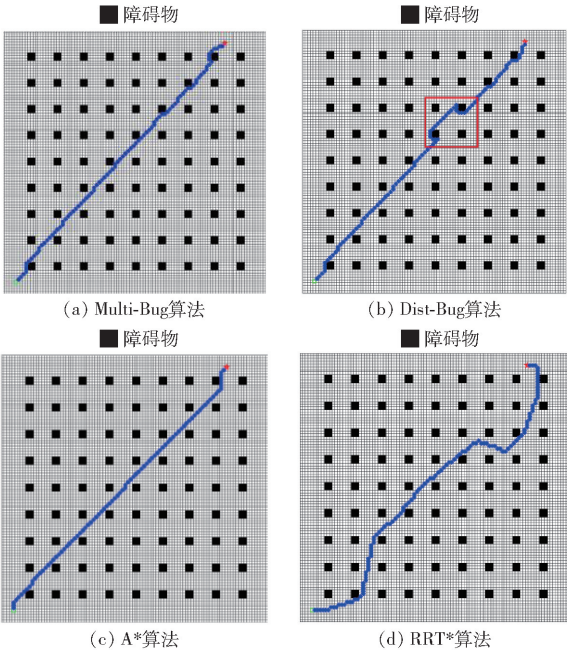


图 10 凸多边形障碍物地图中的各算法路径规划结果比较(100 × 100)

Fig. 10 Comparison of path planning results in convex polygon obstacle maps (100 × 100)

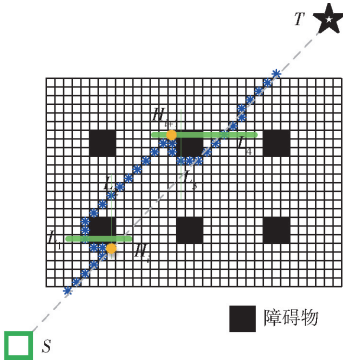


图 11 Dist-Bug 算法初始方向判断问题
Fig. 11 Problem of initial direction judgment of Dist-Bug algorithm

分析,A*算法与 RRT* 算法运算时间均长于 1 s,近乎于 Multi-Bug 算法和 Dsit-Bug 算法运算时间的 5 倍。

3.1.3 迷宫类地图

图 13 ~ 15 所示的迷宫类地图(编号为地图 1 ~ 3)被认为是路径规划问题中较难解决的一类复杂地形,在这类地形中的路径规划通常需要更长的计算时间,也更考验算法的综合性能,4 种算法具体运算时间和路径规划长度见表 1。

在迷宫类地图 1、2 中,4 种算法均得到可行路径。A*得到的路径长度最短,平均值为 76,RRT*算法平均路径长度为 84,Multi-Bug 算法平均路径长度为 91,这 3 种算法路径相差小于 20%,而 Dist-Bug 算法平均路径最长为 182,是前 3 者的 2 倍多。时间上 Multi-Bug 算法平均用时最短,为 0.26 s,A*算法平均用时 1.31 s,5 倍于 Multi-Bug 算法,RRT*算

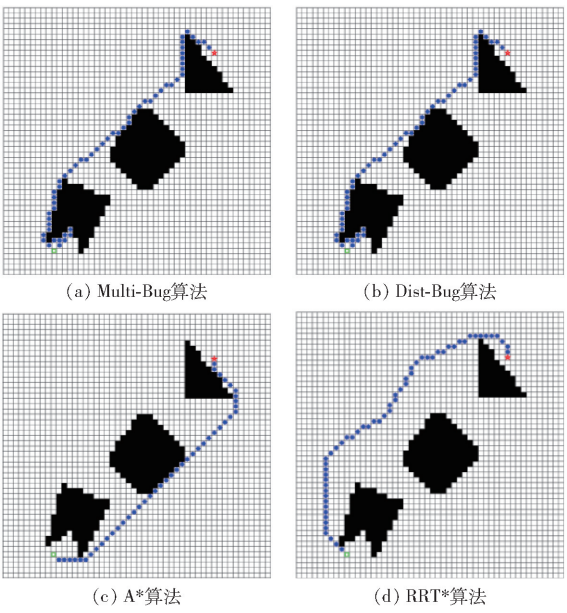


图 12 凹多边形障碍物地图中的各算法路径规划结果比较(50 × 50)

Fig. 12 Comparison of path planning results in concave polygon obstacle maps (50 × 50)

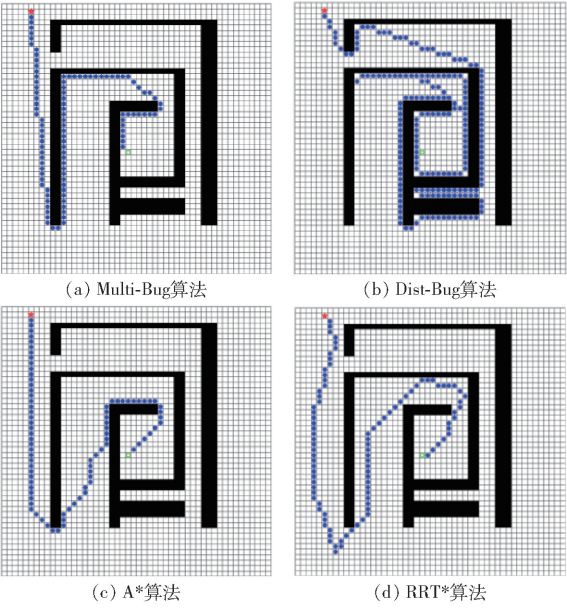


图 13 迷宫类地图 1 中的各算法路径规划结果比较(50 × 50)

Fig. 13 Comparison of path planning results in maze map 1 (50 × 50)

法平均时间依然是最长,为 7.47 s。

在迷宫类地图 3 中,只有 3 种算法得到可行路径,A*算法路径最短,Multi-Bug 算法路径长度与 A*算法相比增加了 17.5%,Dist-Bug 算法路径最长,是 A*算法的 5 倍多,RRT*算法无解。时间上依然是 Multi-Bug 算法最短,为 0.26 s,A*算法耗时达到 Multi-Bug 算法的 8 倍多,在无解的情况下 RRT*算法运算时间超过 1 min,因为完整地产生了 2 000 次随机数。

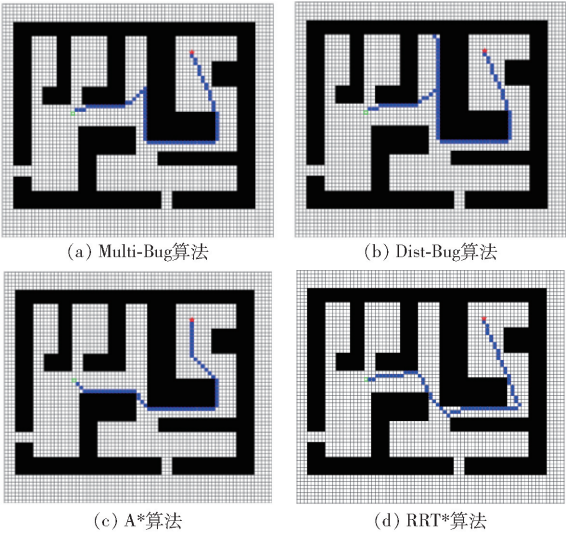


图 14 迷宫类地图 2 中的各算法路径规划结果比较 (65 × 75)

Fig. 14 Comparison of path planning results in maze map 2 (65 × 75)

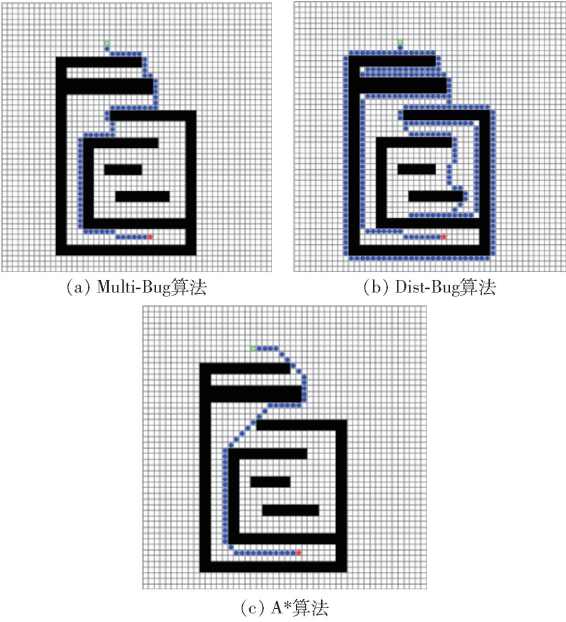


图 15 迷宫类地图 3 中的各算法路径规划结果比较 (50 × 50)

Fig. 15 Comparison of path planning results in maze map 3 (50 × 50)

3.1.4 无可行路径类地图

在图 16 无可行路径类地图中,测试在此极端情况下各算法运算性能,Multi-Bug 算法得出无解结论的时间最短(0.34 s)、不到 A* 算法的 1/10,表现最为优异。

3.2 Multi-Bug 与 A* 算法性能比较分析

综合以上所有类型地图的路径规划性能对比,结合表 1 数据分析可知,本文 Multi-Bug 算法在 3.1.1 ~ 3.1.3 节这 3 类地图中生成的平均路径长度为 80.6,仅次于 A* 算法的 69.0,具体如图 17 所示。与 A* 算法相比,Multi-Bug 算法给出的平均路

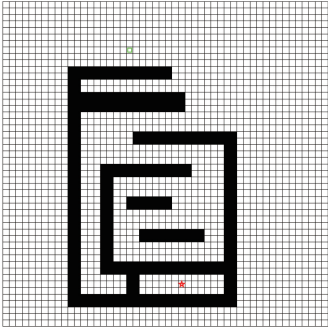


图 16 迷宫类地图 4,无可行路径 (50 × 50)

Fig. 16 Maze map 4, no feasible path (50 × 50)

径长度仅增加了 16.8%,平均用时减少了 86.5%,仅为 0.27 s,在 4 种算法中用时最短,达到了在尽可能不增加路径长度的同时大大提升计算效率的目的。Multi-Bug 算法时间标准差也是 4 种算法中最小的,只有 0.034 5 s,足以说明 Multi-Bug 算法具有更好的稳定性。Multi-Bug 算法在每一类场景地图中都能高效得出可行解,适应性同样可以得到保证。

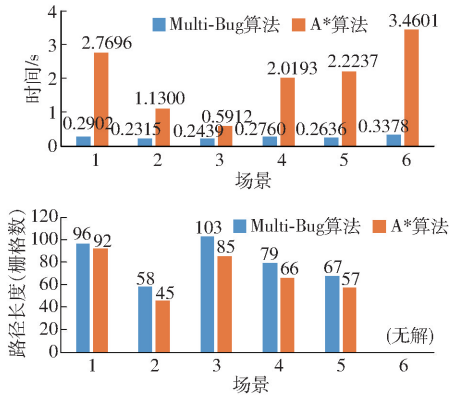


图 17 Multi-Bug 算法和 A* 算法对比

Fig. 17 Multi-Bug algorithm and A* algorithm path length histograms

Multi-Bug 算法参考了局部路径规划算法 Dist-Bug 算法的框架,遵循简单寻路规则而非进行泛搜索,如图 18 所示,Multi-Bug 算法时间在 500 × 500 的栅格地图中接近于 A* 算法时间的万分之一。A* 算法时间复杂度为 $O(n^2)$,而 Multi-Bug 算法时间复杂度只有 $O(n)$,这是其计算效率高的根本原因。而因其同一时间并行更新多条路径并取其最优,所以得到的路径相

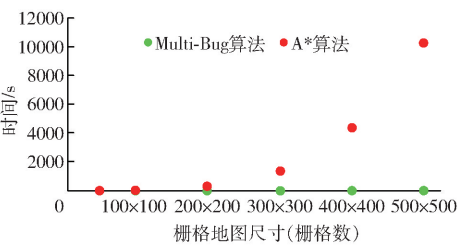


图 18 Multi-Bug 算法与 A* 算法时间趋向对比

Fig. 18 Comparison of time trend between Multi-Bug algorithm and A* algorithm

比传统 Bug 算法更优,且其不同复杂度地图中用时变化量很小,具有更好的计算性能稳定性。

3.3 实际应用案例

图 19 为引用文献[22]中的图像,图 19a 是美国亚拉巴马州拉塞尔县本宁堡住户实景拍摄图像,图 19b 为本宁堡住户整体地图数据,路径任务是从一户人家导航至另一户人家。将图 19a 连续空间缩减为一个离散图,分布着若干形状各异的图形,分别代表着不同的住户的房屋,栅格化处理后,分别利用 Multi-Bug、Dist-Bug、A* 和 RRT* 算法进行路径规划得到如图 20 所示结果。

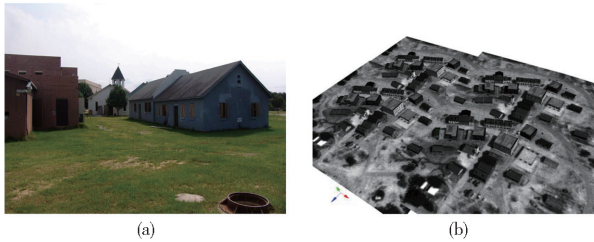


图 19 从本宁堡获得的用于测试的数据
Fig. 19 Data acquired from Fort Benning and used for testing

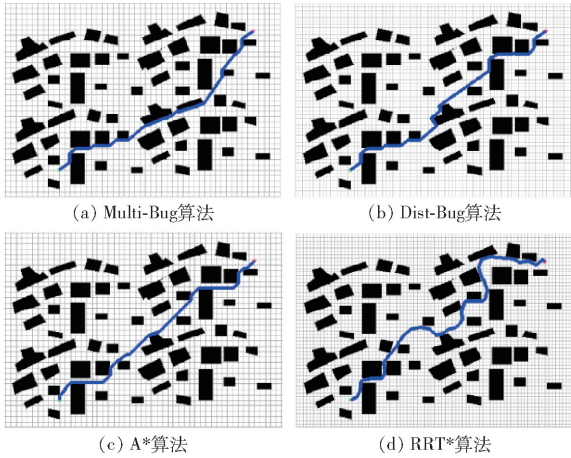


图 20 实际应用类地图 (280 × 400)

由图 20 可以看出,Multi-Bug 和 A* 算法得到路径长度结果基本相近,分别为 337 和 320;Dist-Bug 算法路径较长,为 360;RRT* 算法得到的路径最长,超过 A* 算法 29.3%。从运算时间上分析,Multi-Bug 与 Dist-Bug 算法用时均小于 1 s,分别为 0.859 0、0.500 7 s。A* 算法超过 10 min,为 678.608 0 s。Multi-Bug 与 A* 算法相比路径长度增加了 5.3%,但是用时减少了 99.9%。

3.4 Multi-Bug 失效情况讨论

在以往 Bug 系列算法文献中没有提及但值得注

意的是:当应用于离散栅格地图时,Bug 系列算法在某种极端情况下会存在找不到可行路径的情况,这一特点也被 Multi-Bug 算法继承。当地图中存在单栅格通道时,如图 21 所示,因通道栅格与多个障碍物边界接触,算法绕行策略可能失效。图 21a 与图 21b 唯一不同之处为终点不同,导致图 21a 有解,图 21b 无解。图 21b 中因为碰撞点 H_3 正好在唯一可行路径 L_2 行下面,当路径经过碰撞点 H_4 绕行障碍物遇到以前的碰撞点 H_3 时舍去,因而错过唯一可行路径。这一问题通常不影响 Bug 算法在实际地图中的应用,一个简单的解决方案为将离散栅格尺寸设为最小通道宽度的 1/2 以下。而在实际应用中当地图中存在着极窄通道时,通常会判定机器人通行安全性较低,从而舍弃其作为可行路径的可能。

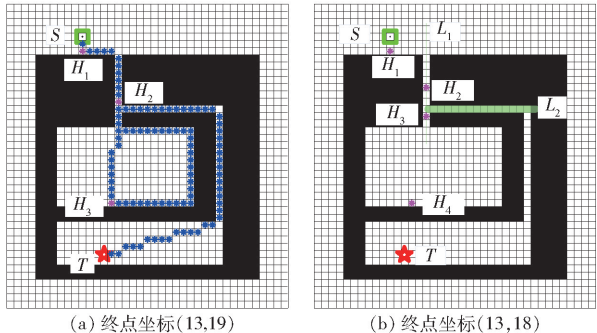


图 21 Multi-Bug 在离散栅格中的错误无解问题

Fig. 21 Missing feasible solution by Multi-Bug algorithm in discrete grids

4 结束语

经过理论分析及仿真验证,提出的 Multi-Bug 算法充分利用了 Bug 系列算法的非搜索特性,同时引入多虫并行寻路策略,以优选最短路径。相较于传统的搜索类算法,其路径长度增加有限,而计算复杂度只有 $O(n)$ 。相对于一般的最优化算法和图搜索算法,具有更强的目标导引性、更少的随机性和优化算法依赖性。仿真结果显示,在 3 类有可行路径的复杂地图下,该算法均能在稳定短时间内给出可行的规划路径,即使在无可行路径的情况下,也会在短时间内给出无解的结论,满足机器人全局路径规划算法对路径较优、时效性强、算法稳定性好的性能需求。因此,在处理大型复杂地图或者强调实时计算的场景中,Multi-Bug 算法具有较强的应用优势。本文 Multi-Bug 算法处理的是全局静态环境下的路径规划问题,对于其在动态空间或者高维地形下的应用拓展还有待进一步研究。

参 考 文 献

[1] 朱大奇,颜明重. 移动机器人路径规划技术综述[J]. 控制与决策,2010,25(7): 961 – 967.

- ZHU Daqi, YAN Mingzhong. Survey on technology of mobile robot path planning[J]. Control and Decision, 2010, 25(7): 961–967. (in Chinese)
- [2] 霍凤财, 迟金, 黄梓健, 等. 移动机器人路径规划算法综述[J]. 吉林大学学报(信息科学版), 2018, 36(6): 46–54.
HUO Fengcai, CHI Jin, HUANG Zijian, et al. Review of path planning for mobile robots[J]. Journal of Jilin University (Information Science Edition), 2018, 36(6): 46–54. (in Chinese)
- [3] 张广林, 胡小梅, 柴剑飞, 等. 路径规划算法及其应用综述[J]. 现代机械, 2011(5): 85–90.
ZHANG Guanglin, HU Xiaomei, CHAI Jianfei, et al. Summary of path planning algorithm and its application[J]. Modern Machinery, 2011(5): 85–90. (in Chinese)
- [4] 王春颖, 刘平, 秦洪政. 移动机器人的智能路径规划算法综述[J]. 传感器与微系统, 2018, 37(8): 5–8.
WANG Chunying, LIU Ping, QIN Hongzheng. Review on intelligent path planning algorithm of mobile robots[J]. Transducer and Microsystem Technologies, 2018, 37(8): 5–8. (in Chinese)
- [5] KARAMAN S, FRAZZOLI E. Incremental sampling-based algorithms for optimal motion planning[J]. The International Journal of Robotics Research, 2010, 30(7): 846–894.
- [6] 李洋, 徐达, 周诚. 基于自适应步长 RRT 的双机器人协同路径规划[J/OL]. 农业机械学报, 2019, 50(3): 358–367.
LI Yang, XU Da, ZHOU Cheng. Cooperation path planning of dual-robot based on self-adaptive stepsize RRT[J/OL]. Transactions of the Chinese Society for Agricultural Machinery, 2019, 50(3): 358–367. http://www.jcsam.org/jcsam/ch/reader/view_abstract.aspx?flag=1&file_no=20190341&journal_id=jcsam. DOI:10.6041/j.issn.1000-1298.2019.03.041. (in Chinese)
- [7] YAN F, LIU Y S, XIAO J Z. Path planning in complex 3D environments using a probabilistic roadmap method[J]. International Journal of Automation and Computing, 2013, 10(6): 525–533.
- [8] 王全. 基于 RRT 的全局路径规划方法及其应用研究[D]. 长沙: 国防科学技术大学, 2014.
WANG Quan. Research on rapidly-exploring random trees based global path planning and its application[D]. Changsha: National University of Defense Technology, 2014. (in Chinese)
- [9] LEE J, KIM D W. An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph[J]. Information Sciences, 2016, 332: 1–18.
- [10] 史恩秀, 陈敏敏, 李俊, 等. 基于蚁群算法的移动机器人全局路径规划方法研究[J/OL]. 农业机械学报, 2014, 45(6): 53–57.
SHI Enxiu, CHEN Minmin, LI Jun, et al. Research on method of global path-planning for mobile robot based on ant-colony algorithm[J/OL]. Transactions of the Chinese Society for Agricultural Machinery, 2014, 45(6): 53–57. http://www.j-csam.org/jcsam/ch/reader/view_abstract.aspx?flag=1&file_no=20140609&journal_id=jcsam. DOI:10.6041/j.issn.1000-1298.2014.06.009. (in Chinese)
- [11] 刘建华, 杨建国, 刘华平, 等. 基于势场蚁群算法的移动机器人全局路径规划方法[J/OL]. 农业机械学报, 2015, 46(9): 18–27.
LIU Jianhua, YANG Jianguo, LIU Huaping, et al. Robot global path planning based on ant colony optimization with artificial potential field[J/OL]. Transactions of the Chinese Society for Agricultural Machinery, 2015, 46(9): 18–27. http://www.jcsam.org/jcsam/ch/reader/view_abstract.aspx?flag=1&file_no=20150903&journal_id=jcsam. DOI:10.6041/j.issn.1000-1298.2015.09.003. (in Chinese)
- [12] MILICA P, NAJDAN V, MARKO M, et al. Integration of process planning and scheduling using chaotic particle swarm optimization algorithm[J]. Expert Systems with Applications, 2016, 64: 569–588.
- [13] 贾会群, 魏仲慧, 何昕, 等. 基于改进粒子群算法的路径规划[J/OL]. 农业机械学报, 2018, 49(12): 371–377.
JIA Huiqun, WEI Zhonghui, HE Xin, et al. Path planning based on improved particle swarm optimization algorithm[J/OL]. Transactions of the Chinese Society for Agricultural Machinery, 2018, 49(12): 371–377. http://www.j-csam.org/jcsam/ch/reader/view_abstract.aspx?flag=1&file_no=20181244&journal_id=jcsam. DOI:10.6041/j.issn.1000-1298.2018.12.044. (in Chinese)
- [14] AMMAR A, BENNACEUR H, CHAARI I, et al. Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments[J]. Soft Computing, 2016, 20(10): 4149–4171.
- [15] 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.
- [16] 顾辰. 改进的 A* 算法在机器人路径规划中的应用[J]. 电子设计工程, 2014(19): 96–98.
GU Chen. Application of improved A* algorithm in robot path planning[J]. International Electronic Elements, 2014(19): 96–98. (in Chinese)
- [17] 李一波, 张庆涛. 室内未知环境遍历路径规划算法综述[J]. 计算机科学, 2012, 39(增刊3): 334–338.
LI Yibo, ZHANG Qingtao. Review of coverage path planning arithmetic in unknown indoor environment[J]. Computer Science, 2012, 39(Supp.3): 334–338. (in Chinese)
- [18] MCGUIRE K N, DE C G, TUYLS K. A comparative study of bug algorithms for robot navigation[J]. Robotics and Autonomous Systems, 2019, 121: 103261.
- [19] NG J S. An analysis of mobile robot navigation algorithms in unknown environments[D]. Perth: University of Western Australia, 2010.
- [20] KAMON I, RIVLIN E. Sensory-based motion planning with global proofs[J]. IEEE Transactions on Robotics and Automation, 1997, 13(6): 814–822.
- [21] HORIUCHI Y, NOBORIO H. Evaluation of path length made in sensor-based path-planning with the alternative following[C]// Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation. USA: IEEE, 2001: 1728–1735.
- [22] VAN D B J, FERGUSON D, KUFFNER J. Anytime path planning and replanning in dynamic environments[C]// Proceedings 2006 IEEE International Conference on Robotics and Automation, USA: IEEE, 2006: 2366–2371.