不确定环境下基于改进萤火虫算法的地面自主车辆全局路径规划方法

杜鹏桢,唐振民,陆建峰,孙 研
(南京理工大学计算机科学与工程学院,江苏南京 210094)

摘 要: 针对地面自主车辆的特点,提出了一种基于改进萤火虫算法(Glowworm Swarm Optimization,GSO)的路径规划方法.首先利用GSO覆盖多个局部最优解的能力,一次生成多条规划路径;然后提出两种路径切换算法,分别用于调优和脱困.在通过路径交叉点时,调优切换算法对交叉路径进行重新评估并切换到较优路径,最终达到实际行驶路径的最优化.在遇到环境发生改变时,脱困切换算法通过启发式搜索快速切换到适当路径,重用了原搜索结果,避免了二次规划.通过大量的仿真实验及实际试用,证明了所提方法的可行性和有效性.

关键词: 路径规划; 地面自主车辆; 人工萤火虫算法; 二次规划; 路径切换
中图分类号: TP. 242 **文献标识码:** A **文章编号:** 0372-2112 (2014)03-0616-09
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.03.031

Global Path Planning for ALV Based on Improved Glowworm Swarm Optimization Under Uncertain Environment

DU Peng-zhen, TANG Zhen-min, LU Jian-feng, SUN Yan
(*College of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, Jiangsu* 210094, *China*)

Abstract: According to the characteristics of autonomous land vehicle, a global path planning method based on improved glowworm swarm optimization (GSO) is proposed. Firstly, more than one path is generated with GSO which covers multiple local optima. Then two path switching algorithms are proposed, of which one aims at optimization and the other aims at rescue. When the cross point is passed through, the optimization switching algorithm revaluates the paths, switches to the optimum path, and ultimately attains optimal actual travel route. When the environment changes, the rescue switching algorithm switches to the appropriate path by heuristic search, which reuses the original search results, avoiding the secondary planning. Many simulation experiments and actual trial show that the proposed method is feasible and effective.

Key words: path planning; autonomous land vehicle (ALV); glowworm swarm optimization (GSO); secondary planning; path switching

1 引言

地面自主车辆(Autonomous Land Vehicle,ALV)是一种非常复杂的智能机器人系统,在军事领域和民用领域都有重要的应用价值.一个典型的ALV系统,包括定位感知层、融合理解层、路径规划层和运动控制层等,其中路径规划是ALV系统智能性的集中体现,规划的好坏直接影响任务的成败.经过40多年的研究发展,ALV的任务范围越来越大,由几百米到几公里,由几公里到十几公里,甚至从一个城市到另一个城市.随着任务范围的扩大,难以避免的会出现全局信息无法全部预知的情况,原有静态确定环境下的全局路径规划算法不再适用.目前,动态不确定环境下的路径规划是智能机器人领域的研究热点之一,各国学者提出了很多针对性的方法和模型,常见的有:人工势场法[1~3]、神经网络[4,5]、D*类算法[6~8]、快速随机搜索树[9,10]、遗传算法[11,12]、蚁群算法[13~16]和人工鱼群算法[17]等.其中部分是对静态路径规划算法的改进和拓展,部分是算法

收稿日期:2013-08-14;修回日期:2013-10-15;责任编辑:覃怀银
基金项目:国家自然科学基金(No.91220301,No.61371040);高等学校学科创新引智计划资助课题(No.B13022)

在不确定环境下的新应用. 文献[1~3]通过对人工势场法的拓展,使之可以进行动态环境下的路径规划,但人工势场法本身的一些问题并没有解决,比如无法到达终点、存在局部最优等. 文献[4]和文献[5]将神经网络应用于动态路径规划,有较好的学习能力,但在大规模不确定环境下网络结构过于庞大. 文献[6~8]为 D^* 及其改进算法在动态路径规划方面的应用,该类算法的核心理念是遇到环境改变时进行重规划,有效的利用了中间计算结果,但对离机器人较远的环境改变处理效果不理想. 文献[9]和文献[10]所用快速随机搜索树算法是相对较新的一种方法,最大的优点是在高维姿态空间中可以较为快速的完成规划,但是,由于其本身随机采样的特性,导致规划结果不唯一,且在低维姿态空间中较果不佳. 文献[11~16]分别用遗传算法和蚁群算法进行动态环境下的路径规划,取得了不错的效果,但都在不同程度上存在实时性较差的问题. 文献[17]用人工鱼群算法进行动态环境下的路径规划并提出了避碰规则库,机器人在预先生成的静态最优路径上行进,同时依据相应规则进行动态避障,该方法是全局路径规划与局部路径规划的结合,不适用于大范围不确定环境下的路径规划.

现有文献大多是先求得静态确定环境下的最优路径,在环境发生改变时采取相应策略:(1)在当前位置重新规划一条到目标点的路径;(2)在当前位置进行动态避障,然后回到原路径. 前者在处理大规模路径规划问题时,时间消耗较多,无法满足实时性要求. 后者是全局和局部的混合路径规划,比较适用于中小规模,在处理大规模路径规划时效果欠佳或不可行. 鉴于此,本文针对 ALV 大范围远距离路径规划的特点,提出了一种基于路径切换的全局路径规划方法. 该方法首先利用人工萤火虫算法(Glowworm Swarm Optimization, GSO)[18]可以覆盖多个局部最优解的特点,完成对地图空间的最大化搜索,一次生成多条路径. 然后,在多条路径的基础上,提出两种切换算法,一种切换算法用于行驶过程中的进一步调优,另一种切换算法用于环境发生改变时的困境摆脱. 经大量的仿真实验以及近一年的实际试用,充分表明了该方法的可行性和有效性.

2　本文算法的现实基础

本文规划方法的提出依据四个现实基础.

(1)ALV 的任务范围远大于一般机器人

在 ALV 的路径规划中,全局路径规划和局部路径规划是完全分开的. 以栅格地图为例,一个 10km 的任务空间被等分为 200×200 个矩形栅格,每一栅格表示 50m×50m 的空间. ALV 全局路径规划的结果是相隔 50m 左右的路点集,以子任务的形式下达给局部路径规划模块,局部路径规划模块负责完成 50m 内的避障、平滑、速度控制等.

(2)ALV 的时间敏感性

在 ALV 中,地图的装载及全局路径的生成是作为全局任务存在的,对时间不敏感. 但是,ALV 在执行任务的过程中,一旦遇到环境发生变化,需要迅速做出反应,这个阶段对算法的实时性要求较高.

(3)环境的不确定性

在大范围不确定环境下,不存在一条预先生成的最优路径. 普遍意义上的最优路径是指在静态或可预知动态环境下满足特定约束的最小代价路径. 但是,现实情况中环境的改变是不可预知的,同时受限于传感器能力,预先生成的最优路径在某些情况下可能就变成了较差的路径.

(4)人的行为

以人开车为例,在行驶过程中遇到某种现实情况无法继续前行时,往往根据个人经验就近选择岔道从另一条路径到达目的地,很少进行“二次规划”.

基于以上四个现实基础,ALV 全局路径规划应满足几点要求:(1)路径生成阶段应充分利用空间信息,尽可能多的对任务空间进行搜索并保留搜索结果,为后续阶段的重用提供前提;(2)在行驶阶段应具备进一步优化的能力,尽可能使实际行驶路径最优;(3)在遇到环境改变时,应充分重用已有的搜索结果,避免二次规划,在摆脱困境的同时尽可能地满足实时性要求. 基于以上分析,提出了本文算法.

3　人工萤火虫算法的基本原理及特点

随着越来越多的工程问题呈现出多峰、高维和非线性的特点,智能优化算法逐渐的成为了人工智能领域的研究热点. 针对这些复杂问题,传统的精确型算法虽然可以从理论上获得最优解,但是往往由于其时间和空间复杂度过高而变得不可行. 常见的智能优化算法,如文献[11~17],均基于随机搜索,无法保证一定获得最优解,但可以在较低的时间和空间复杂度下获得令人满意的解. GSO 是由印度学者 Krishnanand 和 Ghose 在 2005 年提出的一种新型智能优化算法,目前已经成功应用在多模态函数优化[18,19]、多信号源定位[20]、多机器人系统[21]等方面.

在 GSO 中,每个萤火虫代表问题的一个可行解,并把解的适应度表征为该萤火虫的亮度(荧光素值). 每

次迭代开始,每个萤火虫在其感知范围内搜索比自身亮的个体构建邻域集,并以轮盘赌的方式随机向其移动. 在所有萤火虫移动完毕后,依据自身邻域集的大小更新各自感知半径,同时更新亮度进入下一轮迭代. 通过多次迭代,萤火虫将聚集在多个较亮个体的周围. 以连续域函数优化为例,算法步骤如下:

步骤 1 在解空间内随机构建 $m \in N$ 个可行解 $x_1, x_2, \cdots, x_m$.

步骤 2 萤火虫 x_i 以式(1)更新亮度 l_i.

$$l_i(t+1) = (1-\rho) l_i(t) + \gamma J(x_i(t+1)) \quad (1)$$

$\rho \in (0,1)$ 为荧光素挥发因子,$\gamma \in (0,1)$ 为荧光素更新因子,$J(x_i)$ 表示 x_i 的适应度.

步骤 3 萤火虫 x_i 以式(2)构建邻域集 N_i.

$$N_i(t) = \{x_j \mid l_i(t) < l_j(t), \| x_j(t) - x_i(t) \| < r_i(t)\} \quad (2)$$

$\| \cdot \|$ 表示欧氏距离,r_i 为萤火虫 i 的感知半径.

步骤 4 萤火虫 x_i 通过式(3)所得概率 p_{ij} 以轮盘赌的方式向萤火虫 x_j 移动,式(4)为移动公式,s 为移动步长.

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{u \in N_i(t)} l_u(t) - l_i(t)} \quad (3)$$

$$x_i(t+1) = x_i(t) + s \times \left(\frac{x_j(t) - x_i(t)}{\| x_j(t) - x_i(t) \|} \right) \quad (4)$$

步骤 5 萤火虫 x_i 以式(5)更新感知半径.

$$r_i(t+1) = \min\{r_s, \max\{0, r_i(t) + \beta(N_t - |N_i(t)|)\}\} \quad (5)$$

r_s 为最大感知半径,β 为决策域更新系数,N_t 为邻域集所包含元素个数的阈值,$|\cdot|$ 表示集合所包含的元素个数.

步骤 6 如果达到规定次数或当前最优解符合期望,算法完成. 否则,执行步骤 2.

为了可以更为直观的对算法流程进行阐述,本文选取了一个二维连续域函数实例进行优化,式(6)为其表达式,三维曲面图见图 1. 该函数为多峰函数,三个极大值分别存在于(0.06,1.625)、(-0.45,-0.55)和(1.35,0.05)附近. 图 2 为 50 次迭代的优化过程截图. 其中,"*"表示萤火虫个体,个体间连线表示移动路线. 从图 2(a)至图 2(e)所展示的运行过程可以看出,GSO 优化的过程就是个体在其感知范围内根据亮度进行相互移动的过程. 首先,所有个体根据所在位置的优劣更新自身亮度,然后,随机的向感知范围内比自己亮的个体移动,最后,通过感知范围内个体数量的多寡动态的更新感知半径. GSO 的这种运算方式很容易形成以多个较亮个体为中心的萤火虫子群,如图 2(e)所示,每个子群都覆盖一个局部最优解. 本文利用 GSO 覆盖多个局部最优解的能力[18]一次生成多条规划路径,其中每条路径都是一个局部最优解. 当前多数 GSO 研究都集中在连续多模态优化方面,离散域优化[22]方面的文献相对较少,尚无路径规划方面的文献.

$$f(x_1,x_2) = \frac{(1-x_1)^2}{\exp[x_1^2 + (x_2+1)^2]} - \frac{10(x_1/5 - x_1^3 - x_2^5)}{3\exp(x_1^2 + x_2^2)} - \frac{1}{9\exp[(x_1+1)^2 + x_2^2]} \quad (6)$$

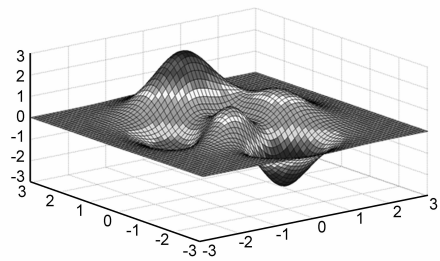
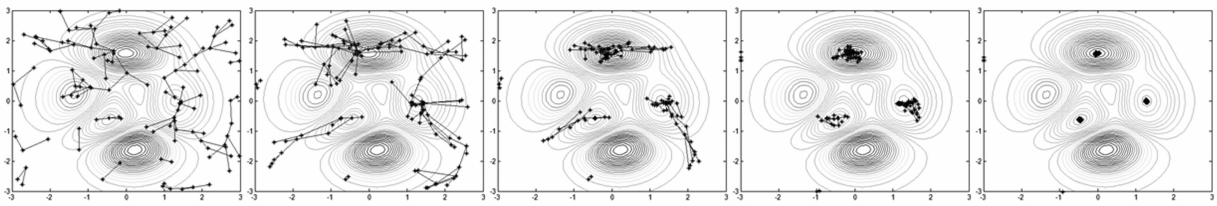


图1 函数三维曲面图



(a) (b) (c) (d) (e)

图2 优化过程图

4 应用于 ALV 全局路径规划的改进人工萤火虫算法

本文算法采用栅格地图,ALV 的规划空间被划分为多个矩形区域,分别按从上到下和从左到右的顺序对行列进行编号,左上角栅格为起点栅格,右下角栅格为目标栅格,其中黑色栅格表示障碍栅格. 为便于进行描述,定义如下:

定义 1 gridmap(i,j)表示地图中第 i 行第 j 列栅格,其邻域栅格集表示为 neighbor(i,j),对角线栅格间距离定义为 14,非对角线栅格间距离定义为 10.

4.1 路径的生成及局部优化

针对于路径规划,每个萤火虫代表一条从起点到终点的无碰路径. 本文算法在启发信息的引导下,从起点开始,通过逐步搜索邻域栅格集构建路径,相关定义及具体步骤如下:

定义 2 gridmap$(i,j)=0$ 表示该栅格无障碍,gridmap$(i,j)=1$ 表示该栅格为障碍栅格,gridmap$(i,j)=2$ 表示该栅格为路径栅格.

定义 3 allowed$(i,j)=\{\text{gridmap}(u,v)\in\text{neighbor}(i,j)\mid\text{gridmap}(u,v)=0\}$ 表示栅格 gridmap(i,j) 的可行栅格集,并记$\chi(i,j)=|\text{allowed}(i,j)|$为 gridmap$(i,j)$的自由度,$|\cdot|$表示集合的大小.

定义 4 有序集 path$=\{\text{gridmap}_1(i_1,j_1),\text{gridmap}_2(i_2,j_2),\cdots,\text{gridmap}_n(i_n,j_n)\}$表示栅格数为 $n(n\in N,n\geq 2)$的一条路径,其中 gridmap$_k(i_k,j_k)(k\leq n)$的自由度简记为χ_k.

为了方便描述算法,gridmap(i_s,j_s)表示起点栅格,gridmap(i_c,j_c)表示当前栅格,gridmap(i_t,j_t)表示目标点栅格,

步骤 1 将起点栅格 gridmap(i_s,j_s)放入路径 path,置 gridmap$(i_s,j_s)=2$,并设置为当前栅格.

步骤 2 针对当前路径栅格 gridmap(i_c,j_c),根据定义 3 计算其可行栅格集 allowed(i_c,j_c)和自由度$\chi(i_c,j_c)$.

(1)若$\chi(i_c,j_c)=0$,则转去执行步骤 6.

(2)若$\chi(i_c,j_c)\neq 0$ 且 gridmap$(i_t,j_t)\in$allowed(i_c,j_c),则将 gridmap(i_t,j_t)加入 path,算法完成,path 即为所得路径.

(3)若$\chi(i_c,j_c)\neq 0$ 且 gridmap$(i_t,j_t)\notin$allowed(i_c,j_c),则生成权值栅格集 weight$(i_c,j_c)=\{w_{\text{gridmap}(u,v)}\mid\text{gridmap}(u,v)\in\text{allowed}(i_c,j_c)\}$,其中 $w_{\text{gridmap}(u,v)}$表示栅格 gridmap(u,v)的权值,置 weight(i_c,j_c)中所有元素初值为 0,执行步骤 3.

步骤 3 对$\forall w_{\text{gridmap}(u,v)}\in\text{weight}(i_c,j_c)$依次执行式(7)、(8)进行更新.

$$w_{\text{gridmap}(u,v)}=\begin{cases}w_{\text{gridmap}(u,v)}+3, & |i_t-i_c|>|i_t-u|\\ w_{\text{gridmap}(u,v)}+2, & |i_t-i_c|=|i_t-u|\\ w_{\text{gridmap}(u,v)}+1, & |i_t-i_c|<|i_t-u|\end{cases}\tag{7}$$

$$w_{\text{gridmap}(u,v)}=\begin{cases}w_{\text{gridmap}(u,v)}+3, & |j_t-j_c|>|j_t-v|\\ w_{\text{gridmap}(u,v)}+2, & |j_t-j_c|=|j_t-v|\\ w_{\text{gridmap}(u,v)}+1, & |j_t-j_c|<|j_t-v|\end{cases}\tag{8}$$

步骤 4 针对$\forall w_{\text{gridmap}(u,v)}\in\text{weight}(i_c,j_c)$,以式(9)所得概率 $p_{\text{gridmap}(u,v)}$采用轮盘赌方式选择要转移到下一栅格gridmap$_{\text{next}}(i_c,j_c)$.

$$p_{\text{gridmap}(u,v)}=\frac{w_{\text{gridmap}(u,v)}}{\sum_{w_{\text{gridmap}(x,y)}\in\text{weight}(i_c,j_c)}w_{\text{gridmap}(x,y)}}\tag{9}$$

步骤 5 将 gridmap$_{\text{next}}(i_c,j_c)$加入路径 path,置 gridmap$_{\text{next}}(i_c,j_c)=2$,置当前栅格自由度$\chi(i_c,j_c)=\chi(i_c,j_c)-1$,然后,将gridmap$_{\text{next}}(i_c,j_c)$设为当前栅格,转去执行步骤 2.

步骤 6 令 $k=|\text{path}|$为路径所含栅格数.

(1)若$\chi_k=0$ 且 $k>1$,则将gridmap$_k(i_k,j_k)$从 path 中删除,转去执行步骤 6.

(2)若$\chi_k=0$ 且 $k=1$,则从起点到目标点无通路,算法结束.

(3)若$\chi_k\neq 0$,则令gridmap$_k(i_k,j_k)$为当前栅格,转去执行步骤 2.

图 3 为流程图,左虚线框内为路径搜索流程,右虚线框内为不可行路线回溯流程.

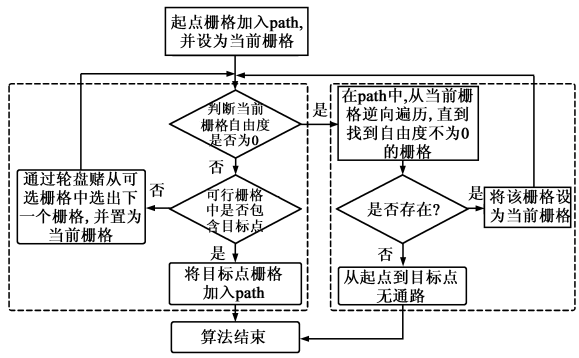
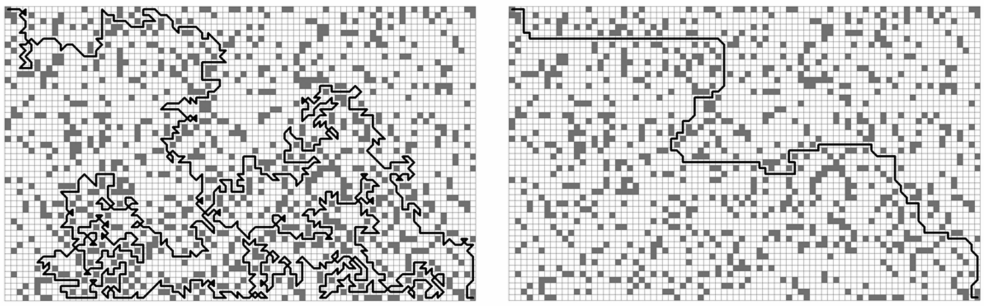


图3 路径生成算法流程图

图 4(a)为路径生成算法的结果图,从中可以看出,生成的路径存在多处自相交,需要对其进行优化. 由于优化方法较为常见,此处仅进行简单描述:从起点栅格开始,对路径所包含的任一栅格,依次进行向右和向下扫描,若该栅格与路径上某栅格之间无障碍阻挡,则将栅格进行直连. 图 4(b)为优化后路径.

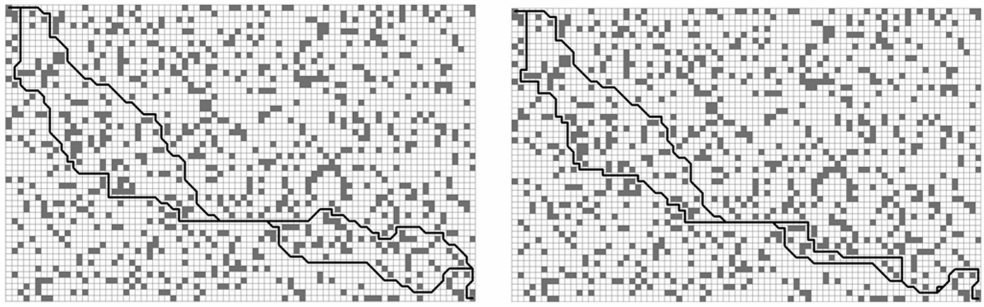
4.2 路径间的距离及移动

求解连续域问题时,GSO 通过欧氏距离衡量解之间的差异,由于路径维度不尽相同,所以并不适用. 同时,用路径长度差衡量也不尽合理,相同长度的两个路径,很可能完全不同. 本文用两条路径所包含的栅格数来表示其距离,两条路径完全重合时,其距离为 0. 相应的,标准 GSO 的移动方式见式(4),也不适用于路径. 本文路径的移动采用重新生成的方法,即在两条路径包含的栅格内以 4.1 节算法重新生成一条路径,并替换原先适应度较差的路径. 图 5(a)为移动前的两条路径,图 5(b)为移动后的两条路径,从中可看出,在移动后两条路径包含的栅格数小了,也即两者距离在缩小.



(*a*) 原始路径 (*b*) 局部优化后的路径

图4 路径局部优化



(*a*) 移动前的两条路径 (*b*) 移动后的两条路径

图5 路径移动图

4.3 人工萤火虫算法的改进及多条路径的生成

为了使 GSO 覆盖更多的局部最优解和获得更高质量的全局最优解,本文对原算法做三点改进:

(1)改变亮度更新方式

基本 GSO 的亮度见式(1),是历史荧光素值与当前适应值在不同提取比例下的累加. 当荧光素挥发因子 ρ 取值较小时,亮度的更新速度将与适应值的更新速度产生较大的延迟,由于 GSO 种群是通过亮度进行相互移动的,所以,该延迟将导致种群有更大的可能在原位置附近进行更多的选择并移动. 此时,在较大的移动步长下,个体将有更多的可能跳出所在的局部最优区域,有利于抑制算法的早熟,但算法收敛速度下降. 当 ρ 取值最大值 1 时,适应值与亮度成正比,种群将直接向较亮(适应值高)的个体聚集,导致算法快速的收敛于多个局部最优解. 由于本文所提算法是一种基于多路径的路径规划方法,更为侧重算法产生的多个局部最优解,因此,式(10)导致的早熟问题对本文算法影响不大. 鉴于其速度优势,本文以式(10)进行亮度的更新.

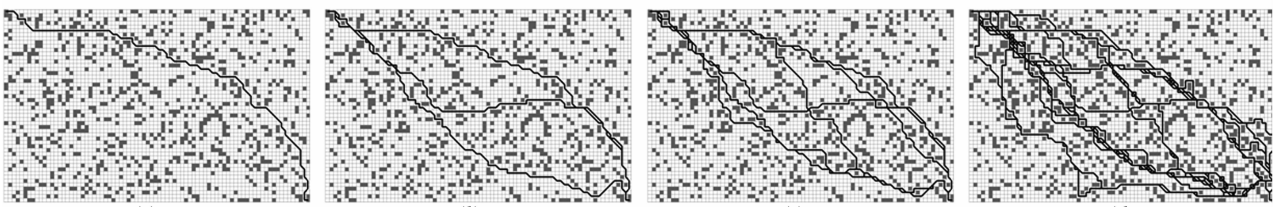
$$l_i(t+1) = \gamma J(x_i(t)) \tag{10}$$

(2)局部最优解的处理

如果萤火虫连续一定代数邻域集为空,则认为该萤火虫所代表的解为一个局部最优解,首先保存该解,然后执行 4.1 算法,重新生成可行解代替该解. 本文算法生成的多条路径实质上就是包括全局最优解在内的各个局部最优解,该处理方法在防止了局部最优解被优化掉的同时提高了算法的搜索能力.

(3)局部次优解的移动

如果一个解与其邻域集中某个解的距离小于一设定阈值,则认为该解为局部次优解,执行 4.1 算法,重新生成可行解替代该解. 路径解有一定的特殊性,当两条路径距离很小时,很可能因为中间存在障碍而无法进



(*a*) (*b*) (*c*) (*d*)

图6 最优路径图

一步移动. 本文局部次优解的移动方法避免了无用的移动,提高了算法的搜索能力,使算法覆盖更多的局部最优解.

在算法完成指定次数的迭代后,本文将保存的局部最优解和迭代完成后的最终解集按路径长度统一进行排序,然后选取指定数量的路径解做为最终结果. 经仿真实验表明,选取路径数等于栅格地图短边长度时效果较为理想.

图 6 为算法的运行结果图. 图 6(a)为最优路径,图 6(b)为最优的 3 条路径,图 6(c)为最优的 6 条路径,图 6(d)为最优的 20 条路径.

5　路径切换算法

为了对路径进一步寻优和遇到环境改变时快速做出反应,本文针对性的提出两种路径切换算法,相关定义如下:

定义 5　若栅格 gridmap(i,j)被 $n \in N$ 条路径包含,则记该栅格重叠度为 n,包含该栅格的路径集为重叠路径集,以 $\psi(i,j)$ 表示.

调优切换算法　设当前栅格为 gridmap(i,j),若其重叠度为 $n(n \in N, n \geq 2)$,则以当前栅格为起点重新计算 $\psi(i,j)$ 中所有路径的剩余长度并排序,选择其中最短路径继续行进.

脱困切换算法　脱困切换算法是一种类似于 A*[23] 的搜索算法,但启发函数和搜索目标与 A* 不同. 在 ALV 因遇到环境改变而无法继续行进时,本算法被调用. 本算法从当前栅格开始,在式(7)、(8)所得权值的启发下,逐步搜索可行栅格集,直至到达适当路径,算法结束. 相关定义及描述如下:

定义 6　ALV 启动脱困切换算法时所在栅格定义为切换起始栅格,发生环境改变的栅格定义为改变栅格.

为方便算法描述,每一个栅格被定义为四元组 $((i,j),f,w,d)$,(i,j) 表示该栅格位置,f 表示该栅格的父栅格,w 表示该栅格的权重,d 表示切换起始栅格到该栅格的最短距离. 并设当前所有路径为 $\text{paths} = \{\text{path}_1, \text{path}_2, \cdots, \text{path}_n\}$ $(n \in N)$,切换起始栅格为 (i_h, j_h),当前改变栅格为 (i_b, j_b).

路径切换算法

PATHSWITCH(gridmap, paths, (i_h,j_h), (i_b,j_b))
输入:全局地图,路径集,切换起始栅格,改变栅格
输出:切换路径
01　创建有序集合 opened = {((i_h,j_h), NULL, 0, 0)}
02　创建集合 closed = { }
03　WHILE opened ≠ { } DO
04　　将 opened 中第一个元素 v 插入 closed
05　　从 opened 中删除 v
06　　IF $v \in \{\text{paths} - \psi(i_b,j_b)\}$ THEN
07　　　算法完成,v 以父栅格回溯到切换起始栅格,即为切换路径
08　　END IF
09　　构建栅格集 $N(v) = \{\text{neighbor}(v) \mid v \notin \text{closed}, \text{gridmap}(v) \neq 1\}$
10　　以式(7)、(8)更新 $N(v)$ 中所有权重
11　　$N(v)$ 依权重排序
12　　FOR $x \in N(v)$ DO
13　　　以式(9)计算 $d(x,v)$
14　　　IF $x \neq x_{\text{opened}} \in$ opened THEN
15　　　　$x.f \leftarrow v$
16　　　　$x.d \leftarrow v.d + d(x,v)$
17　　　　将 x 插入 opened 尾
18　　　ELSE
19　　　　IF $v.d + d(x,v) < x_{\text{opened}}.d$ THEN
20　　　　　$x_{\text{opened}}.f \leftarrow v$
21　　　　　$x_{\text{opened}}.d = v.d + d(x,v)$
22　　　　END IF
23　　　END IF
24　　END FOR
25　END WHILE

6　实验与分析

为验证算法的有效性及正确性,本文在 50 × 80 的栅格地图上随机生成 1000 个障碍进行仿真. 实验环境为 AMD 2.9GHz CPU,2GB 内存,VS2010 编译套件.

6.1　切换算法仿真及耗时分析

6.1.1　调优切换算法仿真

如图 7 所示,选取 50 条路径进行测试,其中蓝色线段为算法在起点评估的最优路径,即静态环境下的最短路径(长度 1156),红色线段为最终的行驶路径(长度 1124). 在行驶过程中,每经过一次路径交叉点,算法都会对交叉路径进行重新评估并切换到最优路径,行驶的过程是一个不断优化的过程.

6.1.2　脱困切换算法仿真

为了便于理解,选取某次实验结果中 6 条距离间

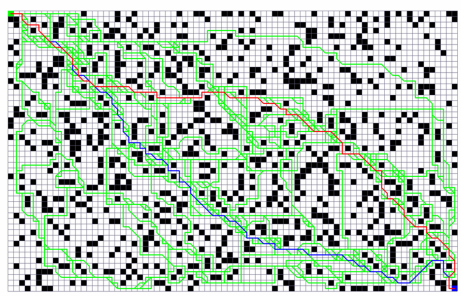
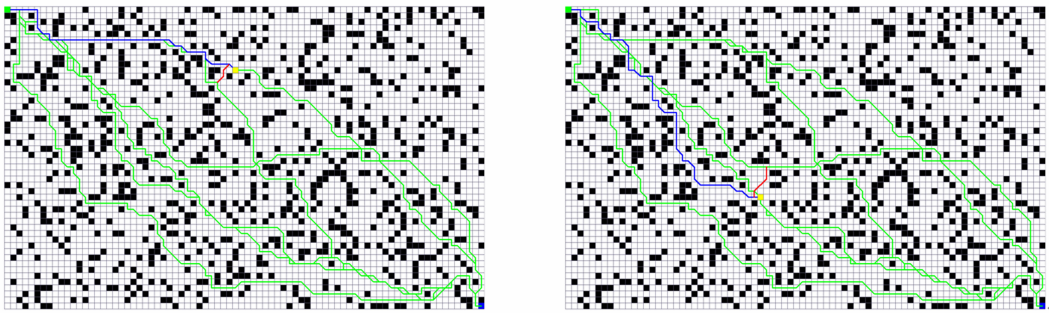


图7　调优路径切换示例

隔较大的路径进行仿真,结果如图 8 所示,其中黄色栅格表示改变栅格,蓝色线段为原行驶路径,红色线段为脱困切换算法产生的切换路径.

图 8(a)表示改变栅格重叠度为 1 的脱困路径切换算法结果,图 8(b)表示改变栅格重叠度为 2 的脱困路径切换算法结果. 可明显看出,算法合理的重用了原有的搜索结果,成功切换到了其它路径,避免了二次规划. 图 8(b)中,路径切换导致了实际行驶路径暂时偏离目标点,但后续会通过调优切换算法逐步进行优化,使代价降到最低.



(a) 改变栅格重叠度为1 (b) 改变栅格重叠度为2

图8 脱困路径切换示例

6.1.3 路径切换算法耗时分析

由于本文算法预先生成多条路径,在行驶过程中仅需调用两种切换算法,在所耗时间方面无法直接与其它算法对比,所以仅对两个切换算法做了统计实验,表 1 为两种算法各进行 50 次独立运算的实验结果.

表 1 切换算法耗时

算法	最短耗时/ms	最长耗时/ms	平均耗时/ms
调优切换	0	0	0
脱困切换	0	3	1.26

本仿真在 WINDOWS 系统上进行,最小计时单位为 1ms,最短耗时和最长耗时有一定的不准确因素,但总体来看,两种切换算法完全满足 ALV 的实时性要求.

6.2 路径质量的比对

为了在寻优能力方面进行验证对比,本文实现了静态环境下的 A* 算法和蚁群算法. 其中,A* 算法使用欧氏距离做启发信息,满足 $\forall h(x) \le h^*(x)$,针对栅格地图必然获得最优解[23],因此在本实验中以 A* 算法所得结果为比对准则. 本文算法和蚁群算法各进行 20 次实验,每次进行 500 次迭代,统计结果见表 2.

表 2 寻优实验比对结果

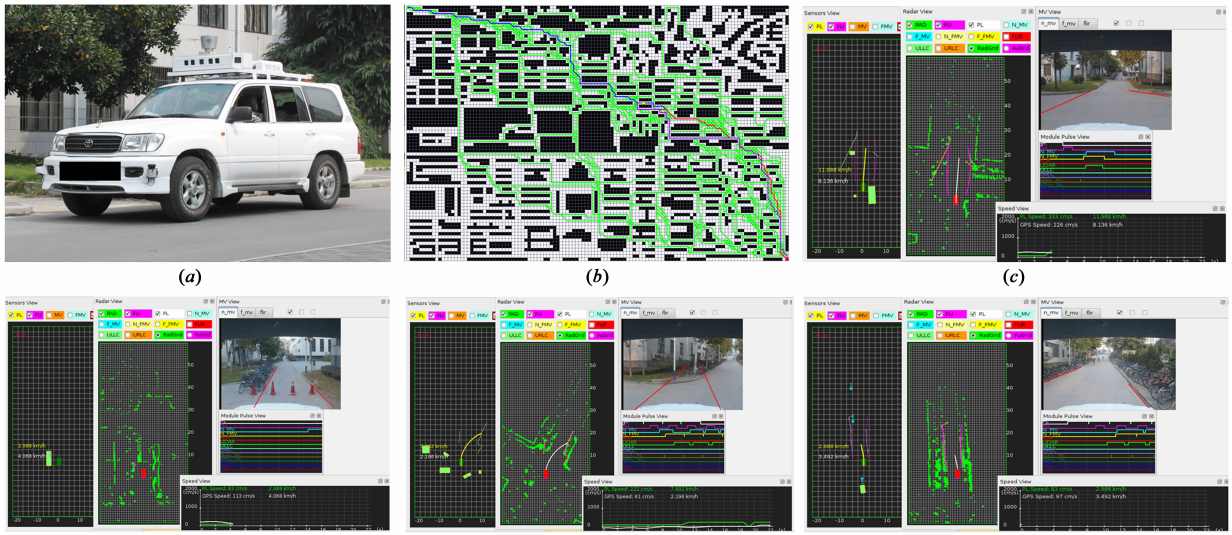
算法	最小偏差(%)	最大偏差(%)	平均偏差(%)	参 数
A* 算法	0	0	0	——
蚁群算法	6.3	21.4	11.7	$m=30,\alpha=1$ $\beta=7,\rho=0.5$
本文算法	1.8	9.9	4.6	$n=80,\beta=30$ $N_t=5,r_x=500$

在表 2 中,所有偏差均相对于 A* 算法,例如 A* 所得路径长度为 100,蚁群算法所得路径长度为 106,则蚁群算法偏差为 6%. 从表中可看出,本文算法有较强的寻优能力,非常接近最优解. 另外,本文调优切换算法会对路径做进一步的优化,实际行驶路径比表中所示更优.

6.3 不确定环境下的实体测试

实验平台为自主研制的 ALV,如图 9(a)所示,其中,全局路径规划模块计算机为 2.0GHz CPU,1GB 内存. 所选实验场地的规模为 2400m × 1600m,以每栅格表示 20m × 20m 的矩形区域,可得 120 × 80 的栅格地图,如图 9(b)所示. 为了充分验证本文算法的可行性和有效性,本实验选取文献[16]的蚁群算法和文献[17]的鱼群算法在是否成功脱困、脱困算法耗时和成功脱困后的路径长度三个方面进行了对比,本文算法参数与 6.2 相同,对比算法参数参照相应文献. 图 9(b)为全局路径规划图,蓝色线段为本文算法所得最优路径,自主车开始行进时,立即在最优路径经过的某位置(黄色栅格)设置人工路障,用以模拟环境的不确定性,红色线段表示执行本文脱困切换算法后的路径,紫色线段为文献[15]的蚁群算法所得重规划路径. 图 9(c)为正常行驶的系统监控图,顶层三个子窗口分别监控局部路径规划、三维雷达和二维视觉. 图 9(d)为 ALV 陷入困境的截图,从中可看出,局部路径规划已经失败,此时向全局路径规划模块发送困境摆脱请求. 图 9(e)中,局部路径规划模块以新路点为目标进行转向规划. 图 9(f)中,ALV 已完成困境摆脱. 为了更为详细的进行对比分析,表 3 列出了对比数据. 由于文献[17]更侧重于中小范围下的动态避障,所以在本实验中并未成功脱

困. 文献[16]通过重规划的方法成功摆脱困境,但重规划耗时较多,无法满足 ALV 路径规划的实时性要求. 相比来看,本文算法不仅可以成功脱困,而且在路径长度和切换耗时方面均优于对比算法. 通过实体实验,充分表明本文算法各项指标均满足 ALV 全局路径规划的需求,效果非常理想.



(a)　(b)　(c)　(d)　(e)　(f)

图9　实体测试

表 3　实体测试对比

算法	是否成功	路径长度(m)	切换耗时(s)
蚁群算法	是	1852	82
鱼群算法	否	——	0
本文算法	是	1708	0

7　结论

针对 ALV 路径规划的特点,本文提出了一种不确定环境下基于改进萤火虫算法的 ALV 全局路径规划方法. 相对于传统处理方法预先生成静态环境下的一条最优路径,本文算法预先生成多条路径,充分挖掘了规划空间的信息,增强了算法的搜索能力,为进一步的重用和调优提供了前提条件. 在生成多条路径的基础上,提出了调优路径切换算法,对路径做进一步的优化,使 ALV 实际行驶路径达到最优化. 本文脱困路径切换算法,在遇到环境改变时直接切换到备选路径,重用了已有的搜索结果,避免了二次规划. 通过大量的仿真实验及将近一年的实际试用,本文算法在实际行驶路径长度和路径切换实时性上均取得了非常理想的效果. 下一步的工作是本文算法在更为复杂环境下的应用以及相关收敛性理论的研究.

参考文献

[1] Ge S S, Cui Y J. Dynamic motion planning for mobile robots using potential field method [J]. Autonomous Robots, 2002, 13(2): 207 – 222.

[2] Csiszar A, Drust M, Dietz T, et al. Dynamic and Interactive Path Planning and Collision Avoidance for an Industrial Robot Using Artificial Potential Field Based Method[M]. Berlin: Springer, 2012. 413 – 421.

[3] Jaradat M A K, Garibeh M H, Feilat E A. Autonomous mobile robot dynamic motion planning using hybrid fuzzy potential field [J]. Soft Computing, 2012, 16(1): 153 – 164.

[4] Pan C Z, Lai X Z, Yang S X, et al. An efficient neural network approach to tracking control of an autonomous surface vehicle with unknown dynamics [J]. Expert Systems with Applications, 2013, 40(5): 1629 – 1635.

[5] Masoud A A. Managing the dynamics of a harmonic potential field-guided robot in a cluttered environment [J]. IEEE Transactions on Industrial Electronics, 2009, 56(2): 488 – 496.

[6] Pal A, Tiwari R, Shukla A. Modified A* Algorithm for Mobile Robot Path Planning[M]. Berlin: Springer, 2012. 183 – 193.

[7] Stentz A. Optimal and efficient path planning for partially-known environments [A]. Proceedings of the 1994 International Conference on Robotics and Automation[C]. San Diego, USA: IEEE Computer Society, 1994. 3310 – 3317.

[8] Ferguson D, Stentz A. The delayed D* algorithm for efficient path replanning [A]. Proceedings of the 2005

IEEE International Conference on Robotics and Automation, ICRA 2005 [C]. Barcelona, Spain: IEEE Press, 2005. 2045 – 2050.

[9] Devaurs D, Siméon T, Cortés Mastral J. Parallelizing RRT on large-scale distributed-memory architectures [J]. IEEE Transactions on Robotics and Automation, 2013, 29(2): 571 – 579.

[10] 宋金泽,戴斌,单恩忠,等. 一种改进的 RRT 路径规划算法[J]. 电子学报,2010,38(2A):225 – 228.
SONG Jin-ze, DAI Bin, SHAN En-zhong, et al. An improved RRT path planning algorithm [J]. Acta Electronica Sinica, 2010, 38 (2A): 225 – 228. (in Chinese)

[11] Lee J, Kang B Y, Kim D W. Fast genetic algorithm for robot path planning [J]. Electronics Letters, 2013, 49 (23): 1449 – 1451.

[12] 李志海,付宜利. 基于遗传算法的仿生双足爬壁机器人越障运动规划[J]. 机器人,2012,34(6):751 – 757.
LI Zhi-hai, FU Yi-li. Motion planning of a bio-inspired biped wall climbing robot stepping over obstacles based on genetic algorithm[J]. Robot, 2012, 34(6): 751 – 757. (in Chinese)

[13] Chen X, Kong Y, Fang X, et al. A fast two-stage ACO algorithm for robotic path planning[J]. Neural Computing and Applications, 2013, 22(2): 313 – 319.

[14] Zhao J, Fu X. Improved ant colony optimization algorithm and its application on path planning of mobile robot[J]. Journal of Computers, 2012, 7(8): 2055 – 2062.

[15] 陈雄,赵一路,韩建达. 一种改进的机器人路径规划的蚁群算法[J]. 控制理论与应用,2010,27(6):821 – 825.
Chen Xiong, Zhao Yi-lu, Han Jian-da. An improved ant colony optimization algorithm for robotic path planning [J]. Control Theory & Applications, 2010, 27(6): 821 – 825. (in Chinese)

[16] 柳长安,鄢小虎,刘春阳,等. 基于改进蚁群算法的移动机器人动态路径规划方法[J]. 电子学报,2011,39(5):1220 – 1224.
LIU Chang-an, YAN Xiao-hu, LIU Chun-yang, et al. Dynamic path planning for mobile robot based on improved ant colony optimization algorithm [J]. Acta Electronica Sinica, 2011, 39 (5): 1220 – 1224. (in Chinese)

[17] 徐晓晴,朱庆保. 动态环境下基于多人工鱼群算法和避碰规则库的机器人路径规划[J]. 电子学报,2012,40(8):1694 – 1700.
XU Xiao-qing, ZHU Qing-bao. Multi-artificial fish-swarm algorithm and a rule library based dynamic collision avoidance algorithm for robot path planning in a dynamic environment[J]. Acta Electronica Sinica, 2012, 40(8): 1694 – 1700. (in Chinese)

[18] Krishnanand K N, Ghose D. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions[J]. Swarm Intelligence, 2009, 3(2): 87 – 124.

[19] Wu B, Qian C, Ni W, et al. The improvement of glowworm swarm optimization for continuous optimization problems [J]. Expert Systems with Applications, 2012, 39(7): 6335 – 6342.

[20] Krishnanand K, Ghose D. A glowworm swarm optimization based multi-robot system for signal source localization[J]. Design and Control of Intelligent Robotic Systems, 2009: 49 – 68.

[21] Krishnanand K N, Ghose D. Theoretical foundations for rendezvous of glowworm-inspired agent swarms at multiple locations [J]. Robotics and Autonomous Systems, 2008, 56(7): 549 – 569.

[22] 周永权,黄正新,刘洪霞. 求解 TSP 问题的离散型萤火虫群优化算法[J]. 电子学报, 2012, 40 (6): 1164 – 1170.
Zhou Y Q, Huang Z X, Liu H X. Discrete glowworm swarm optimization algorithm for TSP problem[J]. Acta Electronica Sinica, 2012, 40 (6): 1164 – 1170. (in Chinese)

[23] 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.

作者简介

杜鹏桢 男,1982 年 11 月出生. 2005 年本科毕业于南京理工大学电子工程与光电技术学院,现为南京理工大学计算机科学与工程学院硕博连读生,从事机器人及智能计算相关研究.
E-mail: h. k@ foxmail. com

唐振民 男,教授,1961 年生. 南京理工大学计算机科学与工程学院院长、博士、博士生导师,主要研究方向是模式识别与智能系统,包括:智能机器人与目标识别、图象处理与模式识别等.
E-mail: tzm. cs@ njust. edu. cn

陆建峰 男,教授,1969 年生. 主要研究方向是图像处理与分析、模式识别、数据挖掘等.
E-mail: lujf@ njust. edu. cn