# 一种满足方向约束的动态航迹规划方法*

周 璐, 李军华
(南昌航空大学 信息工程学院, 南昌 330063)

**摘 要**: 针对动态环境下有方向约束航迹规划问题,提出一种结合引导点的动态航迹规划方法。该方法沿约束方向基于圆拓展的方式产生引导点,并自主选择代价最小的引导点,引导航迹规划算法向引导点区域搜索,提高了规划效率。仿真结果表明,改进算法可以适应动态变化的环境,也能满足从特定方向接近目标点的航迹规划要求。相比于传统 D* 算法,改进算法的航迹总代价更小、规划时间更短。

**关键词**: D* 算法; 动态环境; 航迹规划; 方向约束; 引导点
**中图分类号**: T301.6 **文献标志码**: A **文章编号**: 1001-3695(2018)07-1982-04
doi:10.3969/j.issn.1001-3695.2018.07.013

# Dynamic route planning method based on directional constraints

Zhou Lu, Li Junhua
(*School of Information Engineering*, *Nanchang Hangkong University*, *Nanchang* 330063, *China*)

**Abstract**: Aiming at the directional constraints path planning problem in dynamic environment, this paper proposed a dynamic path planning algorithm which combined the guide points. The algorithm generated guide points on the basis of the circle expansion method along the constraints direction, and then selected the least cost guide point to guide the planning algorithm to search the guide points region. Finally it could improve the overall planning efficiency. The simulation results show that the improved algorithm can adapt to the dynamic environmental changes, as well as plan the resultant path to the target point which goes from the specific direction. Compared with the traditional D* algorithm, the total path cost of the improved algorithm is smaller and the planning time is shorter.

**Key words**: D* algorithm; dynamic environment; path planning; direction constraints; guide point

无人机(UAV)航迹规划是指在规划空间中利用地形和情报等信息,同时考虑无人机自身的性能约束和任务要求,规划出从起始点到目标点飞行航迹代价最小的一条突防轨迹[1,2]。航迹规划作为无人机自主作战的核心,其重要性相当于人类的“大脑”。伴随着无人机在现代战争、灾后救援、安全预警等方面的成功应用,吸引了越来越多的国内外学者投入到无人机航迹规划的研究中来。近些年出现了多种航迹规划方法,可大致将它们分为离线航迹规划方法和在线航迹规划方法两类[3,4]。

离线航迹规划方法主要包括稀疏 A* 搜索(SAS)算法、遗传算法、蚁群算法、粒子群算法等基于进化计算的航迹规划方法。Szczerba 等人[5]提出 SAS 算法,通过融入无人机自身约束条件,如最小步长、最小最大转弯角等,大大缩减了搜索空间,使算法效率得以提高。唐晓东等人[6]对 SAS 算法进行了改进,成功应用于三维空间航迹规划。刘新等人[7]提出基于分层策略的航迹规划方法,该方法在静态环境中先利用遗传算法在全局规划出引导点集,然后在局部规划中结合全局产生的引导点集利用 SAS 算法规划出满足约束条件的可行航迹。相比单一的算法,其较好地缩短了算法规划时间。刘琼昕等人[8]在刘新等人的基础上继续改进,进一步提高了该算法的稳定性。但该方法只适用于静态环境,对于动态变化的环境缺乏一定的适用性。杨杰等人[9]提出一种动态引导 A* 算法,在原有 A* 算法的基础上,引入引导点等启发信息,可以让 A* 算法规划出满足方向约束的航迹。但该方法中引导点的产生是通过 A* 算法反向规划产生,因其未反向规划出整条航迹,所以引导点的质量无法保证;同时该方法中引导点的产生和选择过程无法精确控制,对于动态变化的环境缺乏一定的适用性。

随着科技的不断进步,战场环境不再是一成不变,而是不断动态变化。传统的静态航迹规划方法已经无法满足无人机自主作战与飞行需要,因此在线航迹规划是当今研究的重点。常见的在线航迹规划方法有 D* 算法、基于可行优先准则的实时航迹规划等。魏铁涛等人[10]提出一种在线航迹规划的滚动优化方法,通过引入新的末端惩罚项来构建一种能够实时反映未来航迹段偏离程度的性能指标,使无人机对未来航迹段的变化提前作出反映。Chen 等人[11]对中心引力算法作出改进,使该算法在动态环境中可以有效地规避威胁。Wen 等人[12]提出动态域快速扩展随机树(DDRRT)与线性二次高斯运动规划(LQG-MP)相结合的方法,该方法较快速扩展随机树(RRT)法所规划航迹质量有进一步提升。王绪芝等人[13]对蚁群算法进行改进,将改进的蚁群算法与改进的模型预测控制(MPC)算法相结合应用于动态环境的航迹规划研究。D* 算法是一种常见的在线实时航迹规划算法,该算法能较好地适应环境的动态变化[14,15]。但是当目标点周围出现大面积遮挡,要求无人机从特定的方向进入目标点时,D* 算法在遮挡区会进行大量的反复搜索和回退拓展,消耗大量系统资源,使算法规划效率低下。本文为了解决 D* 算法在上述情形效率低下问题,提出一种能够在动态环境下自动生成、选择引导点的方法,将其与 D* 算法相结合应用于航迹规划,提高算法规划效率。

## 1 D* 算法应用于航迹规划

D* (d-star,dynamic A* )算法即动态 A* 算法,是一种基于航迹代价函数评估的常用航迹规划方法[16]。通过向着目标点方向扩展节点,并使用代价函数评估、选取最优节点,进一步向着目标点扩展,逐步完成航迹规划。由于 D* 算法在进行航迹规划时,扩展所生成的航迹节点都是向着目标点所在位置而产生,在保证规划速度的同时,能够确保规划的航迹长度较短。

**收稿日期**: 2017-03-01; **修回日期**: 2017-04-10 **基金项目**: 国家自然科学基金资助项目(61440049,61262019);江西省自然科学基金资助项目(20161BAB202038);江西省科技厅重点研发基金资助项目(20161BBG70047);南昌航空大学研究生创新基金资助项目(YC2016046)

**作者简介**: 周璐(1992-),男,安徽芜湖人,硕士,主要研究方向为进化计算、智能优化(745902845@qq.com);李军华(1974-),男,教授,博士,主要研究方向为进化计算、智能优化.

### 1.1　D* 算法原理

D* 算法主要依靠 OPEN 和 CLOSED 表来实现算法的搜索遍历。OPEN 表用于存放准备扩展的节点信息,CLOSED 表用于存放已被扩展的节点信息。本文中两表的元素结构如图 1 所示。图 1 中,标志位是用于判断扩展节点当前的位置;$X$、$Y$ 分别表示节点的横、纵坐标。

| OPEN表 | 标志位 | 子节点$X$ | 子节点$Y$ | 父节点$X$ | 父节点$Y$ | $h(n)$ | $g(n)$ | $f(n)$ |
|---|---|---|---|---|---|---|---|---|

| CLOSED表 | 当前节点$X$ | 当前节点$Y$ |
|---|---|---|

图 1　D* 算法 OPEN 和 CLOSED 表元素结构

D* 算法在航迹扩展过程中使用的代价函数为

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

其中:$f(n)$为当前节点的代价值;$g(n)$为起始点到当前节点的实际代价;$h(n)$为当前点到目标点的预估代价,其形式根据具体问题的不同,会有不同的表现形式,一般称为启发函数。

### 1.2　D* 算法航迹规划流程

a)导入数字概率地图;

b)初始化各参数,如最小步长 $L_{\min}$、最大扩展节点数 $M$、最大转弯角 $\theta$ 等;

c)输入起始点和目标点的坐标;

d)将当前点指针指向起始点,将起始点放入 CLOSED 表中;

e)判断当前点指针的节点信息是否为目标点,若是,则由 CLOSED 表的节点信息回溯生成航迹,否则转到步骤 f);

f)让当前点指针的节点按照目标移动的方向进行扩展,并满足约束条件,筛选的节点放入 OPEN 表中;

g)选取 OPEN 表中 $f(n)$ 值最小的节点,将当前指针指向它,并将其放入 CLOSED 表中,转到步骤 e)。

### 1.3　D* 算法的不足

D* 算法在动态环境中寻找可行路径较为有效。算法逐步扩展节点向目标点移动,而扩展只检查理想最短路径上下一节点的变化情况,对距离较远的路径上的变化不敏感。当理想航迹上出现大面积遮挡,同时要求无人机从特定的方向进入目标点时,D* 算法在遮挡区进行大量的回退扩展,寻找可行路径,消耗了大量系统资源和时间,使算法规划效率低下。理想航迹线上遮挡示意图如图 2 所示。

## 2　改进 D* 算法航迹规划

由 1.3 节可知,D* 算法效率低下是因为在遮挡区存在大量的回退拓展,其主要原因是缺少一定的信息引导。如果可以在遮挡区周围或者特定约束的进入方向上预先设置一些引导信息,既可以帮助无人机顺利避开遮挡区,同时完成从特定方向进入目标点的要求。本文提出的引导点的思想,通过分析动态环境中目标点的位置,按照基于圆拓展的方式在遮挡区周围自动生成引导点,增加航迹规划过程中的引导信息,提高 D* 算法在强约束条件下的规划效率。

### 2.1　方向约束模型的构建

传统航迹规划算法不考虑方向约束,规划航迹从起始点出发,一旦到达目标点就认为规划结束,如图 2 中目标点周围无遮挡时的理想航迹所示。当要求从特定方向接近目标点时,传统 D* 算法规划效率低下,尤其是进入目标点的角度背离起始点出发角度较大时,D* 算法的不足更为明显。图 3 展示了起始点与目标点的角度关系。图中 $\partial$ 表示无人机出发角度,$\beta$ 表示进入目标点的角度。针对 $\partial$ 与 $\beta$ 相差较大的情况建立遮挡模型,进行仿真规划,要求规划航迹从特定方向接近目标点,这种方式规划所得结果更具代表性。本文将方向约束等效转换为障碍物遮挡约束,通过在目标点周围设置不同的遮挡物,让规划航迹满足不同的方向约束。由于方向约束是任意的,所以遮挡模型也可以是任意形状,遮挡模型并不局限于图 2、4 和 5 中所示的灰色实多边形,其可以是不规则图形,同时遮挡物之间的夹角也可以是任意的。遮挡物的形状及角度大小不影响本文所述引导点的产生,当给定圆拓展的半径后,即可按照本文方法自动生成引导点。圆拓展半径的大小需根据具体约束情况合理设置,当拓展半径接近遮挡物边缘长度时,规划效果较优。为了在有限的条件下进行仿真实验,本文取两个相互垂直的矩形构成遮挡物(如图 2、4 和 5 中的灰色实多边形所示),使无人机进入目标点的角度背离出发点角度尽量大,在苛刻的情况下进行仿真规划,规划结果更具说服性。

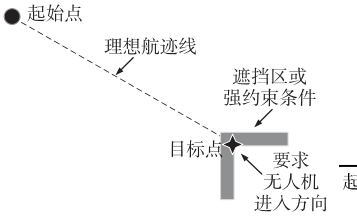



图 2　理想航迹遮挡示意图

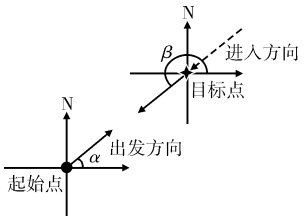



图 3　起始点与目标点的角度示意图

### 2.2　引导点的产生策略

从起始点至目标点的航迹规划过程中,需在目标点附近进行障碍检测,寻找强约束条件下航迹代价最小的规划空间。检测方法是以目标点为圆心,遮挡物 OB 长为半径($O$ 为圆心,即目标点中心),每隔 5°角进行直线检测。将所作圆与障碍物的交点设为引导点。产生引导点后,在半径范围内判断该引导点与起始点的连线方向上是否还存在遮挡。若存在遮挡,则以产生的引导点为新圆心,按上述方法继续产生引导点,直至符合条件为止;若不存在遮挡,则只产生一个引导点即可。引导点产生示意图如图 4 所示。

图 4 中四角星为目标点,黑色圆点为起始点,灰色的实多边形为遮挡物。遮挡物为人为设置,其长度大小均为已知,灰色的点 $D_1 \sim D_5$ 为产生的引导点。首先以目标点为圆心,按照上述方法进行边缘区域检测,得到引导点 $D_1$ 和 $D_2$。由于引导点 $D_1$ 和 $D_2$ 在半径范围内与起始点的连线方向上仍然存在遮挡,所以需要以引导点 $D_1$ 和 $D_2$ 为圆心分别再次进行检测。以引导点 $D_1$ 为圆心进行区域检测,则引导点 $D_3$ 顺时针到目标点的扇形范围为可行引导区域。选取其中半径范围内与起始点连线无遮挡同时距离起始点最近的点作为引导点,即引导点 $D_3$。引导点 $D_4$ 是与第一个检测圆的交点。由于其在引导点 $D_1$ 和 $D_2$ 的范围内,且在遮挡物内部,所以将其舍去。同理,以引导点 $D_2$ 为圆心产生符合条件的可行引导区域是引导点 $D_5$ 逆时针到目标点的扇形范围。其中引导点 $D_5$ 符合条件且最优,引导点 $D_6$ 位于引导点 $D_1$ 的检测圆内,故将其舍去。

### 2.3　引导点的选择

在结合引导点的航迹规划中,引导点的位置对整个航迹走势影响较大,因此引导点设置得合适与否,可直接影响到航迹的优劣。本文设置引导点的目的如下:a)减少 D* 算法在遮挡物附近反复遍历扩展节点所消耗的时间,提高算法规划效率;b)使无人机从目标点的特定方向进入目标点,满足方向约束要求。由 2.2 节的内容可知,图 4 中的可行航迹线有两条,它们分别是:a)起始点—引导点 $D_3$—引导点 $D_1$—目标点;b)起始点—引导点 $D_5$—引导点 $D_2$—目标点。

本文所提出的引导点在航迹规划中不是必经点,即航迹不需要经过该点。引导点只是辅助无人机进行目标搜索,提高其搜索效率,同时帮助无人机高效完成有特殊约束条件的任务,如方向敏感性任务。选择不同的引导点,对整个航迹的走势有很大影响。本文采用代价值函数 $f_{\text{total}}$ 进行引导点的选择,主要考虑航迹长度和航迹威胁代价两个因素。从起始点向着符合条件的引导点进行直线规划,选择航迹总代价较小的引导点为目标引导点。代价值函数表达式为

$$f_{\text{total}}(n)=\sum_{i=1}^{n}(\omega_1 \times l_i+\omega_2 \times f_i) \tag{2}$$

其中:$l_i$ 为节点 $i-1$ 到 $i$ 之间的航迹长度;$f_i$ 为节点 $i-1$ 到 $i$

之间受到的威胁概率值;$\omega_1$、$\omega_2$ 为相应权重系数。

引导点的选择示意图如图 5 所示。图 5 中,圆形威胁区域的威胁值在一定范围内有值,且均匀分布,覆盖整个圆形区域;方形威胁区域为禁飞区,其威胁值无限大,无人机在飞行过程中必须要绕过此威胁区域。由起始点至引导点 $D_3$ 和 $D_5$ 分别进行直线规划,取使得 $f_{total}$ 有较小值的引导点为目标引导点。

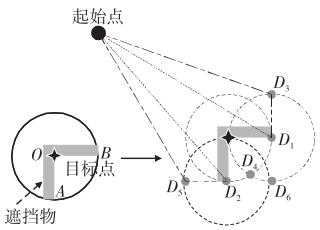



图 4 引导点产生示意图

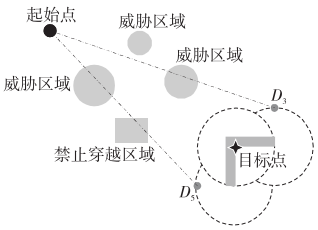



图 5 引导点选择示意图

### 2.4 改进算法代价函数的确定

本文评价航迹的优劣主要考虑航迹长度和航迹威胁代价两个因素。结合 D* 算法的原始代价函数,本文提出的代价函数如式(3)所示。

$$f(n)=\sum_{i=1}^{n}(\omega_1\times l_i+\omega_2\times f_i)+\omega_3\times d(n) \tag{3}$$

其中:等式右边第一项为实际代价部分;$l_i$ 为节点 $i-1$ 到 $i$ 之间的航迹长度;$f_i$ 为节点 $i-1$ 到 $i$ 之间受到的威胁概率值;$\omega_1$、$\omega_2$ 为相应权重系数;第二项为启发函数部分;$d(n)$ 为当前节点 $n$ 到目标点的预估航迹长度;$\omega_3$ 为相应权重系数。

### 2.5 引导点的切换

引导点的作用是帮助无人机在满足方向约束的同时更高效地到达目标点。引导点不是一成不变的,而是根据一定策略适时合理切换。本文采用距离作为引导点切换的标准,当航迹扩展节点与某个引导点的距离小于某一阈值时,自动切换为下一个引导点继续引导。当前航迹扩展节点的子节点将以下一个引导点为方向指引进行节点扩展,直至最终到达目标点。

引导点的切换阈值过大或过小都将对规划航迹产生很大影响。切换阈值过大,容易导致引导点的引导力度不足,同时容易发生多个引导点都在切换阈值范围内的情况,增大系统开销;切换阈值过小,容易导致僵化引导,使规划出的航迹丧失最优性。因此,切换阈值的合理选择是最优航迹成功规划的保证。针对本文仿真模型,经多次实验得出引导点的切换阈值设为三个最小步长时,规划性能较优,本文第 3 章将列出不同切换阈值的实验结果对比。当切换阈值为三个最小步长时,假设需要经过 $n$ 次节点扩展才能到达目标点,且每次节点扩展数为 $M$ 个,对扩展出的每个节点都需要进行阈值判断,则在引导点切换部分的算法复杂度可表示为 $T(n\times M)$。由于文中 $M$ 为常数,取值为 7,所以该部分的算法复杂度为 $O(n)$。引导点的切换示意图如图 6 所示。

图 6 中,从起始点向目标点进行路径规划,$N_0$ 为扩展点,其首先向着引导点 $D_3$ 进行节点扩展,$N_1\sim N_4$ 为节点 $N_0$ 的子节点。节点 $N_3$、$N_4$ 与引导点 $D_3$ 的距离小于距离阈值 $L$,故将其引导点切换为 $D_1$(节点 $N_1$ 与 $N_2$ 的引导点仍为 $D_3$),其后续子节点会朝着引导点 $D_1$ 继续扩展。按此切换方法依次进行,直至最终到达目标点。

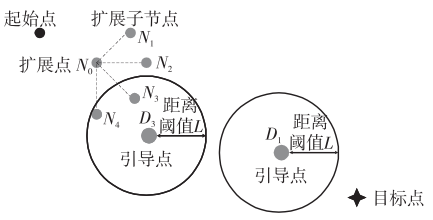



图 6 引导点切换示意图

### 2.6 改进算法航迹规划流程

a)导入数字概率地图;

b)初始化各参数,如最小步长 $L_{min}$、最大扩展节点数 $M$、最大转弯角 $\theta$ 等;

c)输入起始点和目标点的坐标;

d)对目标点基于圆拓展的方式进行障碍检测,确定引导点位置;

e)通过比较代价值函数 $f_{total}$ 确定目标引导点;

f)由起始点向目标引导点进行节点扩展,并将当前点指针指向起始点,将起始点放入 CLOSED 表中;

g)判断当前点指针的节点信息是否为目标点,若是,则由 CLOSED 表的节点信息回溯生成航迹,否则转 h);

h)判断当前点与引导点间的距离是否小于阈值 $L$,若是则进行引导点切换,否则继续向当前引导点进行节点拓展;

i)计算当前点代价值,筛选满足约束条件的节点放入 OPEN 表中;

j)选取 OPEN 表中 $f(n)$ 值最小的节点,将当前指针指向它,并将其放入 CLOSED 表中,转 g)。

算法流程框图如图 7 所示。

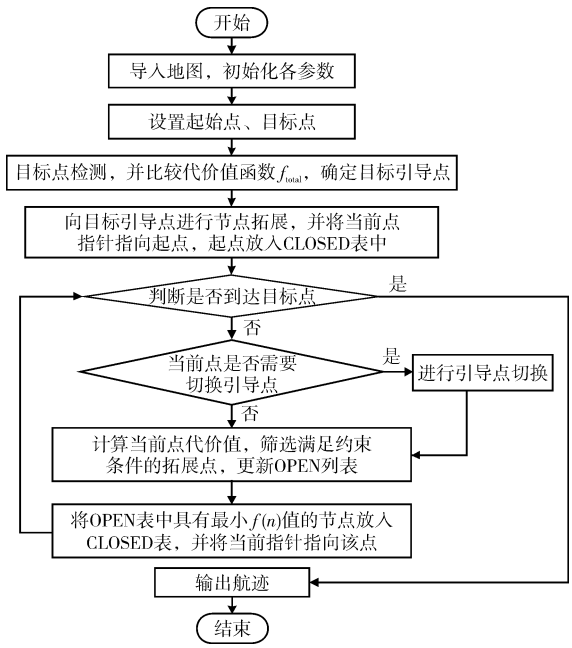



图 7 结合引导点的 D* 算法流程

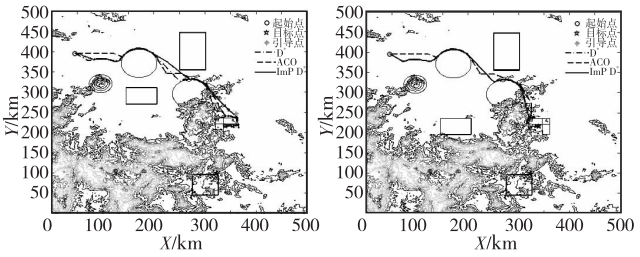
## 3 实验仿真结果与分析

本文在 Intel Xeon CPU E5-2603 v3, 1.6 GHz,8 GB 内存的 PC 机上进行仿真实验,运行环境为 Windows 7 64 位操作系统,编程环境为 MATLAB R2012b。实验使用 500 km × 500 km 的数字高程地图,无人机最大转弯角为 60°,最小步长 $L_{min}$ = 5 km,最大扩展节点数为 7,航迹评估函数的权重系数 $\omega_1$、$\omega_2$、$\omega_3$ 分别为 0.001、300、0.1。仿真实验分为静态规划和动态规划两部分。在规划空间中分别测试目标静止与目标移动时使用 D* 算法、蚁群算法(ACO)、结合引导点的改进 D* 算法(improved D*,Imp D*)三种不同的方法进行规划对比,并使用规划用时、威胁代价、航迹长度、航迹总代价四个性能指标来评价算法。

静态规划中,目标点位置不发生改变,在目标点周围设置方向约束,比较三种不同算法的航迹规划结果。实验分为两组,使无人机从不同的方向接近目标点,满足不同的方向约束。第一组实验起始点坐标为(46,396),目标点坐标为(341,216),实验结果如图 8 所示。为了验证算法的有效性,增加一组对比实验,改变起始点与目标点坐标,并更换数字地图,进行规划对比。第二组实验起始点坐标为(71,196),目标点坐标为(366,366),实验仿真结果如图 9 所示。

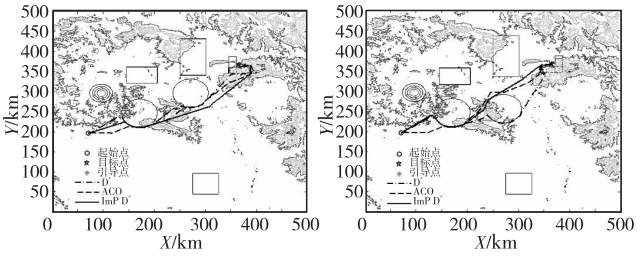
图 8、9 中规则的封闭图形为各种威胁所在区域。左边的圆锥形区域采用多变量高斯分布模型,威胁辐射整个规划空间,离其越近,威胁越大;其余封闭图形均采用均匀分布模型,

威胁只在封闭区域内有值。图中点划线为 D* 算法规划航迹;虚线为 ACO 算法规划航迹;实线为结合引导点的改进 D* 算法规划所得航迹。由图可直观地看到,D* 算法在向目标点扩展节点的过程中,在遇到威胁时再寻找可行路径,绕过威胁,因此规划航程最长;ACO 算法相对于 D* 算法规划所得航迹略有缩短;Imp D* 算法规划所得航迹航程最短。




(a)UAV从目标点的右下角接近　　(b)UAV从目标点的左下角接近

图 8　三种不同算法第一组实验规划航迹对比




(a)UAV从目标点的右上角接近　　(b)UAV从目标点的左上角接近

图 9　三种不同算法第二组实验规划航迹对比

表 1 列出静态环境下两组实验中四种环境情况下的 D*、ACO、Imp D* 算法航迹规划具体实验数据。由表 1 中数据可知,当目标点周围存在方向约束,要求无人机从特定方向进入目标点时,D* 算法规划航迹航程最长,同时对威胁不敏感,其规划所得航迹在遇到威胁之后再寻找可行节点,绕过威胁,航迹总代价最高;ACO 算法相对于 D* 算法其规划所得航迹航程缩短,航迹总代价降低,但规划时间最长;Imp D* 算法由于引导点的作用,其具有提前规避威胁的能力,规划所得航迹长度最短,航迹总代价最小。在两组实验的(b)情形中,D*、ACO 算法在接近目标点附近时,算法节点扩展不断权衡威胁代价与航程代价两个因素的优劣,希望达到航迹总代价最小的目的,所以耗时较为严重;而 Imp D* 算法由于引导点的存在,大大减少算法为满足方向约束在目标点附近进行的节点扩展次数,规划时间有效缩短。

表 1　静态环境下三种算法性能对比

| 规划环境 | 算法 | 规划用时/s | 威胁代价 | 航迹总长/km | 航迹总代价 |
|---|---|---|---|---|---|
| 图 8(a) | D* | 19.914 7 | 0.007 4 | 515.563 6 | 2.728 4 |
| | ACO | 33.565 2 | 0.007 2 | 505.863 6 | 2.665 8 |
| | Imp D* | 2.997 5 | 0.006 8 | 492.388 5 | 2.532 3 |
| 图 8(b) | D* | 193.292 7 | 0.007 4 | 504.590 0 | 2.717 4 |
| | ACO | 208.572 9 | 0.007 1 | 477.632 6 | 2.607 6 |
| | Imp D* | 4.869 8 | 0.007 1 | 450.549 1 | 2.579 5 |
| 图 9(a) | D* | 12.303 5 | 1.1788E-04 | 461.817 1 | 0.497 2 |
| | ACO | 26.705 2 | 5.8223E-06 | 457.367 8 | 0.459 1 |
| | Imp D* | 1.956 9 | 1.2059E-07 | 455.066 2 | 0.455 1 |
| 图 9(b) | D* | 189.813 4 | 1.1788E-04 | 472.319 9 | 0.507 7 |
| | ACO | 202.317 1 | 3.4966E-04 | 396.874 6 | 0.501 8 |
| | Imp D* | 2.614 0 | 3.0632E-04 | 387.340 2 | 0.479 2 |

上述仿真结果中,引导点的切换阈值均采用三个最小步长。经多次实验验证,当切换阈值为 $3L_{min}$ 时,规划效果较优。取静态环境下第一组实验中的(a)情形进行不同切换阈值的航迹规划结果对比。切换阈值选择 $1L_{min}$、$2L_{min}$、$3L_{min}$、$4L_{min}$ 进行仿真实验,结果如图 10 所示。

由图 10 可看出,当切换阈值为 $1L_{min}$ 时,由于切换阈值过小,航迹显得过于僵化,规划所得航迹航程较长;当切换阈值为 $4L_{min}$ 时,由于切换阈值过大,致使引导点的引导作用没有充分发挥,规划航迹过早收敛于遮挡物附近;当切换阈值为 $2L_{min}$、$3L_{min}$ 时,规划所得航迹差异不大,航程较短。具体实验数据如表 2 所示。

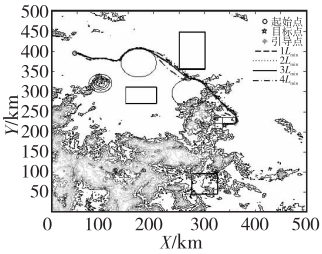


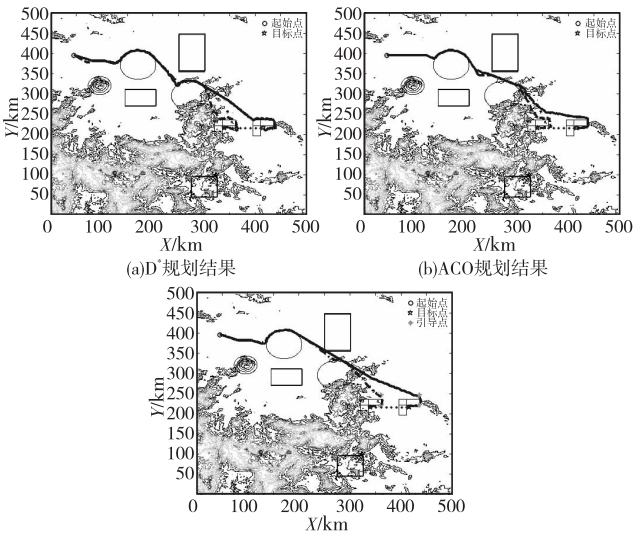

图 10　不同切换阈值的规划航迹对比

表 2　不同切换阈值的实验结果

| 切换阈值 | 规划用时/s | 威胁代价 | 航迹长度/km | 航迹总代价 |
|---|---|---|---|---|
| $1L_{min}$ | 3.753 3 | 0.006 8 | 500.960 1 | 2.540 9 |
| $2L_{min}$ | 3.084 4 | 0.006 8 | 495.352 7 | 2.535 4 |
| $3L_{min}$ | 2.997 5 | 0.006 8 | 492.388 5 | 2.532 3 |
| $4L_{min}$ | 3.790 3 | 0.007 6 | 505.844 1 | 2.785 8 |

由表 2 中数据可知,引导点的切换阈值过大或过小都将对规划航迹产生较大影响,切换阈值过小使规划航程变长,切换阈值过大使航迹代价升高。因此,应结合实际约束情况,合理选择切换阈值。

动态规划中目标点位置发生改变。为了更好地对比算法效果,设目标点沿某一固定方向进行移动。取图 8(a)情形进行动态仿真实验,要求无人机从目标点的右下角进入目标点。起始点位置不变仍为(46,396),目标点位置由原先坐标(341,216)沿某一固定方向移动到(416,216)。假定无人机的飞行速度是目标点移动速度的 5 倍,即目标点每移动一个步长,无人机移动五个航迹段长度。仿真结果如图 11 所示。图 11 中虚线为三种算法规划所得初始航迹;实线为动态环境下无人机实际飞行航迹;点线为目标移动轨迹。由图可直观看到,D* 算法规划所得航迹较为曲折,航程较远;ACO 算法规划航迹对目标移动变化不敏感,但规划航迹能较好地避开威胁区;Imp D* 算法结合引导点,其规划所得航迹最平滑,且能有效避开威胁区域,同时航程最短。具体实验数据如表 3 所示。




(a)D*规划结果　　(b)ACO规划结果

(c)Imp D*规划结果

图 11　动态环境下三种算法规划航迹对比

表 3　动态环境下三种算法性能对比

| 算法 | 规划用时/s | 威胁代价 | 航迹长度/km | 航迹总代价 |
|---|---|---|---|---|
| D* | 25.693 7 | 0.007 2 | 567.126 8 | 2.727 1 |
| ACO | 276.341 2 | 0.006 8 | 565.238 9 | 2.605 2 |
| Imp D* | 4.793 9 | 0.006 3 | 550.732 8 | 2.440 7 |

由表 3 可知,动态环境下,ACO 算法规划用时最长,难以满足航迹实时更新的要求,但其规划所得航迹总代价相比于 D* 算法有所降低;D* 算法规划时间缩短,但其规划所得航迹航程最长,航迹代价最高;Imp D* 算法结合引导点,在动态环境中规划时间大大缩短,同时可有效降低航迹代价,缩短航程。

(下转第 1988 页)

那么($A_{\text{low}(k')-1} \cup D_i^{k'}$, $B_{\text{low}(k')-1} \cup \{k'\}$)一定能控制住{1, 2,…, low($k'$) −1}。

由定理3,对任意的 $k''(k' < k'' \leqslant i)$,若 $f(k'') = 0$,则 $k'$ 与 $k''$ 一定相邻,$D_i^{k'} \subseteq A_i$。

$$w(A_i \backslash D_i^{k'}, B_i \cup \{k'\}) = w(A_i, B_i) - |D_i^{k'}| - 2$$

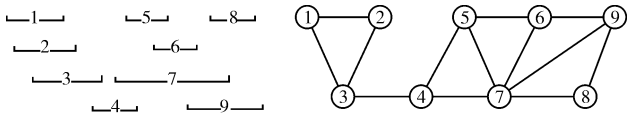
$$w(A_i, B_i) - |D_i^{k'}| - 2 \geqslant w(A_{\text{low}(k')-1}, B_{\text{low}(k')-1})$$

即 $w(A_i, B_i) \geqslant w(A_{\text{low}(k')-1}, B_{\text{low}(k'')-1}) + |D_i^{k'}| + 2 \geqslant w_3$ (4)

由式(3)(4)知 $w(A_i, B_i) = w_3$,证毕。

## 2 算法示例

为了能更清晰地说明算法运行的过程,本文举出一个九个点的区间图示例,如图1所示。




(a) 区间图$G$的对应区间 (b) 区间图$G$

图1 九个顶点的区间图 $G$

示例的算法过程中运行结果及步骤如下:

low(1) =1, low(2) =1, low(3) =1, low(4) =3,
low(5) =4, low(6) =5, low(7) =4,
low(8) =7, low(9) =6;
max low(1) =1, max low(2) =1, max low(3) =1,
max low(4) =3, max low(5) =4, max low(6) =5,
max low(7) =5, max low(8) =7, max low(9) =7;
$L_1 = \{1\}, M_1 = \{2,3\}; L_2 = \{1,2\}, M_2 = \{3\};$
$L_3 = \{1,2,3\}, M_3 = \{4\}; L_4 = \{3,4\}, M_4 = \{5,7\};$
$L_5 = \{4,5\}, M_5 = \{6,7\}; L_6 = \{5,6\}, M_6 = \{7,9\};$
$L_7 = \{5,6,7\}, M_7 = \{8,9\}; L_8 = \{7,8\}, M_8 = \{9\};$
$L_9 = \{7,8,9\}, M_9 = \varnothing;$
$A_0 = \varnothing, B_0 = \varnothing;$
$i=1, w_1=1, w_2=2, A_1 = \{1\}, B_1 = \varnothing, w=1,$
$i=2, w_1=2, w_2=3, A_2 = \varnothing, B_2 = \{1\}, w=2,$
$i=3, w_1=3, w_2=2, A_3 = \varnothing, B_3 = \{3\}, w=2,$
$i=4, w_1=3, w_2=2, A_4 = \varnothing, B_4 = \{3\}, w=2,$
$i=5, w_1=3, w_2=4, A_5 = \{5\}, B_5 = \{3\}, w=3,$
$i=6, w_1=4, w_2=4, A_6 = \varnothing, B_6 = \{3,5\}, w=4,$
$i=7, w_1=5, w_2=4, A_7 = \varnothing, B_7 = \{3,5\}, w=4,$
$i=8, w_1=5, w_2=4, A_8 = \varnothing, B_8 = \{3,7\}, w=4,$
$i=9, w_1=5, w_2=4, A_9 = \varnothing, B_9 = \{3,7\}, w=4。$

所以,该区间图的罗马控制函数是 $V_1 = \varnothing, V_2 = \{3,7\}, w=4$。

## 3 结束语

对区间图提出了一种求罗马控制集和控制数的动态规划的新算法,采取逐步搜索的策略,迅速而有效地解决了区间图的罗马控制染色问题,然后进行了严格的数学推理和证明,最后给出了示例演示。结果表明该算法不仅运算快捷,而且简便有效。

**参考文献:**


[1] Stewart I. Defend the Roman empire! [J]. Scientific American, 1999, 281(6): 136-138.
[2] Cockayne E J, Dreyer P A, Hedetniemi S M, *et al*. Roman domination in graphs[J]. Discrete Mathematics, 2004, 278(1-3): 11-22.
[3] Liedloff M, Kloks T, Liu Jiping, *et al*. Efficient algorithms for Roman domination on some classes of graphs[J]. Discrete Applied Mathe-matics, 2008, 156(18): 3400-3415.
[4] Habib M, McConnell R, Paul C, *et al*. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing[J]. Theoretical Computer Science, 2000, 234(1-2): 59-84.
[5] Zhang Peisen, Schon E A, Fischer S G, *et al*. An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA [J]. Bioinformatics, 1994, 10(3): 309-317.
[6] Booth K S, Johnson J H. Dominating sets in chordal graphs[J]. SIAM Journal on Computing, 1982, 11(1): 191-199.
[7] Farber M. Independent domination in chordal graphs[J]. Operations Research Letters, 1982, 1(4): 134-138.
[8] Yang Hong. General vertex-distinguishing total coloring of complete bipartite graphs[J]. ARS Combinatoria, 2016, 125: 371-379.
[9] 罗茜. 图的集控制与罗马控制[D]. 南昌: 华东交通大学, 2012.
[10] Ramalingam G, Rangan C P. A unified approach to domination problems on interval graphs[J]. Information Processing Letters, 1988, 27(5): 271-274.
[11] 皮军德, 康丽英, 许光俊. 直线簇上区间图的最小独立控制集[J]. 运筹学学报, 2006, 10(1): 107-115.
[12] 李鹏. 区间图相关图类若干结构与算法问题[D]. 上海: 上海交通大学, 2014.
[13] Raychaudhuri A. On powers of interval and unit interval graphs[J]. Congressus Numerantium, 1987, 59: 235-242.
[14] Chang G J. Algorithmic aspects of domination in graphs[M]//Handbook of Combinatorial Optimization. New York: Springer, 1998: 1811-1877.


(上接第 1985 页)

## 4 结束语

随着无人机应用环境的日益复杂,研究有方向约束的航迹规划问题具有重要意义。本文提出了一种在目标点周围设置引导点的改进航迹规划方法,通过在目标点周围设置合适的引导点,将引导点与 D* 算法相结合设计一种改进航迹规划算法。在满足接近目标的方向约束的同时,新的规划方法结合引导点,可以大大加快航迹规划速度,降低航迹规划代价,更好的满足实时性要求。同时也应该注意到,在引导点的产生阶段,由于需要预先获得圆拓展半径,而半径大小的合适与否会影响生成的引导点的优劣。结合引导点的改进算法虽然可以满足方向约束,但引导点设置不当,也会使规划航迹代价增大,散失最优性。尽管如此,但对于复杂多变的实际飞行环境,要求无人机从特定方向进入目标点时,结合引导点的改进算法可有效缩短规划时间和规划航程。无人机飞行时间、距离缩短,相应地减少了突发情况影响其完成任务的成功率。因此,在实际应用中应根据具体任务要求,合理设置引导点。

**参考文献:**


[1] 周青, 张锐, 索晓杰, 等. 具有时间约束的无人机遗传算法航迹规划[J]. 航空计算技术, 2016, 46(2): 93-96.
[2] 张帅, 李学仁, 张鹏, 等. 基于改进 A* 算法的无人机航迹规划[J]. 飞行力学, 2016, 34(3): 39-43.
[3] Huptych M, Röck S. Online path planning in dynamic environments using the curve shortening flow method[J]. Production Engineering, 2015, 9(5-6): 613-621.
[4] Pehlivanoglu Y V. A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV[J]. Aerospace Science & Technology, 2011, 16(1): 47-55.
[5] Szczerba R J, Galkowski P, Glicktein I S, *et al*. Robust algorithm for real-time route planning[J]. IEEE Trans on Aerospace & Electronic Systems, 2000, 36(3): 869-878.
[6] 唐晓东, 吴静. 基于改进 A* 算法的三维航迹规划技术研究[J]. 电子技术应用, 2015, 41(5): 163-166.
[7] 刘新, 周成平, 俞琪, 等. 基于分层策略的三维航迹快速规划方法[J]. 宇航学报, 2010, 31(11): 2524-2529.
[8] 刘琼昕, 王景, 高春晓, 等. 基于引导点的无人机三维航迹规划方法[J]. 北京理工大学学报, 2014,34(11): 1163-1168.
[9] 杨杰, 蔡超, 孙希霞. 一种具有端点方向约束的快速航迹规划方法[J]. 计算机工程, 2014, 40(2): 11-15.
[10] 魏铁涛, 王剑薇, 屈香菊. 动态环境下在线航迹规划的滚动优化方法[J]. 飞行力学, 2012, 30(3): 218-222.
[11] Chen Yongbo, Yu Jianqiao, Mei Yuesong, *et al*. Modified central force optimization (MCFO) algorithm for 3D UAV path planning[J]. Neurocomputing, 2016, 171(1): 878-888.
[12] Wen Naifeng, Su Xiaohong, Ma Peijun, *et al*. Online UAV path planning in uncertain and hostile environments[J]. International Journal of Machine Learning & Cybernetics, 2015,8(2): 1-19.
[13] 王绪芝, 姚敏, 赵敏, 等. 基于蚁群算法的无人机航迹规划及其动态仿真[J]. 指挥控制与仿真, 2012, 34(1): 29-32.
[14] Saranya C, Unnikrishnan M, Ali S A, *et al*. Terrain based D* algorithm for path planning[J]. IFAC-Papers OnLine, 2016, 49(1): 178-182.
[15] 陈侠, 刘冬. 应用 D*Lite 算法的目标移动时无人机三维航迹规划[J]. 电光与控制, 2013, 20(7): 1-5.
[16] Al-Mutib K, Alsulaiman M, Emaduddin M, *et al*. D* lite based real-time multi-agent path planning in dynamic environments[C]// Proc of the 3rd International Conference on Computational Intelligence, Modelling and Simulation. 2012: 170-174.