doi:10.3969/j.issn.1671-637X.2014.07.008

改进A*算法的多约束航迹规划

李世晓[1], 朱 凡[1], 张 健[1], 刘 杰[1], 隋晓奎[2]
(1.空军工程大学航空航天工程学院,西安 710038; 2.哈尔滨飞行学院理论训练系,哈尔滨 150001)

摘 要:针对当前航迹规划的研究偏于理论、规划效率不高的问题,从工程应用出发,分析了航迹规划的多种约束条件,提出了一种多约束条件下的快速航迹规划方法,设计了基于航向角有限离散变化思想的工程化搜索策略;改进了A*算法流程,采用结构体链表式的最小二叉堆技术管理OPEN表和CLOSE表,显著提高算法的规划效率;提出了删除航路点的航迹优化方法,对传统A*算法和改进的算法分别进行了仿真。结果表明,在多约束条件下,该方法显著提高了规划效率。

关键词:无人机;航路优化;多约束;A*算法;搜索策略;最小二叉堆
中图分类号:V249.4;TP391.9 **文献标志码**:A **文章编号**:1671-637X(2014)07-0036-05

Multi-Restriction Path Planning Based on Improved A* Algorithm

LI Shi-xiao[1], ZHU Fan[1], ZHANG Jian[1], LIU Jie[1], SUI Xiao-kui[2]
(1. Engineering College of Aeronautics and Astronautics, Air Force Engineering University, Xi'an 710038, China;
2. Theory Training Department, Aviation Flight College of Harbin, Harbin 150001, China)

Abstract: Considering that the path planning research generally focused on theoretical study and the plan efficiency is not high, we analyzed the multiple restrictions for path planning from the point of engineering application, and proposed a rapid path planning method under multiple restrictions. An engineering search strategy was designed based on the limited discrete changing idea of course angle. The flow path of A* algorithm was improved. The minimum binary heap in structure list way was applied to manage the OPEN list and CLOSED list, which improved the search efficiency of the algorithm notably. An optimization method deleting some path nodes was proposed. Simulations were carried out respectively on traditional algorithm and the improved algorithm. The results indicate that the plan efficiency of the later can be improved considerably under the condition of multiple restrictions.

Key words: UAV; route optimization; multi-restriction; A* algorithm; search strategy; minimum binary heap

0 引言

无人机航迹规划是综合考虑到达时间、油耗、防空威胁及飞行区域等因素,为无人机完成某种飞行任务规划出最优或是满意可行的飞行航迹[1]。目前,研究用于航迹规划的算法很多,如最优式算法[2],启发式算法和随机型算法。A*算法作为启发式算法的代表,于1971年由N. Nillson提出[3],一经提出就广泛应用于路径规划。目前,为满足不同要求,研究人员已对A*算法做了诸多改进。Szczerba等人提出稀疏A*搜索算法(SAS),通过将约束条件与搜索算法结合起来,有效地剪裁搜索空间[4]。文献[5]在SAS的基础上,通过考虑最大爬升下滑角,将航迹搜索扩展至三维空间,提出采用变步长搜索策略,提高规划效率,但变步长机制过于复杂,考虑的环境过于理想,离工程应用还有一定差距。文献[6]提出改进A*算法数据结,用有序二叉树管理OPEN表和CLOSE表,提高搜索效率,但没有考虑表的动态性可能会导致树的极度不平衡,从而丧失效率优势。本文从工程应用出发,分析并处理了航迹规划的多种约束条件,基于航向角有限离散变化

收稿日期:2013-08-07 修回日期:2013-09-14
作者简介:李世晓(1987—),男,河北辛集人,硕士生,研究方向为系统建模理论与仿真技术。

的思想，结合约束条件设计了易于工程实现的航迹搜索策略，通过改进算法流程，使 OPEN 表和 CLOSE 表的维护工作实现部分解耦。采用结构体链表形式的最小二叉堆技术维护 OPEN 表，提高了最优节点的提取速度，也克服了数组二叉堆的容量可能导致 OPEN 表溢出的问题；采用三层嵌套的二叉堆管理 CLOSE 表，有效解决了动态二叉树易出现的极度不平衡问题，保证了节点查找的高效率。

1　航迹规划约束的分析及处理

根据约束来源不同，航迹约束可以分为 4 类。

1）威胁约束。被敌方防空雷达发现的概率 P_{det}，被敌方防空导弹和高炮的击落概率 P_{mis} 和 P_{anti}，撞地概率 P_{col}，最小离地高度 $h_{\min}$，最小横侧向离地间隙 $d_{\min}$。

2）任务战术约束。任务完成时间 t，起始点 S，目标点 T，固定航向角约束，一般包括初始航向角 ψ_{S} 和目标航向角 ψ_{T}。

3）无人机性能约束[7]。最小平飞速度 $V_{\min}$，最大平飞速度 $V_{\max}$，最大航程 $L_{\max}$，最大飞行高度 $H_{\max}$，水平最小转弯半径 $R_{\min}$，最大爬升角 $\theta_{\max}$，最大纵向曲率 $\rho_{\max}$，最大法向过载 $n_{z\max}$。

4）其他约束。禁飞区约束，复杂气象区约束。

为了简化问题的复杂度，采用“约束转化、分步解决”的方式解决。

1）规划前期处理。基于数字地图，采用地形坡度限制平滑算法解决最大爬升角 $\theta_{\max}$ 和最小横侧向离地间隙 $d_{\min}$ 的约束。采用地形曲率限制平滑算法解决最大过载限制约束 $n_{z\max}$[8]。采用安全曲面技术实现对最小离地高度 $h_{\min}$ 的限制，直接在二维平面进行航迹搜索，然后将航迹投射到安全曲面上，通过二维航迹搜索实现了三维航迹规划。

2）A* 算法搜索策略解决。设置合适的规划步长满足水平最小转弯半径 $R_{\min}$ 限制；设置次起始点和虚拟禁飞区来分别实现初始航向角和目标航向角的约束；最小平飞速度 $V_{\min}$ 和最大平飞速度 $V_{\max}$，任务完成时间 t 与航程有着密切关系，可将其转化为最大航程 $L_{\max}$ 的约束，节点扩展时，将满足航程条件作为扩展条件来惩罚那些超出最大航程约束的节点。

3）代价函数的解决。A* 算法的代价函数为

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

式中：$g(n)$ 为初始节点到节点 n 已经实际付出的代价；$h(n)$ 是从节点 n 到目标节点的估计代价，称为启发函数。$g(n)$ 设计为

$$g(n)=w_{\mathrm{T}}T(n)+w_l\cdot l(n) \tag{2}$$

$$T(n)=w_1\cdot P_{\mathrm{det}}+w_2\cdot P_{\mathrm{mis}}+w_3\cdot P_{\mathrm{anti}}+w_4\cdot P_{\mathrm{col}} \tag{3}$$

式中：$T(n)$ 表示实际付出的威胁代价；$l(n)$ 表示实际的航程代价；w_{T}、w_l、w_1、w_2、w_3、w_4 为权重系数。通过代价函数完成对 P_{det}、P_{mis}、P_{anti}、P_{col} 的处理。

2　搜索策略的设计

A* 算法是一种基于栅格的启发式搜索算法。作战区域栅格的划分是首先要解决的问题。理论上作战区域可以任意划分，为研究方便，本文采用正方形网格规则划分。传统的扩展方式是不断扩展临近所有节点，需要较长的时间和较大的内存，本文借鉴稀疏 A* 算法的思想[4,9]，采用基于航向角有限离散变化的搜索策略，在节点扩展时，结合无人机机动性约束，对搜索区域进行剪裁，提高搜索效率。为了论述方便，先给出关于搜索策略的两个定义。

定义 1　规划步长：对作战区域进行网格划分的相邻平行直线段之间的距离，即所产生的正方形单元的边长，用 a 表示。

定义 2　搜索步长：航迹搜索时设定的单步最大搜索距离，用 λ 表示。

2.1　结合机动性能约束的搜索策略

无人机在飞行时，受到无人机的机动性能约束，在无侧滑时，$R_{\min}$ 的算式为

$$R_{\min}=V^2/(g\cdot\sqrt{n_{z\max}^2-1}) \tag{4}$$

式中：V 为无人机的飞行速度；$n_{z\max}$ 为无人机的最大法向过载。

λ 与 $R_{\min}$ 的相互关系决定了单步搜索可以达到的飞行区域，在某一规划步长 a 下，无人机在下一步搜索可以飞抵的区域如图 1 阴影部分所示。显然，情形 a 扩展的节点最少，节点扩展的逻辑方式也最简单，裁剪搜索空间的效果最好，最有利于提高算法搜索效率，故本文选取情形 a，即选取 $\lambda<2R_{\min}$。

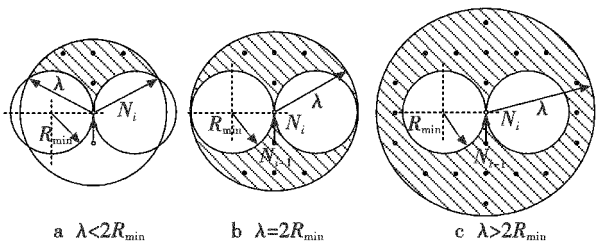

图 1　单步搜索无人机的飞抵区域

Fig. 1　The flying area of UAV in single step search

规划步长 a 的大小决定了网格划分的精细程度，进而影响计算量和所需内存。为了兼顾航迹精度和运算效率，本文选用满足图 2 所示的几何关系的一组 a、λ 和 $R_{\min}$，下面定量分析三者之间的大小关系。方式 a 和方式 b 均扩展 5 个子节点，方式 c 扩展 4 个子节点。考虑扩展的节点时，以航迹方向角的改变量为节点扩

展的依据,每个可以扩展的方向上选择一个节点扩展。方式a中,保证N_{i+1}^1被扩展即可保证5个节点均可被扩展,必要条件为

$$\begin{cases}\sqrt{8}\cdot a\leqslant\lambda\\ 2l_0\geqslant R_{\min}\end{cases}。\tag{5}$$

在方式a的基础上,考虑方式b,只需保证N_{i+1}^1被扩展即可,必要条件为

$$\begin{cases}3a\leqslant\lambda\\ 3a\geqslant\sqrt{2}R_{\min}\end{cases}。\tag{6}$$

下面考虑方式c,结合方式a和方式b的限制要求,需保证N_{i+1}^1和N_{i+1}^4可以被扩展,必要条件分别为

$$\begin{cases}\dfrac{2}{\sqrt{5}}R_{\min}\leqslant 2a\\ 2a\leqslant\lambda\end{cases}\tag{7}$$

$$\left[\frac{2(\sqrt{5}a-R_{\min})}{\sqrt{5}}\right]^2+\left[\frac{2a-(\sqrt{5}a-R_{\min})}{\sqrt{5}}\right]^2\geqslant R_{\min}\tag{8}$$

又$\lambda<2R_{\min}$,综合式(5)~式(8)可得a、λ和$R_{\min}$之间的大小关系为

$$\frac{6\sqrt{5}}{25}R_{\min}\leqslant a\leqslant\frac{1}{3}\lambda<\frac{2}{3}R_{\min}。\tag{9}$$

采用满足式(9)的规划步长a和搜索步长λ即可保证在进行节点扩展时就考虑了无人机的最小转弯半径约束,最终产生的航迹是一系列节点顺次连接而形成的,相邻航迹段之间航向角的变化由节点的扩展方式决定。采用图2的节点扩展方式,节点的航向角可取如图3所示的标号为0~15共16种离散值。

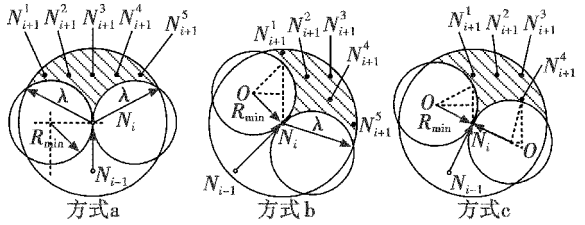

图2 节点扩展示意图

Fig. 2 The schematic diagram of node extension

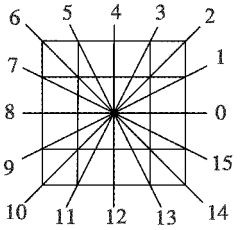

图3 节点的离散航向角

Fig. 3 The discrete course angle of node

2.2 结合固定航向角约束的搜索策略

对固定航向角的处理通常是通过添加虚拟威胁或虚拟禁飞区,利用算法搜索中规避机制来实现的。这种方法只能逼近航向角,且设置较为繁琐,并增加了算法的负担。本文结合搜索策略中的航向角有限离散变化的思想,对初始航向角的约束提出一种生成次起始点的方法,设置简便,可以准确实现且不会增加算法负担。原理如图4所示。

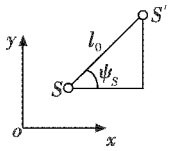

图4 次起始点生成示意图

Fig. 4 The schematic diagram of sub-start node generating

设起始点为$S(x_S,y_S)$,要求以ψ_S飞离起始点,则可以根据几何关系,生成次起始点$S'(x_{S'},y_{S'})$,然后以S'为起始点,进行航迹规划,再连接SS',作为航迹的第一段,这样可以满足初始航向角约束。其坐标关系为

$$\begin{cases}x_{S'}=x_S+l_0\cdot\cos\psi_S\\ y_{S'}=y_S+l_0\cdot\sin\psi_S\end{cases}。\tag{10}$$

A*算法是单向从起始点顺序扩展至目标点的,无法保证次目标点后满足飞机性能约束,该方法对目标航向角约束不适用。对目标航向角约束采用设置虚拟禁飞区的方法。

3 A*算法的改进

A*算法需要设置OPEN表和CLOSE表,OPEN表用来存储已经被计算但还没有被扩展的节点,CLOSE表用来存储已经被扩展的节点。搜索过程如下[6]:

1) 清空OPEN表和CLOSE表;

2) 将起点StartNode放入OPEN表;

3) 选取OPEN表中最优(f最小)节点BestNode放入CLOSE表;

4) 若BestNode是目标点,规划完成,退出;

5) 扩展BestNode,产生子节点Child,建立从Child指向BestNode的指针;

6) 若Child与OPEN表中节点Node相同,比较两点的g值,若$g_{cld}<g_{Node}$,将Node的父节点设置为Child,返回3);

7) 若Child与CLOSE表中节点Node相同,比较两点的g值,若$g_{cld}<g_{Node}$,将Node的父节点设置为Child,返回3);

8) 将Child放入OPEN表中,返回3)。

航迹规划中A*算法一般采用数组式最小二叉堆和单向链表分别维护OPEN表和CLOSE表,数组式二叉堆在步骤3)中查找最佳节点效率较高,但存在数组容量有限易导致OPEN表溢出的问题。单向链表在步骤7)查找重复节点效率也较低。文献[6]提出的有序

二叉树查找效率与树的形状有关，而树的形状与构建二叉树时数据的插入顺序有关。A* 算法两张表都是动态变化的，插入的节点数据有较强随机性，用二叉树维护 CLOSE 表查找效率也有较强的随机性。虽然 DSW 算法和 AVL 树可以是树恢复平衡[10]，但算法较复杂，一般适用于静态树的平衡。

3.1　算法流程的改进

从算法的搜索过程上看，OPEN 表和 CLOSE 表的维护是算法的关键，直接影响到算法的效率。两个表的维护均包含节点的插入和相同节点的查找，OPEN 表还包含最优节点的选择和提取。本文借鉴文献[6]的思想，从算法流程和数据结构两个方面进行算法改进。在上述步骤 2）和 8）中，将节点放入 OPEN 表的同时也放入到 CLOSE 表中，步骤 3）选出的节点不再放入 CLOSE 表，这样 CLOSE 表包含了 OPEN 表的全部节点，可以省去步骤 5）。改进后，算法的原理本质没有改变，但两张表的维护工作实现部分解耦，OPEN 表主要负责最优节点的选择和提取，CLOSE 表主要完成相同节点查找。

3.2　最小二叉堆

最小二叉堆是一种特殊类型的二叉树，它具有如下两个性质[10]：1）每个节点的值小于或等于其每个子节点的值；2）该树完全平衡，最后一层的叶子都处于最左侧的位置。

没有父节点只有子节点的节点叫根，没有子节点的节点叫叶子。在最小二叉堆中，插入元素和提取元素都需要恢复堆的性质。性质恢复算法的伪代码如下。

```
heapEnqueue(val)  // 插入元素
  {将 val 放在堆的末尾;
  while val 不在根部且 val > parent(val);
  交换 val 与它的父节点;}
heapDequeue( )  // 提取元素
  {从根节点中提取元素;
  将最后一个叶子节点中的元素放在根节点中;
  删除最后一个叶子节点;
  P = 根节点;
  while P 非叶子节点,且 P > 它的任何子节点;
  交换 P 与其较小的子节点;}
```

查找节点时，按照广度优先法进行遍历，即从最底层向下逐层从左到右访问每个节点，若遇到小于待查节点值的节点，则停止对该节点子节点的遍历，直至找到相同节点或是遍历结束。二叉堆的查找效率虽然低于平衡性好的有序二叉树，但远高于单向链表。

3.3　OPEN 表和 CLOSE 表的数据结构

改进流程后，OPEN 表的维护本质上属于优先队列问题。堆是实现优先队列的极好方法，依据节点 f 值的大小建立最小二叉堆，根节点值最小即为最优节点。提取和插入节点按照 3.2 节所示的算法，解决了传统单向链表选择极值效率低的缺点。采用结构体链表形式实现最小二叉堆，解决了传统数组式容量有限、易导致 OPEN 表溢出的问题。

改进流程后，CLOSE 表的维护主要为相同节点的查找。根据搜索策略，航迹节点可以由 (x,y,ψ) 唯一确定，本文分层查找策略，基于最小二叉堆构建三层嵌套结构，即顶层依据 ψ 的大小建立二叉堆 *Heap_ψ*。*Heap_ψ* 的每个元素内嵌一个以 x 大小为依据建立的子堆 *Heap_x*。*Heap_x* 每个单元再内嵌一个以 y 为依据的子堆 *Heap_y*。

插入节点时，先依据 ψ 值查找 *Heap_ψ*：若不存在对应的 ψ 值，则直接将节点的 ψ 值插入 *Heap_ψ*；若存在对应的 ψ 值，则在该 ψ 值嵌套下的 *Heap_x* 查找待插入节点的 x 值。同理处理 *Heap_x* 和 *Heap_y*：查找重复节点时，先根据 ψ 值查找 *Heap_ψ*，若存在相同的 ψ 值，便在该 ψ 值嵌套下的 *Heap_x* 中查找 x 值；同理，再查找 *Heap_y* 直至找到节点或无法找到而返回。

4　航迹优化

A* 算法规划出来的航迹存在航路点过于密集和转弯频繁的情况，这势必会增大导航设备的误差及增加飞控系统的计算负担，因此有必要对航迹进行优化处理。本文通过删除一系列中间节点的方法，使得航路更加优化。对航迹点进行连续编号，用 N_i 表示，$i=1,2,3,\cdots$。用 g_{AB} 表示航迹段 AB 的代价，如果 $g_{N_{i-1}N_i}+g_{N_iN_{i+1}}<g_{N_{i-1}N_{i+1}}$，则称 N_i 点可删除。从起点检测各点的可删除性，删除可删除点，保留不可删除点，直至到达目标点。优化流程如图 5 所示，经过此法航迹将得到较好的优化。

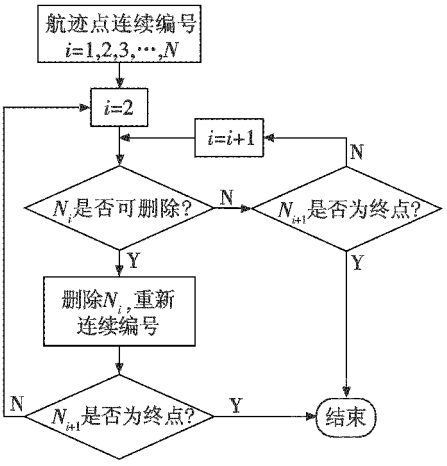

图 5　优化流程图

Fig. 5　The flow chart of optimization

5　仿真算例

仿真计算机使用 Intel(R) Core(TM) i3 处理器，主频为 2.93 GHz，内存为 2 GB。规划算法采用 Microsoft

Visual C++ 6.0 编程技术实现，使用范围为 E117° ~ E118° N23° ~ N24°的真实数字地图和模拟生成的威胁信息，起始点 E117.14° N23.18°，目标点 E117.85° N23.83°，其他主要仿真参数见表1。

表1 主要仿真参数

Table 1 Main parameters of simulation

R_{min}/m	λ/m	α/m	ψ_S/(°)	ψ_T/(°)	H_{max}/km	h_{min}/m	d_{min}/m
1000	1900	600	0	165 ~ 195	2	200	100

分别采用传统的稀疏 A* 算法和本文提出的改进 A* 算法进行仿真，二者只是算法流程和数据结构不同，其他仿真参数和条件完全相同。通过仿真都可以得到如图6的仿真结果，航迹优化后如图7所示。二者航迹规划所用的时间见表2。

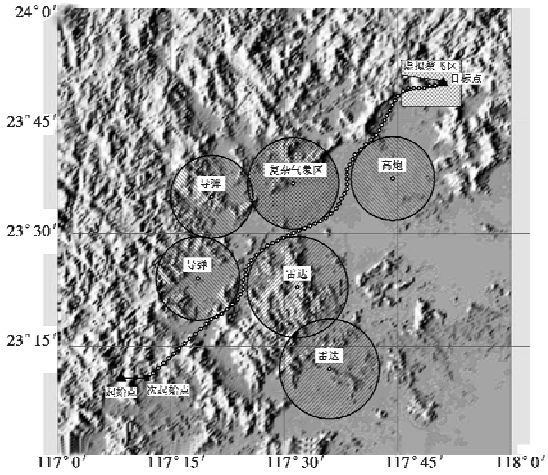

图6 仿真效果图

Fig. 6 The figure of simulation

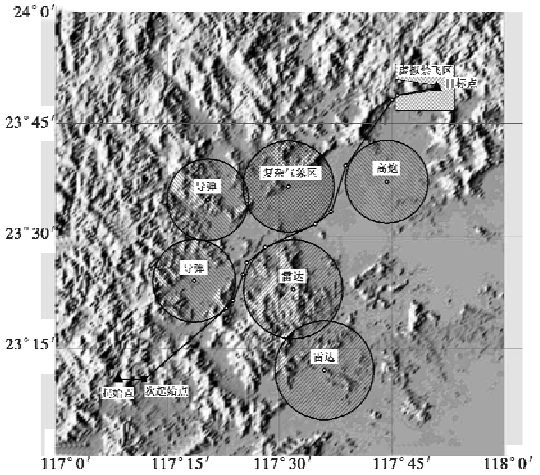

图7 航迹优化图

Fig. 7 The optimized flight path

表2 规划耗时对比

Table 2 The comparison of time-consuming

	OPEN 表	CLOSE 表	耗时/s
传统算法	数组式最小二叉堆	单向链表	68
改进算法	结构体式最小二叉堆	嵌套最小二叉堆	13

采用本文设计的易于工程化的搜索策略，传统 A* 算法和改进后的 A* 算法都可以较好地处理多种约束，得到较好的航迹，顺利到达目标点。在保证规划效果的前提下，在相同的规划环境和规划任务下，从表2可得，本文改进算法的规划效率有了很大提高。

6 结束语

本文从工程应用出发，考虑多种航迹约束，基于改进的 A* 算法，提出了一种易于工程化的快速航迹规划方法。仿真结果表明，该方法可以较好地处理航迹规划中的多种约束条件，显著提高了规划效率，具有较高的工程应用价值。

参 考 文 献

[1] 郑昌文，严平，丁越明，等. 飞行器航迹规划研究现状和趋势[J]. 宇航学报，2007，28(6)：1441-1446.
ZHENG C W, YAN P, DING Y M, et al. Research status and trend of route planning for flying vehicles[J]. Journal of Astronautics, 2007, 28(6): 1441-1446.

[2] 刘鹤鸣，黄长强，黄汉桥，等. 快速避障三维最优轨迹规划研究[J]. 电光与控制，2013，20(3)：1-5.
LIU H M, HUANG C Q, HUANG H Q, et al. Fast and optimal three dimensional trajectory planning with obstacle avoidance performance [J]. Electronics Optics & Control, 2013, 20(3): 1-5.

[3] NILSSON N. Problem-solving methods in artificial intelligence[M]. New York: McGraw-Hill, 1971.

[4] SZCZERBA R J. Robust algorithm for real-time route planning[J]. IEEE Transactions on Aerospace and Electronic System, 2000, 36(3): 869-878.

[5] 孟中杰，黄攀峰，闫杰. 基于改进稀疏 A* 算法的高超声速飞行器航迹规划技术[J]. 西北工业大学学报，2010，28(2)：182-186.
MENG Z J, HUANG P F, YAN J. Exploring trajectory planning for hypersonic vehicle using improved sparse A* algorithm[J]. Journal of Northwestern Polytechnical University, 2010, 28(2): 182-186.

[6] 刘希，朱凡，蔡满意，等. 一种改进的快速航路规划方法[J]. 飞行力学，2011，29(1)：89-92.
LIU X, ZHU F, CAI M Y, et al. Improved method for fast path planning[J]. Flight Dynamics, 2011, 29(1): 89-92.

[7] 马培军，毛云云，张洪涛，等. 基于 3DSAS 的多约束航迹协同规划与搜索方法[J]. 系统工程与电子技术，2011，33(7)：1527-1533.
MA P J, MAO Y Y, ZHANG H T, et al. Cooperative planning for multiple trajectories with multiple constraints

（下转第89页）