



(12) 发明专利

(10) 授权公告号 CN 101694752 B

(45) 授权公告日 2012. 05. 02

(21) 申请号 200910088651. X

CN 1592919 A, 2005. 03. 09, 全文.

(22) 申请日 2009. 07. 06

刘星, 胡明华, 韩松臣. 自由飞条件下的冲突探测与解脱方法. 《南京理工大学学报》. 2002, 第 26 卷 (第 126 期), 56-60.

(73) 专利权人 民航数据通信有限责任公司

地址 100083 北京市海淀区北四环中路 238 号柏彦大厦 16 层

专利权人 北京航空航天大学

刘星, 胡明华, 董襄宁. 遗传算法在飞行冲突解脱中的应用. 《南京航空航天大学学报》. 2002, 第 34 卷 (第 1 期), 35-38.

(72) 发明人 朱衍波 李立群 张军 许有臣
赵灿 唐金翔

审查员 栗彬彬

(74) 专利代理机构 北京中北知识产权代理有限公司 11253

代理人 吴立

(51) Int. Cl.

G08G 5/00 (2006. 01)

(56) 对比文件

WO 2007072015 A2, 2007. 06. 28, 全文.

US 2009164122 A1, 2009. 06. 25, 全文.

CN 101385059 A, 2009. 03. 11, 全文.

CN 101197082 A, 2008. 06. 11, 全文.

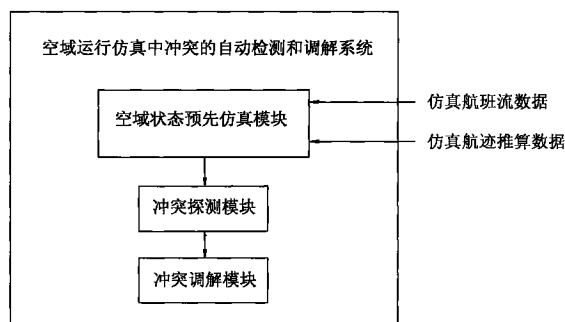
权利要求书 5 页 说明书 11 页 附图 9 页

(54) 发明名称

空域运行仿真中冲突的自动检测和调解系统及方法

(57) 摘要

本发明公开了一种空域运行仿真中冲突的自动检测和调解系统, 包括: 空域状态预先仿真模块, 用于预测在未来一段时间内仿真空域的运行情况; 冲突探测模块, 用于探测某个时间点空域内飞行流的冲突情况; 冲突调解模块, 用于在规定允许的调解时间余量内调解已经探测到的飞行冲突; 本系统实现了空域仿真运行中飞行冲突的自动检测和处理, 算法简单, 并可使得空域仿真更加贴近空域的实际运行情况, 为空域运行的科学评估、科学规划提供辅助决策手段。



1. 一种空域运行仿真中冲突的自动检测和调解系统,包括:

空域状态预先仿真模块,用于预测在未来一段时间内仿真空域的运行情况,具体是:采用离散仿真的方法预测未来一段时间内空域的运行状况,先设定仿真刷新时间以及连续预测次数两个变量,以确定系统预先仿真的时间量“仿真刷新时间*连续预测次数”,定义一个维数为连续预测次数的空域状态循环数组,每个数组项中保存有当前空域中运行的飞行器对象,循环数组中每一个数据项都比其之后的数据项增加仿真刷新时间的时间量,根据当前空域中飞行器的运行状况增加或删除更新飞行器对象,或者根据冲突调解方案进行调整;

冲突探测模块,用于根据空域状态预先仿真模块预测的数据探测某个时间点空域内飞行流的冲突情况,具体是:假定飞行器在巡航过程中始终位于RVSM指定的高度层上,因此飞行冲突探测仅在同高度层进行,判断的标准是用动态的Delaunay图判断某个时刻两架飞行器的水平距离是否小于指定距离间隔标准;

冲突调解模块,用于根据冲突探测模块的探测结果在规定允许的调解时间余量内调解已经探测到的飞行冲突,具体是:首先为冲突飞机建立冲突图,图中的节点代表探测到飞行冲突的飞行器,边代表两飞行器之间探测到了冲突,每次选择节点度最大的飞行器作为调解对象;设计了调高、盘旋、调速三种调解方案,并设定三种调解方案使用的优先级,按优先级顺序使用调解方案调解冲突,冲突调解成功后,即将该飞行器从冲突图中删除,继续选择节点度最大的飞行器调解,直到冲突图中没有节点。

2. 一种空域运行仿真中冲突的自动检测和调解方法,其特征在于,包含下述内容:

预测未来一段时间内仿真空域的运行情况:采用离散仿真的方法预测未来一段时间内空域的运行状况,先设定仿真刷新时间以及连续预测次数两个变量,以确定系统预先仿真的时间量“仿真刷新时间*连续预测次数”,定义一个维数为连续预测次数的空域状态循环数组,每个数组项中保存有当前空域中运行的飞行器对象,循环数组中每一个数据项都比其之后的数据项增加仿真刷新时间的时间量,根据当前空域中飞行器的运行状况增加或删除更新飞行器对象,或者根据冲突调解方案进行调整;

根据仿真空域的运行情况探测某个时间点空域内飞行流的冲突情况:假定飞行器在巡航过程中始终位于RVSM指定的高度层上,因此飞行冲突探测仅在同高度层进行,判断的标准是用动态的Delaunay图判断某个时刻两架飞行器的水平距离是否小于指定距离间隔标准;

根据飞行流冲突情况探测结果在规定允许的调解时间余量内调解已经探测到的飞行冲突:首先为冲突飞机建立冲突图,图中的节点代表探测到飞行冲突的飞行器,边代表两飞行器之间探测到了冲突,每次选择节点度最大的飞行器作为调解对象;设计了调高、盘旋、调速三种调解方案,并设定三种调解方案使用的优先级,按优先级顺序使用调解方案调解冲突,冲突调解成功后,即将该飞行器从冲突图中删除,继续选择节点度最大的飞行器调解,直到冲突图中没有节点。

3. 根据权利要求2所述的空域运行仿真中冲突的自动检测和调解方法,其特征在于:预测在未来一段时间内仿真空域运行情况的方法包含下述步骤:

步骤301,设定系统更新时间步长tStep,和循环状态数组的维数nStatus,预先仿真tStep*nStatus时间内空域的状态,下转302;

步骤 302, 建立循环状态数组 AirspaceStatus, 用于保存在 nStatus 个连续系统刷新点空域的状态, 下转 303;

步骤 303, 设 $i = 0$, $time = 0$, i 为系统当前刷新过的步数, $time$ 为系统当前仿真的时间, 下转 304;

步骤 304, 判断是否 $i < nStatus$, 如为是转步骤 305, 如为否转步骤 310;

步骤 305, 判断是否 $i = 0$, 如是转到步骤 308, 如否转 306;

步骤 306, 拷贝 AirspaceStatus[$i-1$] 中的飞行器对象到 AirspaceStatus[i] 中, 下转 307;

步骤 307, 依据仿真航迹推算数据和当前时间 $time$ 更新 AirspaceStatus[i] 中的飞行器状态信息, 下转 308;

步骤 308, 使用仿真航班流数据选择在 $time$ 时刻进入仿真空域的飞行器对象加入 AirspaceStatus[i], 下转 309;

步骤 309, 设 $i = i+1$, $time = i*tStep$, 并转 304;

步骤 310, 拷贝 AirspaceStatus[$(i-1) \bmod nStatus$] 中的飞行器对象到 AirspaceStatus[$i \bmod nStatus$] 中, 对 i 取模来进行循环利用, 下转 311;

步骤 311, 依据仿真航迹推算数据和当前时间 $time$ 更新 AirspaceStatus[$i \bmod nStatus$] 中的飞行器状态信息, 下转 312;

步骤 312, 删除 AirspaceStatus[$i \bmod nStatus$] 中达到终结状态的飞行器对象, 下转 313;

步骤 313, 使用仿真航班流数据选择在 $time$ 时刻进入仿真空域的飞行器对象加入 AirspaceStatus[$i \bmod nStatus$], 下转 314;

步骤 314, 使用冲突探测模块, 对 AirspaceStatus[$i \bmod nStatus$] 中飞行冲突进行检测, 并使用冲突调解模块在整个状态数组中予以解决, 下转 315;

步骤 315, 设 $i = i+1$, $time = i*tStep$, 下转 316;

步骤 316, 判断 $time$ 是否达到仿真终止时刻, 否则转步骤 304, 是则结束。

4. 根据权利要求 2 所述的空域运行仿真中冲突的自动检测和调解方法, 其特征在于: 探测某个时间点空域内飞行流的冲突情况的方法包含下述步骤:

步骤 401, 设仿真时刻 $time = 0$, 仿真更新步长为 $tStep$, 下转 402;

步骤 402, 根据当前整个管理空域飞行器的位置构建 Delaunay 图, 对每个高度层单独构建平面 Delaunay 图, 下转 403;

步骤 403, 对于每一个以 Delaunay 边为公共边的三角对, 初始化拓扑事件, 按时间顺序加入事件队列 eventQueue, 下转 404;

步骤 404, 对于每个 Delaunay 边判断其是否小于安全距离, 是则转 405, 否则转 407;

步骤 405, 将冲突飞机对传入冲突调解模块进行冲突调解, 下转 406;

步骤 406, 删除调解过的飞机对应的拓扑事件, 重新计算后按时间顺序插入 eventQueue, 下转 407;

步骤 407, 更新空域内飞行器的位置, 重新计算各 Delaunay 边的长度, 下转 408;

步骤 408, 判断当前时间是否有 eventQueue 事件, 否则转 410, 有则转步骤 409;

步骤 409, 对拓扑事件对应的三角形对执行 flip 操作来更新 Delaunay 图, 实现

Delaunay 图的拓扑结构更新,然后转 410;

步骤 410,当前时间飞行流生成模块是否有新的飞行器生成进入仿真空域,有则转 411,否则转 412;

步骤 411,在 Delaunay 图中增加一个顶点并相应执行更新,然后转 412;

步骤 412,判断当前时间是否有飞行器达到终结状态退出仿真空域,有则转步骤 413,否则转 414;

步骤 413,更新 Delaunay 图和拓扑事件,然后转 414;

步骤 414,设 $\text{time} = \text{time} + \text{tStep}$,下转 415;

步骤 415,判断是否达到仿真终止时刻,否则转步骤 404,是则结束。

5. 根据权利要求 2 所述的空域运行仿真中冲突的自动检测和调解方法,其特征在于:调解探测到的飞行冲突的方法包含下述步骤:

步骤 501,建立冲突图,节点为有冲突飞机,边代表两节点飞机之间有冲突,下转 502;

步骤 502,选择节点度最大的冲突飞机 t 作为解脱对象,下转 503;

步骤 503,确定各解脱算法的优先级,按优先级高低确定调解顺序;

步骤 504,采用高度调整方法解脱冲突,然后转 507;

步骤 505,采用盘旋等待方法解脱冲突,然后转 507;

步骤 506,采用速度调整方法解脱冲突,然后转 507;

步骤 507,判断冲突是否已经被调解,若是转步骤 510,若否转步骤 508;

步骤 508,判断是否还有其他优先级较低的调解方案;若是则按 503 确定的优先级顺序转 504、505 或 506,若否转步骤 509;

步骤 509,将调解失败记录写入日志;并转步骤 510;

步骤 510,将冲突飞机 t 从冲突图中删除,并删除与其直接相连的边,下转 511;

步骤 511,删除图中的孤立节点,下转 512;

步骤 512,判断冲突图中是否已经没有节点,否则说明还有其他冲突未调解,转步骤 502,是则所有的冲突都已被调解,该流程结束。

6. 根据权利要求 5 所述的空域运行仿真中冲突的自动检测和调解方法,其特征在于:用调整高度调解冲突的方法包含下述步骤:

步骤 601,给定 $\text{AirspaceStatus}[i]$ 中的一个冲突飞机 t ,下转 602;

步骤 602,随机设定高度层调整方向 upOrDown 为向上或者向下,以保证每次只朝同一个方向搜索,下转 603;

步骤 603,记录飞机的历史高度层 layerHistory ,下转 604;

步骤 604,判断飞机的飞行方向,如向东则转步骤 605,如向西则转步骤 606;

步骤 605,根据 RVSM 的规定确定向东可调整的高度层数量上限 $\text{layerMax} = 7$,然后下转 607;

步骤 606,根据 RVSM 的规定确定向西可调整的高度层数量上限 $\text{layerMax} = 6$,然后下转 607;

步骤 607,设定当前已尝试调整过的高度层数量 $\text{layerNum} = 1$,当前尝试调整的高度层 $\text{layerNow} = \text{layerHistory}$,下转 608;

步骤 608,如 upOrDown 值为 up,则判断是否已经调整到同向的最高层,否则判断是否已

经调整到同向的最低层,如已经达到调整上限则转步骤 609,否则转步骤 611;

步骤 609,将飞机的高度层重置为 layerHistory,下转 610;

步骤 610,将调整方向 upOrDown 取反,如为 up 则置为 down,反之则置为 up,下转 611;

步骤 611,按照 upOrDown 指示的方向调整 layerNow 到最近的同向的高度层,下转 612;

步骤 612,设 layerNum = layerNum+1,下转 613;

步骤 613,查询飞机 t 的飞行器性能表,获得该机型的爬升率或下降率,将状态数组 AirspaceStatus 中飞机 t 的高度层按照时间顺序和一定的爬升或下降率,从 layerHistroy 渐变为 layerNow,下转 614;

步骤 614,根据 ATC 标准判断调解过程中飞机 t 是否与其他飞机有冲突,如没有冲突,则调解成功结束,否则转步骤 615;

步骤 615,判断是否 layerNum < layerMax,如是则转步骤 608,继续尝试其他高度层,如否则调解失败。

7. 根据权利要求 5 所述的空域运行仿真中冲突的自动检测和调解方法,其特征在于:用盘旋等待调解冲突的方法包含下述步骤:

步骤 701,给定 AirspaceStatus[i] 中的一个冲突飞机对 t1、t2,更新时间步长 tStep,以及循环数组维数 nStatus,下转 702;

步骤 702,计算飞机 t1 需要盘旋等待的时间 time1,下转 703;

步骤 703,判断是否满足 time1 < tStep*nStatus, tStep*nStatus 是系统预先仿真的最大时间间隔,也是系统预留的最长冲突调解时间;是则转 704,否则转 707;

步骤 704,计算 n1 = time1/tStep+1,将冲突调解所需要的时间转化为盘旋等待所经历的系统状态刷新点的个数,下转 705;

步骤 705,设定循环数组 AirspaceStatus 里面 (i-n1)mod nStatus 到 (i-1)mod nStatus 数据项中飞机 t1 的状态为盘旋等待,下转 706;

步骤 706,判断飞机 t1 改变状态后是否与其他飞机产生冲突;是则转 707 步,否则结束;

步骤 707,计算飞机 t2 需要盘旋等待的时间 time2,下转 708;

步骤 708,判断是否 time2 < tStep*nStatus,是则转 709 步,否则冲突调节失败结束;

步骤 709,计算 n2 = time2/tStep+1,下转 710;

步骤 710,设定循环数组 AirspaceStatus 里面 (i-n2)mod nStatus 到 (i-1)mod nStatus 数据项中飞机 t2 的状态为盘旋等待,下转 711;

步骤 711,判断飞机 t2 改变状态后是否与其他飞机产生冲突,无则调解成功,结束;否则冲突调解失败结束。

8. 根据权利要求 7 所述的空域运行仿真中冲突的自动检测和调解方法,其特征在于:702 和 707 步中计算飞机需要盘旋等待时间是采用下述公式计算:

$$t = \frac{\text{distance}}{v_{\text{pass}}} = \frac{|v_y(P_{1x} - P_{2x}) - v_x(P_{1y} - P_{2y}) + d\sqrt{v_x^2 + v_y^2}|}{v_1 \bullet (-v_y, v_x)} \quad (3)$$

或者

$$t = \frac{\text{distance}}{v_{\text{pass}}} = \frac{|v_y(P_{1x} - P_{2x}) - v_x(P_{1y} - P_{2y}) - d\sqrt{v_x^2 + v_y^2}|}{v_1 \bullet (v_y, -v_x)} \quad (4)$$

$v_1 \bullet (-v_y, v_x) > 0$ 时使用公式 (3), $v_1 \bullet (v_y, v_x) > 0$ 时使用公式 (4);

其中 (P_{1x}, P_{1y}) 为飞机 t1 的位置, v_1 为相应速度, (P_{2x}, P_{2y}) 为飞机 t2 的位置, v_2 为相应速度, v 为飞机 t2 相对于飞机 t1 的速度, $(-v_y, v_x)$ 和 $(v_y, -v_x)$ 是与速度 v 垂直但反向的两个速度矢量, d 为安全间隔, v_{pass} 为飞机 t1 垂直飞越危险碰撞区域的速度。

9. 根据权利要求 5 所述的空域运行仿真中冲突的自动检测和调解方法, 其特征在于: 用调整速度调解冲突的方法包含下述步骤:

步骤 801, 给定 AirspaceStatus[i] 中的一个冲突飞机对 t1、t2, 更新时间步长 tStep, 以及循环数组维数 nStatus, 下转 802;

步骤 802, 计算飞机 t1 最优速度改变量 speed1, 以及飞越 t2 保护区所需的时间 time1, 下转 803;

步骤 803, 判断速度改变量 speed1 是否在飞行器性能的允许范围以内; 是则转 804 步, 否则转 808 步;

步骤 804, 判断调解时间是否小于系统提供的调解时间余量; 是则转 805 步, 否则转 808 步;

步骤 805, 计算 $n1 = \text{time1}/\text{tStep} + 1$, 下转 806;

步骤 806, 改变循环数组 AirspaceStatus 里面 $(i - n1) \bmod n\text{Status}$ 到 $(i - 1) \bmod n\text{Status}$ 数据项中飞机 t1 的速度, 下转 807;

步骤 807, 判断飞机 t1 改变状态后是否与其他飞机产生冲突; 否则调解成功结束, 是则转 808 步;

步骤 808, 计算飞机 t2 最优速度改变量 speed2, 以及飞越 t1 保护区所需的时间 time2, 下转 809;

步骤 809, 判断速度改变量 speed2 是否在飞行器性能的允许范围以内; 是则转 810, 否则转 508 判断是否还有其他优先级低的解决方案;

步骤 810, 判断调解时间是否小于系统提供的调解时间余量; 是则转 811, 否则调解失败, 转 508 判断是否还有其他优先级低的解决方案;

步骤 811, 计算 $n2 = \text{time2}/\text{tStep} + 1$, 下转 812;

步骤 812, 改变循环数组 AirspaceStatus 里面 $(i - n2) \bmod n\text{Status}$ 到 $(i - 1) \bmod n\text{Status}$ 数据项中飞机 t2 的速度, 下转 813;

步骤 813, 判断飞机 t2 改变状态后是否与其他飞机产生冲突, 无则冲突调解成功结束, 是则失败, 转 508 判断是否还有其他优先级低的解决方案。

空域运行仿真中冲突的自动检测和调解系统及方法

技术领域

[0001] 本发明属空中交通管理技术领域,具体涉及一种空域仿真运行过程中的飞行冲突的自动检测和调解系统及方法。

背景技术

[0002] 目前,我国各种空管新技术的应用层出不穷。雷达管制手段的逐步推进,数据链通信技术的普及应用,自动相关监视 (Automatic Dependent Surveillance Broadcast,简称 ADS-B) 技术的推广使用等,无不推动我国空管系统朝着更为有效、更为强大的方向发展。

[0003] 民航运输的快速发展,奥运空域保障的需要以及空管新技术的应用,使得空域结构复杂化,对空域提出了更大的需求。进行空域的科学规划,合理使用空域资源是提高空域利用率、增大空域容量的主要途径。而对空域的运行状况进行仿真,模拟全国或某一区域的空域运行情况,定量计算空域资源使用的瓶颈,测试空域规划的结果、检验对飞行矛盾解决的情况,并进一步进行科学的评估和论证,是进行空域设计和规划重要的辅助工具和手段。

[0004] 在现实环境中,飞机的飞行是在管制员的监视和干预下进行的。管制员实时收集军民航各单位对于空域的使用申请,并通过对当前空中交通态势的把握和对未来一段时间内的空中态势的主观判断,进行冲突探测并采取干预措施实施控制,从而达到规避风险、保证安全的目的。在虚拟环境下,要达到贴近实际的仿真效果,必须对管制员这一角色进行抽象,模拟其不同环境下的思考方式、判断行为和发出的指令,来构建一个逼近真实管制员的“虚拟管制员”,通过其自主判断和行为来探测并化解空中交通冲突。同时,利用专家系统技术,通过建立管制规则库,逼近模拟管制员探测、化解空中交通冲突的行为。

[0005] 国外航空运输比较发达的国家在空域管理方面都研究开发了空域管理和辅助设计系统,对空域的开发和管理方案进行科学、充分的评估和论证,以合理地利用有限的空域资源。由欧洲研制开发的 Reorganized ATM Mathematical Simulator (RAMS) 系统,根据机型的 4 维航迹精确计算,冲突检测和仿真,3 维空域划设构建和比较同一共享空域内的 4 维信息,动态的协调管制、流量和空域的冲突;由澳大利亚开发的 Total Airspace and Airport Modeler (TAAM) 系统具有先进的数据预处理技术,3D 图形与仿真技术,机场运行、终端区和 ATC 扇区容量评估,容量和流量控制,动态划设航路等功能;美国开发了一些全国空域系统 (NAS) 的仿真系统,来测验和推动流量管理的新方法和新系统,主要有 FAA 技术中心主持研发了 SIMMOD 系统,MITRE 公司开发了 TMAC 系统,运筹学办公室开发了 NASPAC 系统,MIT 开发的 ASCENT 系统,波音公司开发的 TAAM 系统等。其中 ASCENT 系统是麻省理工学院的 Draper 实验室的工作人员和研究生历时 10 年建立起来的,已被 NASA 采用来对空域和流量管理方法进行效益分析。这些系统的成功之处在于其对飞行过程中的随机因素的充分考虑和较为完善的空中交通管制逻辑,因此仿真效果比较贴近实际,经常被 FAA 用来进行空域和机场仿真。

[0006] 近年来,我国军民航先后开发了一些以空域容量评估为主的空域仿真系统,例如,民航总局空中交通管理局投资建设的“空域管理与评估系统”采用理论仿真结合雷达模拟

机的方法,对影响民用航空的各种要素进行了综合处理和建模,实现了机场、航路和单机场终端区的理论容量评估,并在一些大机场的容量评估和管制员工作负荷评估方面得到了成功的应用。同时目前国内也陆续开发出更符合实际应用的空域运行仿真飞行流生成系统(如中国专利申请:200910085829.5“一种空域运行仿真中飞行流引擎的实现系统及方法”),以及空域运行仿真4D航迹推算系统(如中国专利申请200910078451.6“航班推演系统和方法”,和200910076686.1“083163GF飞行流样本数据的获取方法”)。

[0007] 在仿真飞行流形成系统和仿真航迹推算系统的基础上,进一步地还可对空域运行仿真中可能出现的冲突进行检测和干预,现有的这方面技术存在解脱算法(如最优控制理论方法)计算复杂度高、调解方案执行繁琐的缺陷,不能适应大规模空域实时仿真的要求。

发明内容

[0008] 本发明的目的是在现有空域运行仿真系统中已形成的航班仿真流以及航班4D航迹推算数据的基础上,提供一种算法简单、并可更加贴近真实的空域运行情况的空域运行仿真中冲突的自动检测和调解系统及方法。

[0009] 为实现上述目的,本发明提出的空域运行仿真中冲突的自动检测和调解系统包括:

[0010] 空域状态预先仿真模块,用于预测在未来一段时间内仿真空域的运行情况;

[0011] 冲突探测模块,用于根据空域状态预先仿真模块预测的未来一段时间内仿真空域的运行情况数据探测某个时间点空域内飞行流的冲突情况;

[0012] 冲突调解模块,用于根据冲突探测模块的探测结果在规定允许的调解时间余量内调解已经探测到的飞行冲突。

[0013] 所述的空域状态仿真模块是根据已知的仿真航班流数据和仿真4D航迹推算数据预测未来一段时间内仿真空域的运行情况的。

[0014] 进一步方案是:所述冲突调解模块具体包括:

[0015] 高度调整冲突调解单元,用于调整飞行器的巡航高度层来调解飞行冲突;

[0016] 速度调整冲突调解单元,用于调整飞行器的巡航速度来调解飞行冲突;

[0017] 盘旋等待冲突调解单元,用于使飞行器盘旋等待一段时间后来调解飞行冲突。

[0018] 本发明空域运行仿真中冲突的自动检测和调解方法包含下述内容:

[0019] 预测在未来一段时间内仿真空域的运行情况;

[0020] 根据未来一段时间内仿真空域的运行情况探测某个时间点空域内飞行流的冲突情况;

[0021] 根据探测到的飞行流冲突情况在规定允许的调解时间余量内调解已经探测到的飞行冲突。

[0022] 本发明方法的进一步方案是:

[0023] 所述预测在未来一段时间内仿真空域的运行情况的方法是:采用离散仿真的方法预测未来一段时间内空域的运行状况,先设定仿真刷新时间以及连续预测次数两个变量,以确定系统预先仿真的时间量“仿真刷新时间*连续预测次数”,定义一个维数为连续预测次数的空域状态循环数组,每个数组项中保存有当前空域中运行的飞行器对象,循环数组中每一个数据项都比其之后的数据项增加仿真刷新时间的的时间量,根据当前空域中飞行器

的运行状况增加或删除更新飞行器对象,或者根据冲突调解方案进行调整。

[0024] 所述探测某个时间点空域内飞行流的冲突情况的方法是:假定飞行器在巡航过程中始终位于RVSM指定的高度层上,因此飞行冲突探测仅在同高度层进行,判断的标准是用动态的Delaunay图判断某个时刻两架飞行器的水平距离是否小于指定距离间隔标准;

[0025] 所述调解已经探测到的飞行冲突的方法是:首先为冲突飞机建立冲突图,图中的点代表探测到飞行冲突的飞行器,边代表两飞行器之间探测到了冲突,每次选择节点度最大的飞行器作为调解对象;设计了调高、盘旋、调速三种调解方案,并设定三种调解方案使用的优先级,按优先级顺序使用调解方案调解冲突,冲突调解成功后,即将该飞行器从冲突图中删除,继续选择节点度最大的飞行器调解直到冲突图中没有节点。

[0026] 上述系统和方法是在已有的空域运行仿真飞行流生成系统和空域运行仿真4D航迹推算系统的基础上,针对一些现有的冲突检测和解脱算法(如最优控制理论方法)计算复杂度高、调解方案执行繁琐的缺陷,在满足飞行器性能约束、空中交通管制规则和一定的管制员行为偏好的基础上,提出了一种基于几何最优的、离散的、确定型的冲突检测和调解系统,该系统和方法具有算法运算复杂度低的优势,能够满足大规模空域实时仿真对算法复杂度的要求。非常适合在全国空域仿真中对飞行冲突的自动检测和调解。

[0027] 本发明可在虚拟空域仿真环境中,实现空域仿真运行中飞行冲突的自动检测和处理,使得空域仿真更加贴近空域的实际运行情况,为空域运行的科学评估、科学规划奠定基础,并提供更加可行的辅助决策手段。

[0028] 下面通过附图和实施例,对本发明的技术方案做进一步的详细描述。

附图说明

[0029] 图1、本发明系统实施例方框示意图

[0030] 图2、图1所示冲突调节模块方框示意图

[0031] 图3、为空域状态预先仿真模块提供数据的飞行流引擎模块流程图

[0032] 图4、为空域状态预先仿真模块提供数据的4D航迹推算模块流程图

[0033] 图5、图1所示空域状态预先仿真模块流程图

[0034] 图6、图1所示冲突探测模块流程图

[0035] 图7、图1所示冲突调解模块流程图

[0036] 图8、图1所示高度调整冲突调解单元流程图

[0037] 图9、图1所示盘旋等待冲突调解单元流程图

[0038] 图10、图1所示速度调整冲突调解单元流程图

具体实施方式

[0039] 本实施例是一种空域运行仿真中冲突的自动检测和调解系统。

[0040] 现有空域运行仿真系统,已有空域运行仿真飞行流生成系统和空域运行仿真4D航迹推算系统,其中仿真飞行流生成系统可生成类似航班时刻表的数据,主要包括航班的起飞目的机场、起飞降落时间、飞行器型号,并生成起降机场、起飞时间、机型等信息汇总形成的模拟航班流;空域运行仿真4D航迹推算系统是在给定“航班时刻表”的情况下,推算出航班中间运行的轨迹,包括中间各个时刻,飞机的位置、高度、速度、加速度、航向信息,本系

统是在上述数据基础上,进一步在虚拟空域仿真环境中,对管制员这一角色进行抽象构建“虚拟管制员”,在掌握当前空域交通态势和未来一段时间内空域仿真态势的基础上,模仿真实管制员的行为,自动进行仿真空域内的冲突检测,并在满足飞行器性能约束、空中交通管制规则和一定的管制员行为偏好的基础上自动采取干预措施化解仿真空域中的冲突,实现空域仿真运行中飞行冲突的自动检测和处理,

[0041] 参见图 1,本系统包括:

[0042] 空域状态预先仿真模块,用于根据飞行流引擎模块提供的仿真航班流数据和 4D 航迹推算模块提供的仿真航迹推算数据预测在未来一段时间内仿真空域的运行情况;

[0043] 冲突探测模块,用于根据空域状态预先仿真模块预测的数据探测某个时间点空域内飞行流的冲突情况;

[0044] 冲突调解模块,用于根据冲突探测模块的探测结果在规定允许的调解时间余量内调解已经探测到的飞行冲突;

[0045] 图 2 所示的是冲突调解模块的结构图,包括:

[0046] 高度调整冲突调解单元,用于调整飞行器的巡航高度层来调解飞行冲突;

[0047] 速度调整冲突调解单元,用于调整飞行器的巡航速度来调解飞行冲突;

[0048] 盘旋等待冲突调解单元,用于使飞行器盘旋等待一段时间后来调解飞行冲突。

[0049] 图 3 是为空域状态预先仿真模块提供数据的飞行流引擎模块工作流程图;该模块的主要工作流程包含下述内容:

[0050] 确定航班流的要素信息,分析各要素的分布特征,确定相应的数据概率模型;

[0051] 根据航班流要素数据概率模型统计现行航班计划中各要素的取值情况,确定分布参数;

[0052] 根据统计得到的数据分别建立现行航班计划中各要素的概率分布模型;并综合形成航班流建模模型。

[0053] 在航班流要素建模模型基础上依据用户输入的仿真流流量等引擎条件求解,随机生成符合用户需求的航班流。

[0054] 图 4 是为空域状态预先仿真模块提供数据的 4D 航迹推算模块工作流程图,包含下述步骤:

[0055] 步骤 1、获取推演时间点的航班对象集合;

[0056] 步骤 2、判断集合中是否还有未处理的航班对象;

[0057] 步骤 3、获取航班对象的当前状态包括航班号、航空器型号、巡航速度、航线、位置信息、飞行航向信息和航班状态;

[0058] 步骤 4、根据航空器型号获取航空器性能信息,根据航线获取下一飞行航路;

[0059] 步骤 5、根据航班对象的当前状态和航空器性能推演出下一推演时间点的航班动态;

[0060] 步骤 6、存储下一推演时间点的航班对象集合。

[0061] 图 5 所示的是空域状态预先仿真模块工作流程图,具体包括:

[0062] 步骤 301,设定系统更新时间步长 $tStep$,和循环状态数组 $AirspaceStatus$ 的维数 $nStatus$ 。由于本系统是离散仿真系统,所以必须设定系统的刷新率 $tStep$,也即是说每隔 $tStep$ 的时间间隔更新一次空域的状态,而由于是预先仿真,所以必须在系统中保存几

个连续系统刷新点空域的状态,连续刷新点的个数即为 $nStatus$,也就是说系统预先仿真了 $tStep * nStatus$ 时间内空域的状态;

[0063] 步骤 302,建立循环状态数组 $AirspaceStatus$,用于保存在 $nStatus$ 个连续系统刷新点空域的状态;

[0064] 步骤 303,设 $i = 0, time = 0$, i 为系统当前刷新过的步数, $time$ 为系统当前仿真的时间;

[0065] 步骤 304,判断是否 $i < nStatus$,如为真表示状态数组 $AirspaceStatus$ 正在初始化阶段,转步骤 305,如为假则空域仿真处于运行阶段,转步骤 310;

[0066] 步骤 305,判断是否 $i = 0$,如为真则整个 $AirspaceStatus$ 数组状态为空,是空域第一次初始化,转到步骤 308;

[0067] 步骤 306,拷贝 $AirspaceStatus[i-1]$ 中的飞行器对象到 $AirspaceStatus[i]$ 中;

[0068] 步骤 307,依据仿真航迹推算数据和当前时间 $time$ 更新 $AirspaceStatus[i]$ 中的飞行器状态信息,下转 308;

[0069] 步骤 308,使用仿真航班流数据选择在 $time$ 时刻进入仿真空域的飞行器对象加入 $AirspaceStatus[i]$,下转 309;

[0070] 步骤 309,设 $i = i+1, time = i * tStep$;

[0071] 步骤 310,拷贝 $AirspaceStatus[(i-1) \bmod nStatus]$ 中的飞行器对象到 $AirspaceStatus[i \bmod nStatus]$ 中,由于 $AirspaceStatus$ 为循环数组,在仿真运行中最多只记录连续 $nStatus$ 个系统刷新点空域的状态,所以需要对 i 取模来进行循环利用;

[0072] 步骤 311,使用 4D 航迹推测模块的接口,根据 $time$ 更新 $AirspaceStatus[i \bmod nStatus]$ 中的飞行器状态信息;

[0073] 步骤 312,删除 $AirspaceStatus[i \bmod nStatus]$ 中达到终结状态的飞行器对象;

[0074] 步骤 313,使用仿真航班流数据选择在 $time$ 时刻进入仿真空域的飞行器对象加入 $AirspaceStatus[i \bmod nStatus]$,下转 314;

[0075] 步骤 314,使用冲突探测模块,对 $AirspaceStatus[i \bmod nStatus]$ 中飞行冲突进行检测,并使用冲突调解模块在整个状态数组中予以解决,系统预先仿真的时间为 $tStep * nStatus$,这个时间同时也是系统预留的冲突调解的时间余量,探测到的飞行冲突必须在 $tStep * nStatus$ 时间内得以调解,否则将认为调解失败;

[0076] 步骤 315,设 $i = i+1, time = i * tStep$;

[0077] 步骤 316,判断 $time$ 是否达到仿真终止时刻,如为假则转步骤 304,如为真则结束。

[0078] 飞行器的保护区采用 ATC(Air Traffic Control,简称 ATC) 标准,为一个以飞行器为中心的虚拟的竖直圆柱体,圆柱体半径为 2.5 海里,高度为 2000 英尺,任何一架飞行器的保护区进入到另一架飞行器的保护区则认为发生了飞行冲突。由于本仿真系统假定飞行器在巡航过程中,始终位于 RVSM(Reduced Vertical Separation Minimum,简称 RVSM) 高度层中,所以巡航阶段飞行冲突的判断仅在同高度层进行。

[0079] 传统的任意两飞机对距离判断的方法其算法复杂度为 $O(n^2)$,本例冲突探测模块采用了 Delaunay 图进行冲突检测,采用 Delaunay 图后冲突判断仅需要对 Delaunay 边进行,将判断局部化。构建 Delaunay 图最坏情况下算法的复杂度为 $O(n^2)$,而更新 Delaunay 图的算法复杂度仅为 $O(\lg n)$,这样就简化了在冲突检测上的运算,使得系统更加满足实时

冲突探测和解脱的需要。首先需要明确 Delaunay 图的几个概念：

[0080] 空圆：平面上内部没有顶点的圆；

[0081] Delaunay 边：对于平面上给定一组生成点 P 生成的任意三角系 T，若存在一个空圆过 T 中一条边的两个顶点，那么这条边为 Delaunay 边；

[0082] Delaunay 图：对于平面上给定一组生成点 P 生成的任意三角系 T，如果 T 中的每一条边都是 Delaunay 边，则 T 为 Delaunay 图。可以证明，如果在 P 中任意 4 点不共圆的情况下，生成的 Delaunay 图是唯一的；

[0083] flip 操作：有公共边的三角形，将其看作一个四边形，公共边看作一条对角线，则当另一条对角线于该边交于边上，则可擦去这条对角线换成另一条，使原来的三角形对变成另一个三角形对。此过程称为 flip 操作；

[0084] 拓扑事件：Delaunay 图中的顶点在移动过程中，Delaunay 图只发生两种类型的改变，一种是形状改变，但拓扑结构不发生变化，另一种是拓扑结构发生变化。使 Delaunay 图发生拓扑结构变化的顶点移动事件称为拓扑事件。

[0085] 图 6 所示的是冲突探测模块的流程图，具体包括：

[0086] 步骤 401，设仿真时刻 $time = 0$ ，仿真更新步长为 $tStep$ ；

[0087] 步骤 402，根据当前全国空域飞行器的位置构建 Delaunay 图，对每个高度层单独构建平面 Delaunay 图；

[0088] 步骤 403，对于每一个以 Delaunay 边为公共边的三角对，初始化拓扑事件，按时间顺序加入事件队列 `eventQueue`；

[0089] 步骤 404，对于每个 Delaunay 边判断其是否小于安全距离，如有则转步骤 405，如无则转步骤 407；

[0090] 步骤 405，将冲突飞机对传入冲突调解模块进行冲突调解；

[0091] 步骤 406，删除调解过的飞机对应的拓扑事件，重新计算后按时间顺序插入 `eventQueue`，由于飞行器的飞行状态在调解前后发生了改变，其对应的拓扑事件也会发生变化，所以需要重新计算；

[0092] 步骤 407，更新空域内飞行器的位置，重新计算各 Delaunay 边的长度；

[0093] 步骤 408，判断当前时间是否有 `eventQueue` 事件，有则转步骤 409；

[0094] 步骤 409，对拓扑事件对应的三角形对执行 flip 操作来更新 Delaunay 图，实现 Delaunay 图的拓扑结构更新；

[0095] 步骤 410，当前时间飞行流生成模块是否有新的飞行器生成进入仿真空域，有则需要在 Delaunay 图中增加一个顶点并相应执行更新，转步骤 411；

[0096] 步骤 411，加入一个新的飞行器顶点，找出当前时刻所有外接圆包含新加入节点的三角形并删除，形成空的凸壳，将删除三角形边对应的拓扑事件删除；将凸壳顶点与新加入顶点连接；对新生成边为公共边的三角形对计算其可能的拓扑事件按时间顺序插入 `eventQueue` 中；

[0097] 步骤 412，判断当前时间是否有飞行器达到终结状态退出仿真空域，有则需要在 Delaunay 图中删除相应的顶点并更新，转步骤 413；

[0098] 步骤 413，更新 Delaunay 图和拓扑事件，具体操作与步骤 411 类似；

[0099] 步骤 414，设 $time = time + tStep$ ；

- [0100] 步骤 415,判断是否达到仿真终止时刻,否则转步骤 404,是则结束。
- [0101] 图 7 所示的是冲突调解模块的流程图,具体包括:
- [0102] 步骤 501,建立冲突图,节点为有冲突飞机,边代表两节点飞机之间有冲突;
- [0103] 步骤 502,选择节点度最大的冲突飞机 t 作为解脱对象,节点度越大表示与此飞机冲突的飞机数量越多,首先对此飞机进行冲突调解可以达到最好的调解效果;
- [0104] 步骤 503,确定各解脱算法的优先级,按优先级高低确定调解顺序,各解脱算法的优先级排序反映了虚拟管制员的管制偏好,或者代表了某管制区既定的管制规则;
- [0105] 步骤 504,使用高度调整冲突调解单元解脱冲突;
- [0106] 步骤 505,使用盘旋等待冲突调解单元解脱冲突;
- [0107] 步骤 506,使用速度调整冲突调解单元解脱冲突;
- [0108] 步骤 507,判断冲突是否已经被调解,否则转步骤 508;
- [0109] 步骤 508,判断是否还有其他优先级较低的调解方案;
- [0110] 步骤 509,如调解失败,则将调解失败记录写入日志;
- [0111] 步骤 510,将冲突飞机 t 从冲突图中删除,并删除与其直接相连的边;
- [0112] 步骤 511,删除图中的孤立节点,删除冲突图中的某些边以后,有些节点将被孤立,这代表空域中已经没有其他节点飞机与此飞机冲突,所以将其删除;
- [0113] 步骤 512,判断冲突图中是否已经没有节点,有则说明还有其他冲突未调解,转步骤 502,无则所有的冲突都已被调解,该模块流程结束。
- [0114] 民航于 2007 年 11 月 21 日实施缩小垂直间隔 RVSM 飞行高度层配备标准以后,RVSM 空域可用的高度层有 13 个,其中向东航向的高度层为 7 个,向西航向的高度层为 6 个。按照 RVSM 的要求,飞行器必须飞行在与其巡航方向相匹配的高度层上,这就是说飞行器可选择的高度层是非常有限的,所以本系统采取遍历寻优的方式来寻找可以解决飞行冲突的高度层。
- [0115] 图 8 所示的是高度调整冲突调解单元的流程图,具体包括:
- [0116] 步骤 601,给定 AirspaceStatus[i] 中的一个冲突飞机 t (数组 AirspaceStatus 的第 i 个成员);
- [0117] 步骤 602,随机设定高度层调整方向 upOrDown 为 up(向上)或者 down(向下),设定调整方向的目的是保证每次只朝同一个方向搜索(upOrDown 是变量名称,可以取 up 或 down 两个值);
- [0118] 步骤 603,记录飞机的历史高度层 layerHistory(变量名称 layerHistory 是一个高度值);
- [0119] 步骤 604,判断飞机的飞行方向,如向东则转步骤 605,如向西则转步骤 606;
- [0120] 步骤 605,根据 RVSM(Reduced Vertical Separation Minimum,缩小垂直间隔标准)的规定,RVSM 空域向东有 7 个高度层,也就是说可调整的高度层数量上限 layerMax = 7;
- [0121] 步骤 606,同理,RVSM 空域向西有 6 个高度层,也就是说可调整的高度层数量上限 layerMax = 6;
- [0122] 步骤 607,设定当前已尝试调整过的高度层数量 layerNum = 1,当前尝试调整的高度层 layerNow = layerHistory;

[0123] 步骤 608, 如 upOrDown 值为 up, 则判断是否已经调整到同向的最高层, 否则判断是否已经调整到同向的最低层。如已经达到调整上限则转步骤 609, 否则转步骤 610;

[0124] 步骤 609, 将飞机的高度层重置为 layerHistory;

[0125] 步骤 610, 将调整方向 upOrDown 取反, 如为 up 则置为 down, 反之则置为 up;

[0126] 步骤 611, 按照 upOrDown 指示的方向调整 layerNow 到最近的同向的高度层;

[0127] 步骤 612, $\text{layerNum} = \text{layerNum} + 1$;

[0128] 步骤 613, 查询飞机 t 的飞行器性能表, 获得该机型的爬升率或下降率, 将状态数组 AirspaceStatus 中飞机 t 的高度层按照时间顺序和一定的爬升或下降率, 从 layerHistroy 渐变为 layerNow;

[0129] 步骤 614, 判断除 AirspaceStatus[i] 外其他数组项中, 飞机 t 有没有与其他飞机产生冲突。高度层调整后, 飞机 t 当前的冲突被调解, 但很有可能在调解过程中引发同其他飞机的冲突, 由于这个冲突只会产生在调解飞机与空域内其他飞机之间, 所以仅需要根据 ATC 标准 (Air Traffic Control, 空中交通管制) 判断调解过程中飞机 t 是否与其他飞机有冲突。如没有冲突, 则调解成功, 否则转步骤 615;

[0130] 步骤 615, 判断是否 $\text{layerNum} < \text{layerMax}$, 如为真则转步骤 608, 继续尝试其他高度层, 如为假则调解失败。

[0131] 如果两架飞机 t1 和 t2 发生冲突, 以飞机 t1 为参照系, 则飞机 t2 飞行的安全区域为, 与两架飞机相对飞行方向平行的并且与飞机 t2 为圆心, 安全距离为半径的圆相切的两条直线之间的区域, 称为飞机 t2 的“影子”区域。采用飞机 t2 盘旋等待的方法解决飞行冲突, 相当于飞机 t2 不动, 而飞机 t1 飞出飞机 t2 的“影子”区域。而飞机 t2 盘旋等待的时间也即为飞机 t1 离开飞机 t2 的“影子”区域所需要的时间。

[0132] 设飞机 t1 的位置为 (P_{1x}, P_{1y}) , 速度为 v_1 , 飞机 t2 的位置为 (P_{2x}, P_{2y}) , 速度为 v_2 , 飞机 t2 相对于飞机 t1 的速度 $v = v_2 - v_1$, 则:

[0133] 定义与速度 v 垂直但反向的两个速度矢量 $(-v_y, v_x)$ 和 $(v_y, -v_x)$, 其中 v_x 和 v_y 分别为 v 在 x 轴上的坐标分量和 v 在 y 轴上的坐标分量。以飞机 t2 为坐标原点, 两个速度矢量分别对应飞机 t2 “影子”区域的两条边界直线:

$$[0134] \quad (-v_y, v_x) \text{ 对应 } xv_y - yv_x + d\sqrt{v_x^2 + v_y^2} = 0 \quad (1)$$

$$[0135] \quad (v_y, -v_x) \text{ 对应 } xv_y - yv_x - d\sqrt{v_x^2 + v_y^2} = 0 \quad (2)$$

[0136] d 为安全间隔。如果 $v_1 \cdot (-v_y, v_x) > 0$ 则飞机 t1 须飞越边界 (1), 垂直飞越速度为:

$$[0137] \quad v_{pass} = \frac{v_1 \cdot (-v_y, v_x)}{\sqrt{v_y^2 + v_x^2}}$$

[0138] 飞机 t1 $(P_{1x} - P_{2x}, P_{1y} - P_{2y})$ 到边界 (1) 的距离为

$$[0139] \quad \text{distance} = \frac{|v_y(P_{1x} - P_{2x}) - v_x(P_{1y} - P_{2y}) + d\sqrt{v_x^2 + v_y^2}|}{\sqrt{v_x^2 + v_y^2}}$$

[0140] 飞越时间即盘旋等待时间

$$[0141] \quad t = \frac{\text{distance}}{v_{\text{pass}}} = \frac{|v_y(P_{1x} - P_{2x}) - v_x(P_{1y} - P_{2y}) + d\sqrt{v_x^2 + v_y^2}|}{v_1 \bullet (-v_y, v_x)} \quad (3)$$

[0142] 如果 $v_1 \bullet (v_y, -v_x) > 0$ 则飞机 t1 须飞越边界 (2), v_{pass} 和 distance 相应变为:

$$[0143] \quad v_{\text{pass}} = \frac{v_1 \bullet (v_y, -v_x)}{\sqrt{v_y^2 + v_x^2}}$$

$$[0144] \quad \text{distance} = \frac{|v_y(P_{1x} - P_{2x}) - v_x(P_{1y} - P_{2y}) - d\sqrt{v_x^2 + v_y^2}|}{\sqrt{v_x^2 + v_y^2}}$$

[0145] 飞越时间即盘旋等待时间

$$[0146] \quad t = \frac{\text{distance}}{v_{\text{pass}}} = \frac{|v_y(P_{1x} - P_{2x}) - v_x(P_{1y} - P_{2y}) - d\sqrt{v_x^2 + v_y^2}|}{v_1 \bullet (v_y, -v_x)} \quad (4)$$

[0147] 图 9 所示的是盘旋等待冲突调解单元的流程图,具体包括:

[0148] 步骤 701, 给定 AirspaceStatus[i] 中的一个冲突飞机对 t1、t2, 更新时间步长 tStep, 以及循环数组维数 nStatus;

[0149] 步骤 702, 根据公式 (3) 或 (4) 计算飞机 t1 需要盘旋等待的时间 time1;

[0150] 步骤 703, 判断是否满足 $\text{time1} < \text{tStep} * \text{nStatus}$, $\text{tStep} * \text{nStatus}$ 是系统预先仿真的最大时间间隔, 也是系统预留的最长冲突调解时间;

[0151] 步骤 704, 计算 $\text{n1} = \text{time1} / \text{tStep} + 1$, 将冲突调解所需要的时间转化为盘旋等待所经历的系统状态刷新点的个数;

[0152] 步骤 705, 设定循环数组 AirspaceStatus 里面 $(i - \text{n1}) \bmod \text{nStatus}$ 到 $(i - 1) \bmod \text{nStatus}$ 数据项中飞机 t1 的状态为盘旋等待;

[0153] 步骤 706, 判断飞机 t1 改变状态后是否与其他飞机产生冲突, 由于冲突只会发生在 t1 与其他飞机之间, 所以仅需要判断改变状态 t1 与其他飞机直接的距离是否小于安全间隔;

[0154] 步骤 707, 根据公式 (3) 或 (4) 计算飞机 t2 需要盘旋等待的时间 time2;

[0155] 步骤 708, 判断是否 $\text{time2} < \text{tStep} * \text{nStatus}$, 是则转 709 步, 否则冲突调解失败;

[0156] 步骤 709, 计算 $\text{n2} = \text{time2} / \text{tStep} + 1$;

[0157] 步骤 710, 设定循环数组 AirspaceStatus 里面 $(i - \text{n2}) \bmod \text{nStatus}$ 到 $(i - 1) \bmod \text{nStatus}$ 数据项中飞机 t2 的状态为盘旋等待;

[0158] 步骤 711, 判断飞机 t2 改变状态后是否与其他飞机产生冲突, 无则调解成功, 结束, 否则调解失败。

[0159] 速度调整解决冲突的方法就是在飞行器飞行性能允许的范围内, 调解一对冲突飞机中的一架飞机来解决冲突的方法。设飞机 t1 的位置为 (P_{1x}, P_{1y}) , 速度为 $\bar{v}_1$, v_1 和 x_1 分别为大小和方向, 飞机 t2 的位置为 (P_{2x}, P_{2y}) , 速度为 $\bar{v}_2$, v_2 和 x_2 分别为大小和方向, 飞机 t2 相对于飞机 t1 的速度 $\bar{v} = \bar{v}_1 - \bar{v}_2$, v 和 x 分别为大小和方向, 则满足:

$$[0160] \quad v = \sqrt{v_1^2 + v_2^2 - 2v_1v_2 \cos(x_1 - x_2)}$$

$$[0161] \quad \chi = \arctan \left\{ \frac{v_1 \sin \chi_1 - v_2 \sin \chi_2}{v_1 \cos \chi_1 - v_2 \cos \chi_2} \right\} \quad (5)$$

[0162] 设 $\bar{r}_{LOS}$ 为从飞机 t1 到飞机 t2 的方向向量, r_{LOS} 和 χ_{LOS} 分别为大小和方向, 则满足:

$$[0163] \quad r_{LOS} = \sqrt{(P_{2x} - P_{1x})^2 + (P_{2y} - P_{1y})^2}$$

$$[0164] \quad \chi_{LOS} = \arctan \frac{P_{2y} - P_{1y}}{P_{2x} - P_{1x}}$$

[0165] 假设调整飞机 t1 的速度 $\bar{v}_1$ 为 $\bar{v}_1^*$ 得到新的相对速度 $\bar{v}^*$, 则速度 $\bar{v}^*$ 的最优方向 χ^* 为经过飞机 t1, 与飞机 t2 为圆心, 安全距离 d 为半径的圆相切的方向, 亦满足:

$$[0166] \quad \chi^* = \chi_{LOS} \pm \arcsin \frac{d}{r_{LOS}}$$

[0167] 与公式 (5) 同理得:

$$[0168] \quad \frac{\sin \chi^*}{\cos \chi^*} = \frac{v_1^* \sin \chi_1^* - v_2 \sin \chi_2}{v_1^* \cos \chi_1^* - v_2 \cos \chi_2}$$

[0169] 由于仅改变飞机 t1 的速度, 变化前后 $\chi_1^* = \chi_1$, 则对上式做代数变换得:

$$[0170] \quad v_1^* = v_2 \frac{\sin(\chi^* - \chi_2)}{\sin(\chi^* - \chi_1)} \quad (6)$$

[0171] 根据飞机 t1 到飞机 t2 保护区边缘的距离已经速度调整后的相对速度大小不难算出冲突调解所需要的最短时间。

[0172] 图 10 所示的是速度调整冲突调解单元的流程图, 具体包括:

[0173] 步骤 801, 给定 AirspaceStatus[i] 中的一个冲突飞机对 t1、t2, 更新时间步长 tStep, 以及循环数组维数 nStatus;

[0174] 步骤 802, 根据公式 (6) 计算飞机 t1 最优速度改变量 speed1, 以及飞越 t2 保护区所需的时间 time1;

[0175] 步骤 803, 判断速度改变量 speed1 是否在飞行器性能的允许范围以内;

[0176] 步骤 804, 判断调解时间是否小于系统提供的调解时间余量;

[0177] 步骤 805, 计算 $n1 = \text{time1}/\text{tStep} + 1$;

[0178] 步骤 806, 改变循环数组 AirspaceStatus 里面 $(i - n1) \bmod n\text{Status}$ 到 $(i - 1) \bmod n\text{Status}$ 数据项中飞机 t1 的速度;

[0179] 步骤 807, 判断飞机 t1 改变状态后是否与其他飞机产生冲突, 否则调解成功, 结束, 是则转 808 步;

[0180] 步骤 808, 根据公式 (6) 计算飞机 t2 最优速度改变量 speed2, 以及飞越 t1 保护区所需的时间 time2;

[0181] 步骤 809, 判断速度改变量 speed2 是否在飞行器性能的允许范围以内, 是则转 810, 否则转 508 判断是否还有其它优先级低的解决方案;

[0182] 步骤 810, 判断调解时间是否小于系统提供的调解时间余量, 是则转 811, 否则转 508 判断是否还有其它优先级低的解决方案;

[0183] 步骤 811, 计算 $n2 = \text{time2}/\text{tStep} + 1$;

[0184] 步骤 812, 改变循环数组 AirspaceStatus 里面 $(i - n2) \bmod n\text{Status}$ 到 $(i - 1) \bmod$

nStatus 数据项中飞机 t2 的速度；

[0185] 步骤 813, 判断飞机 t2 改变状态后是否与其他飞机产生冲突, 无则冲突调解成功, 结束, 否则失败, 转 508 判断是否还有其它优先级低的解决方案。

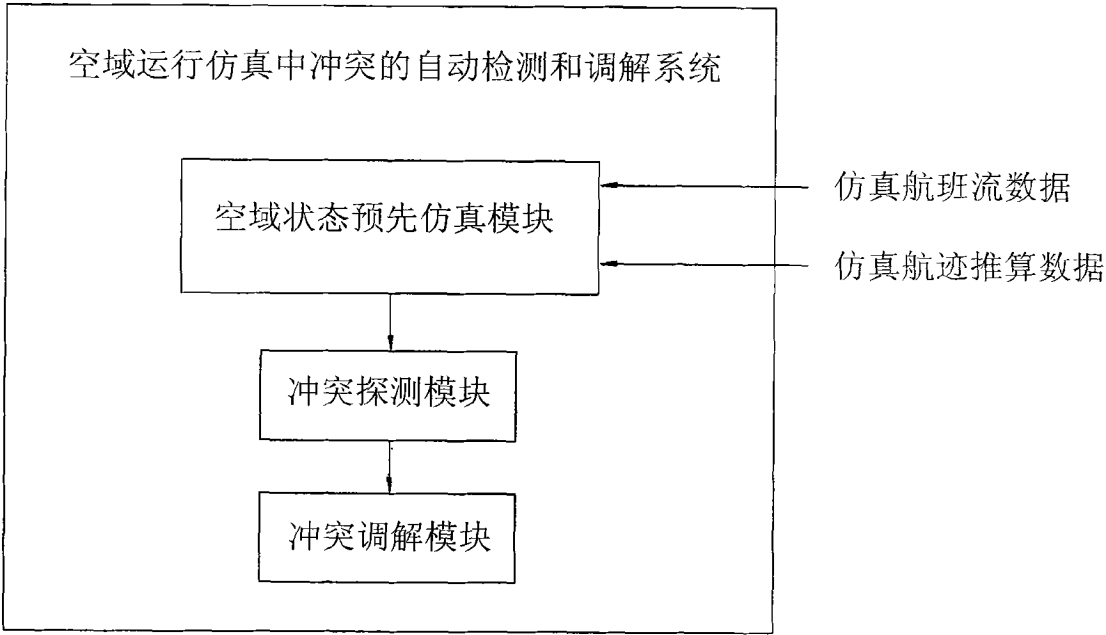


图 1

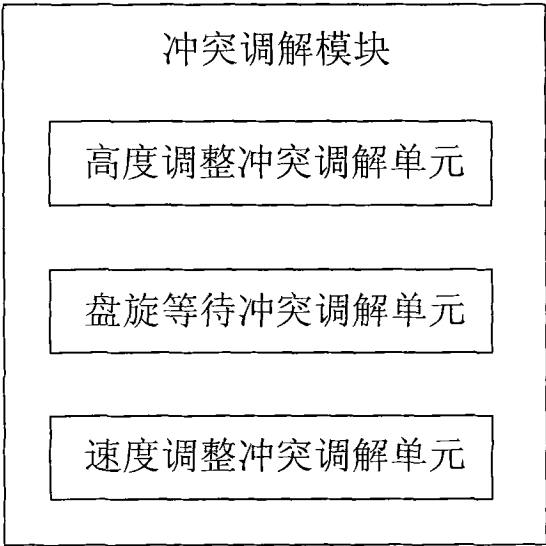


图 2

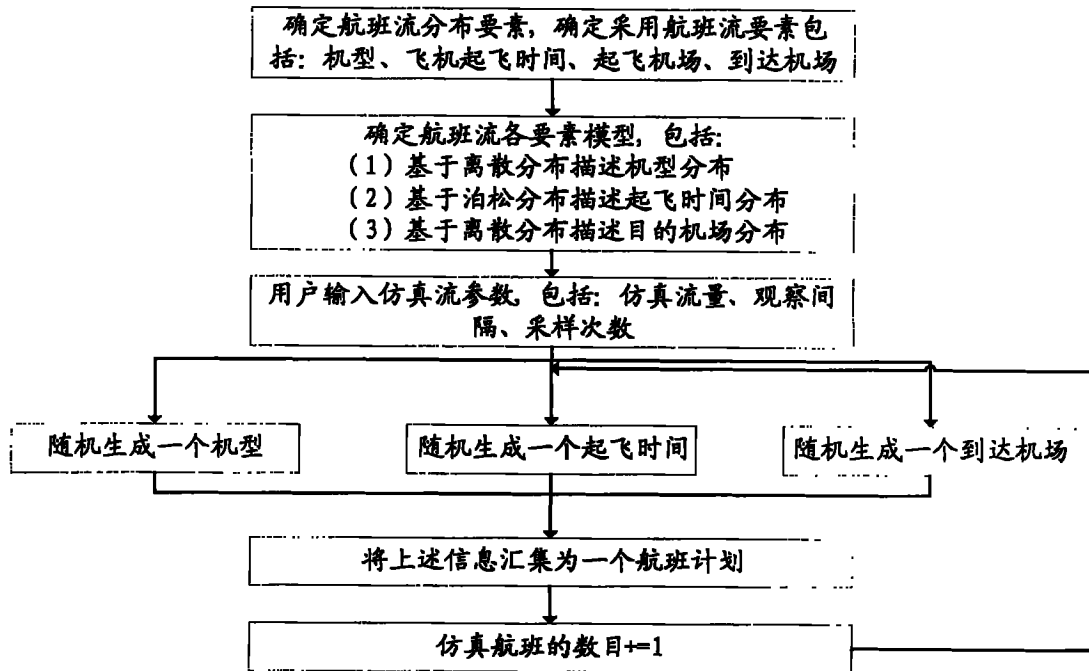


图 3

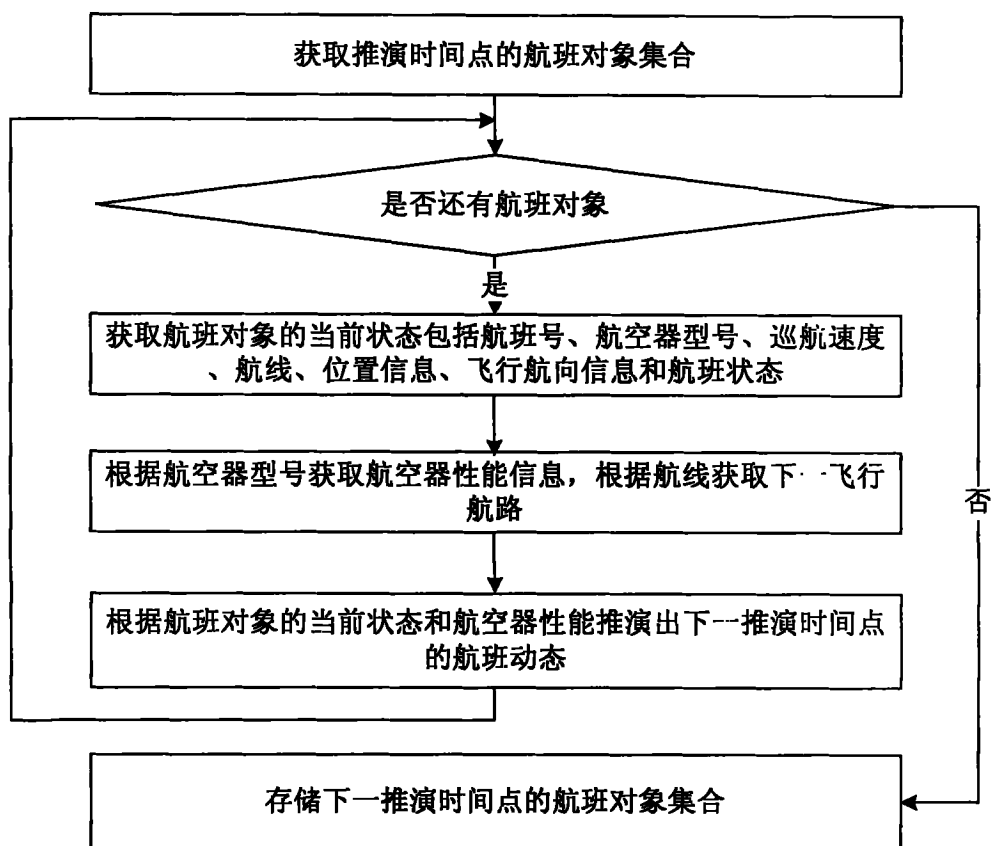


图 4

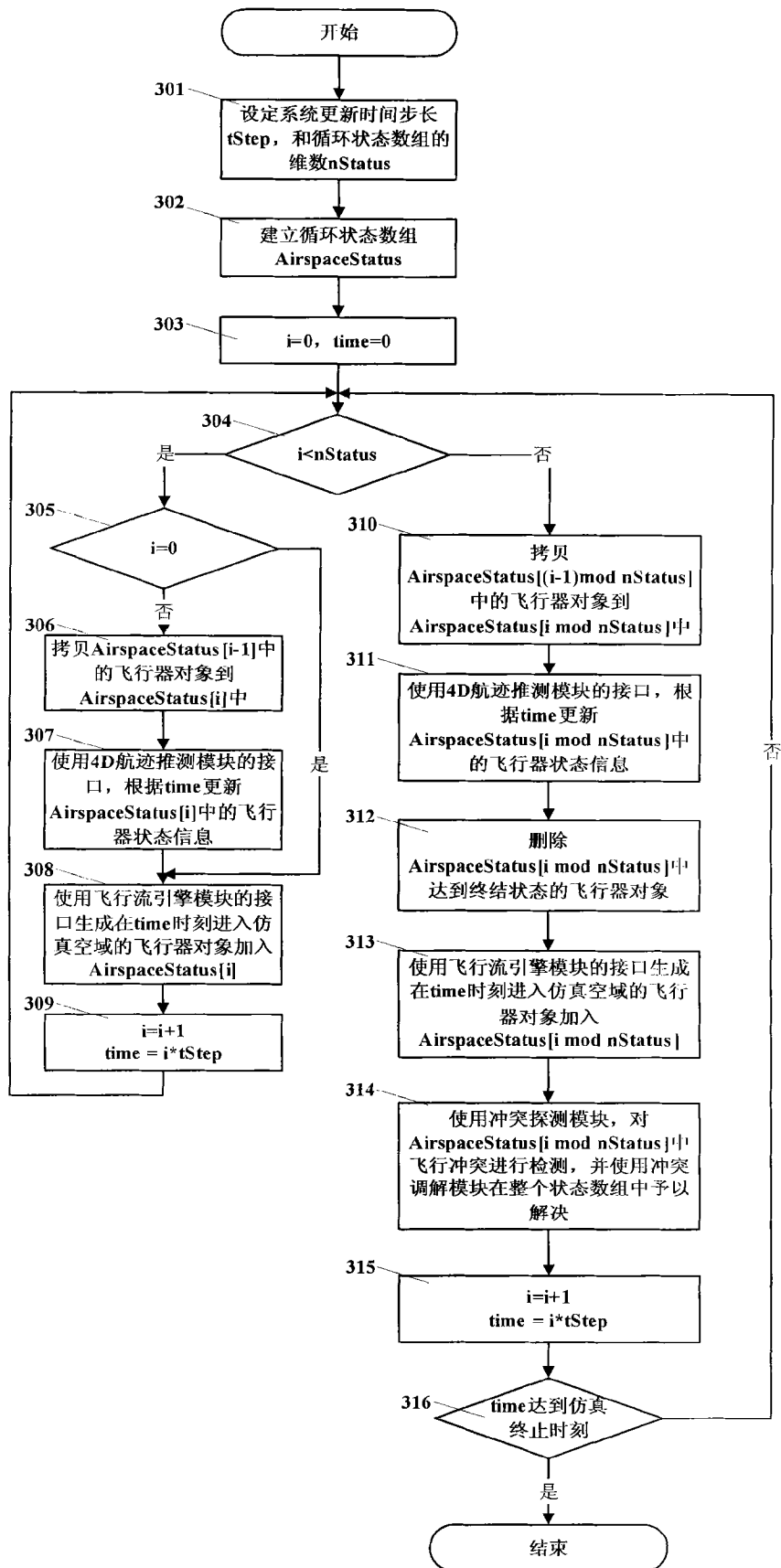


图 5

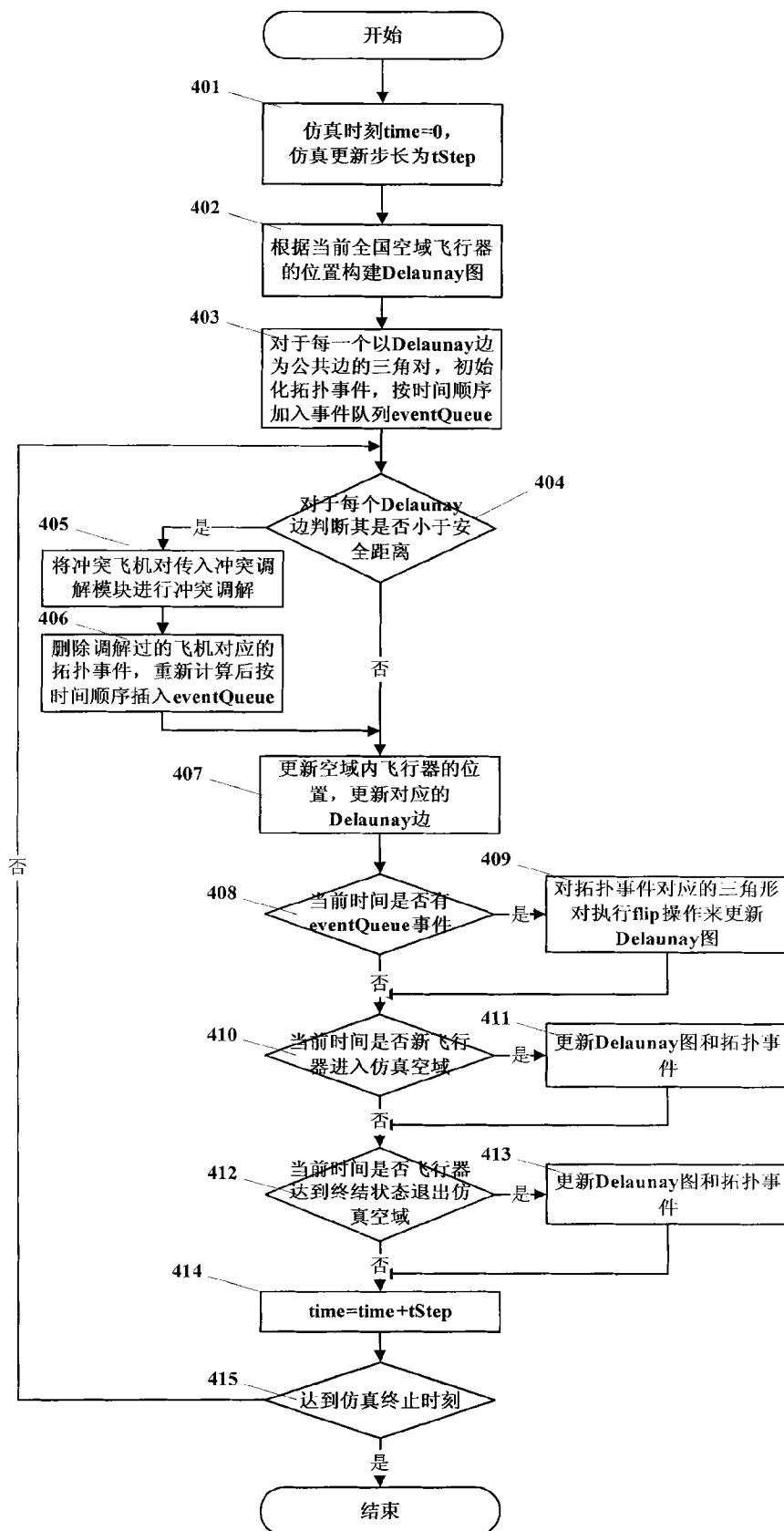


图 6

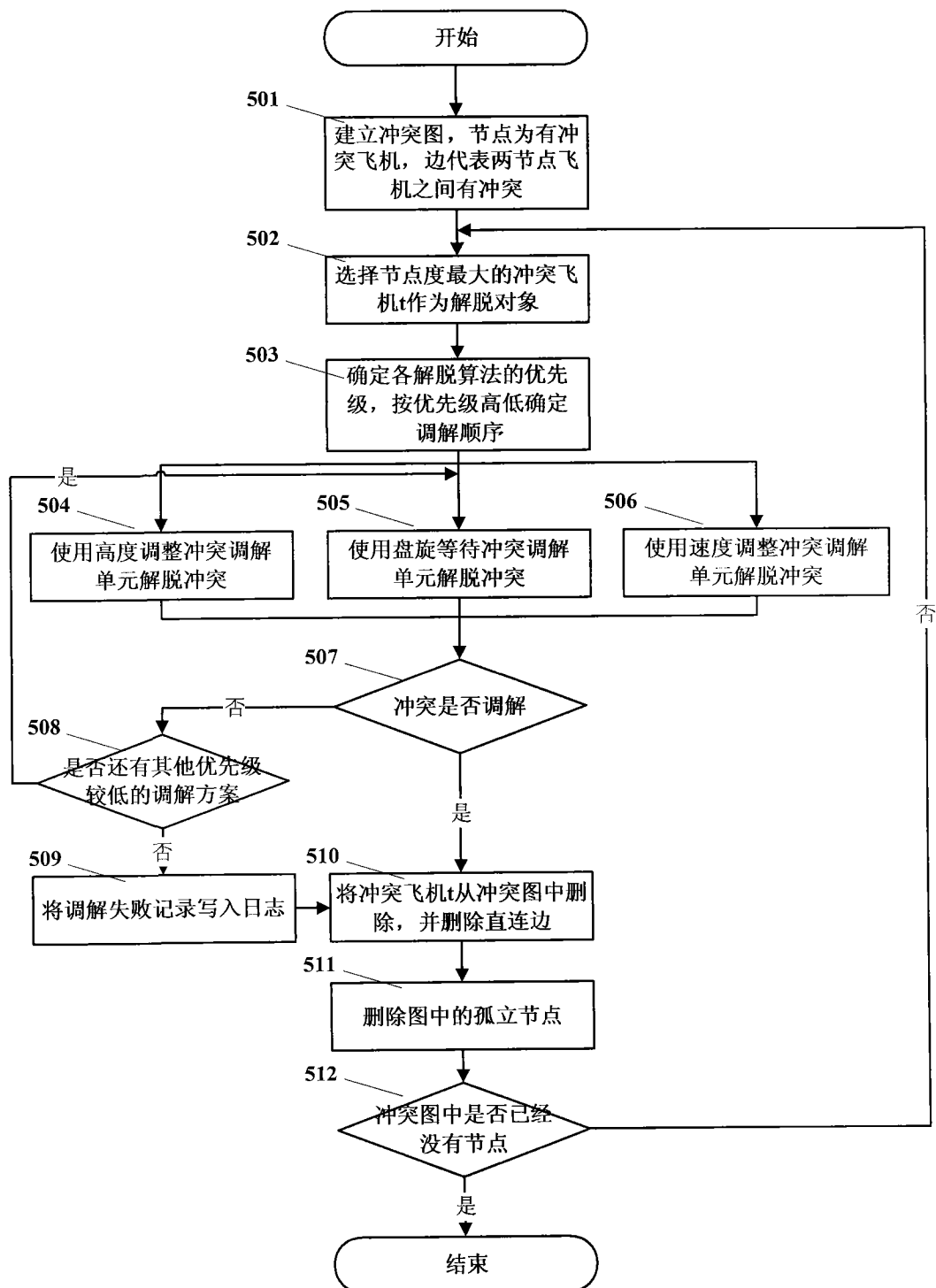


图 7

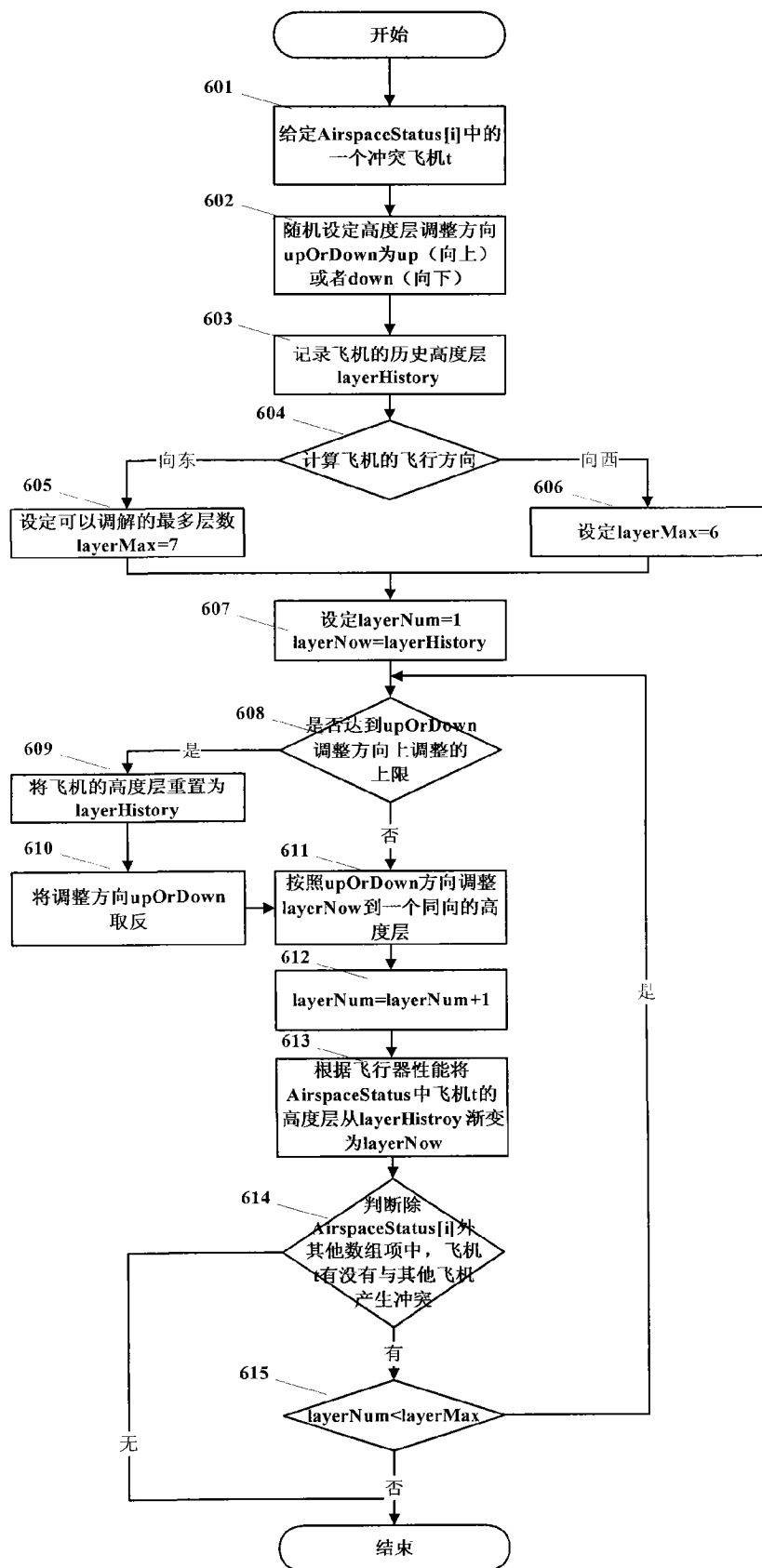


图 8

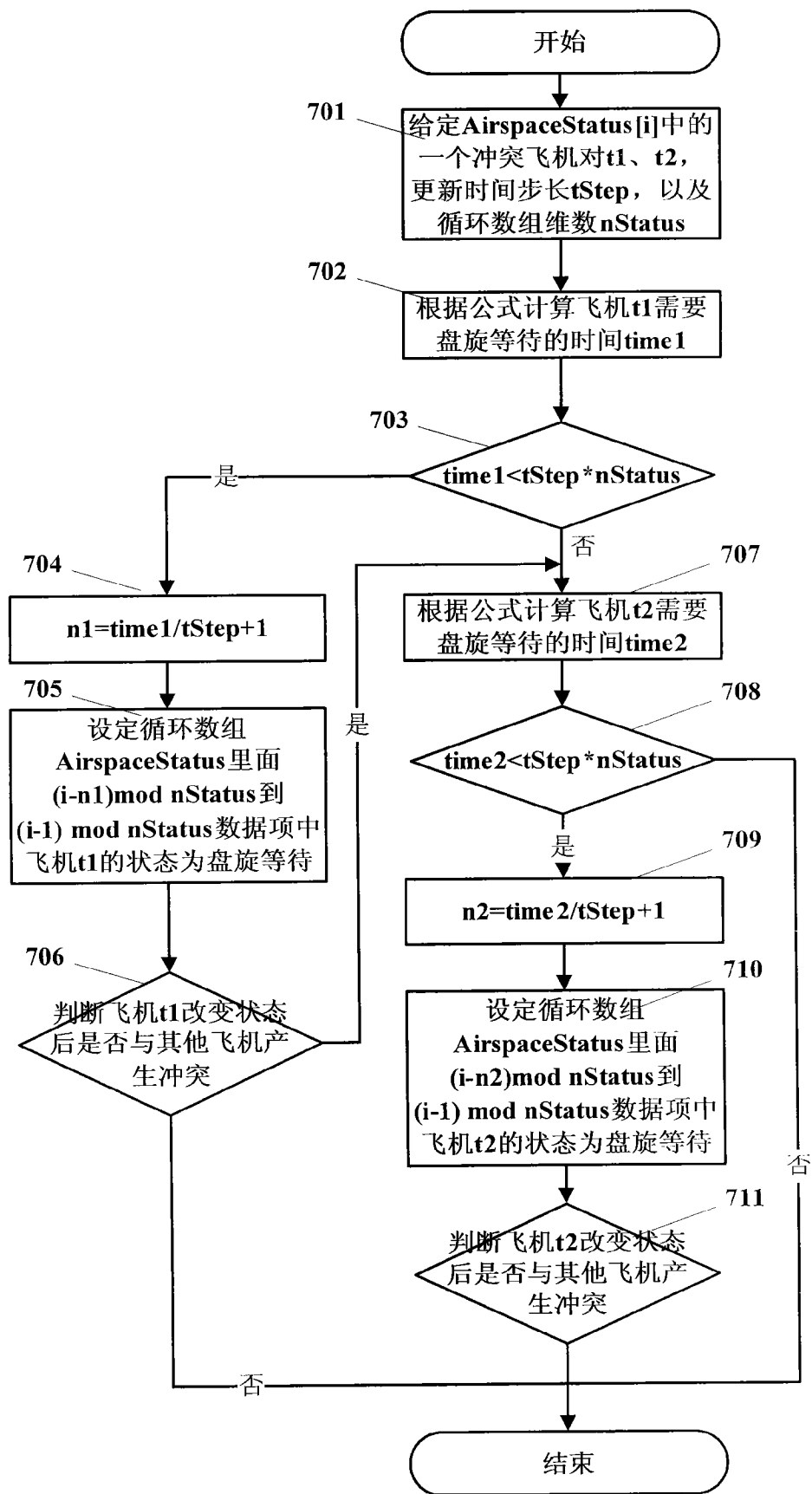


图 9

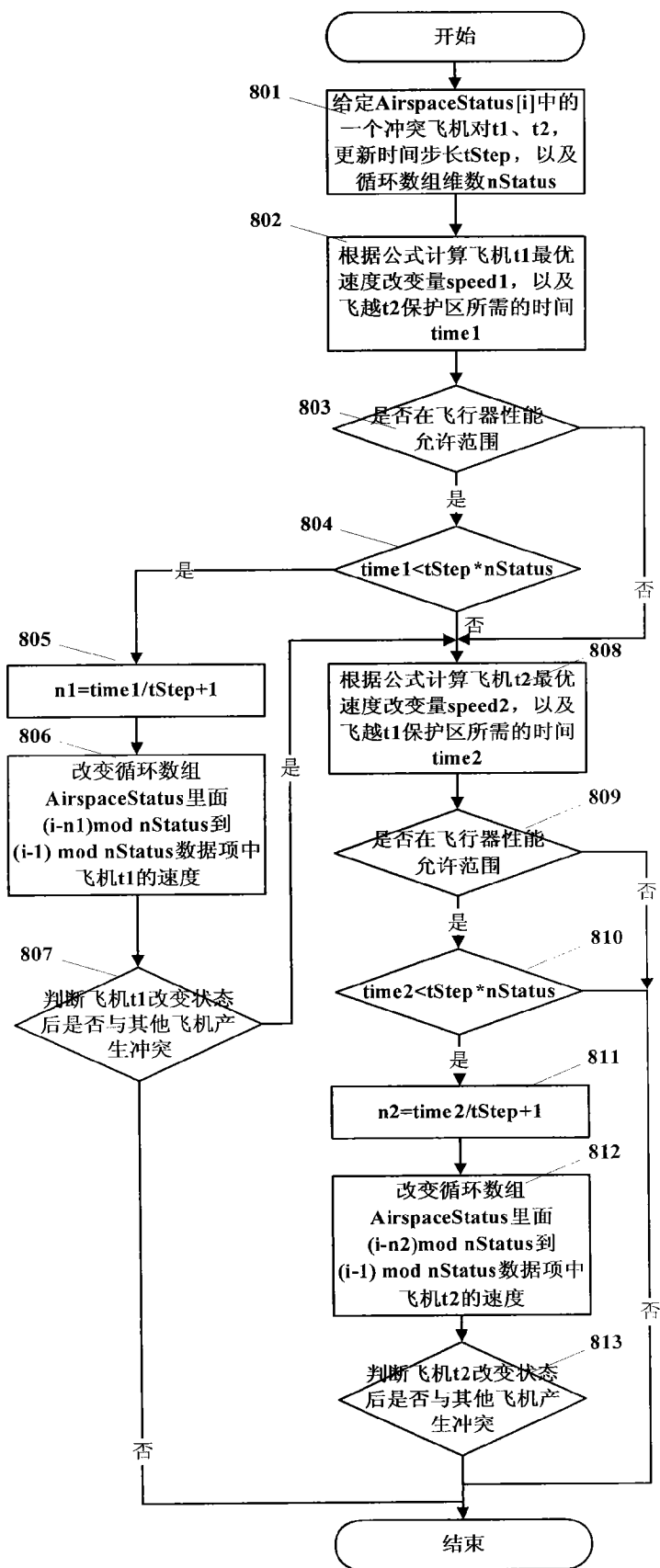


图 10