(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102122251 B
(45)授权公告日 2016.09.14

(21)申请号 201110067398.7

(22)申请日 2011.03.21

(65)同一申请的已公布的文献号
申请公布号 CN 102122251 A

(43)申请公布日 2011.07.13

(73)专利权人 北京航空航天大学
地址 100191 北京市海淀区学院路37号
专利权人 北京空间飞行器总体设计部

(72)发明人 郑征 林树民 常进 刘一帆 蔡开元

(74)专利代理机构 北京慧泉知识产权代理有限公司 11232
代理人 王顺荣 唐爱华

(51)Int.Cl.
G06F 9/48(2006.01)
G06F 9/44(2006.01)

(56)对比文件
CN 1776554 A,2006.05.24,
CN 101916404 A,2010.12.15,
US 2003158766 A1,2003.08.21,
US 2008144493 A1,2008.06.19,
梁旭等.基于遗传算法的并行测试调度算法研究.《电子测量与仪器学报》.2009,第23卷(第2期),第19-24页.
Mathias M. Adankon等.Genetic algorithm-based training for semi-supervised SVM.《Neural Computing and Applications》.2010,第9卷第1197-1206页.
Orhan Engin等.An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems.《Applied Soft Computing Journal》.2010,第11卷第3056-3065页.
胡雷刚等.基于随机遗传算法的并行测试任务调度研究.《电测与仪表》.2008,第45卷(第514期),第41-45页.

审查员 杨龙

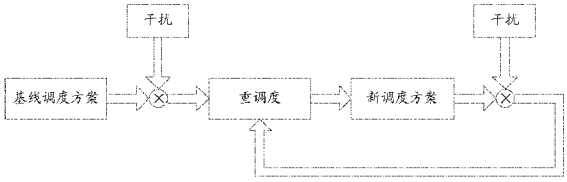
权利要求书2页 说明书11页 附图4页

(54)发明名称

一种基于遗传算法的多航天器并行测试任务调度方法

(57)摘要

本发明涉及一种基于遗传算法的多航天器并行测试任务调度方法,其步骤如下:第一步遗传算法;第二步获得基线调度方案;第三步重调度;第四步设计结束。本发明的优点具体包括:①考虑了不确定性因素的影响,调度方案能够始终保持可行性;②生成的调度方案具有较短的测试总工期;③能够有效处理大规模的航天器并行测试任务调度问题。



CN 102122251 B

1.一种基于遗传算法的多航天器并行测试任务调度方法，其特征在于：步骤如下：

第一步 遗传算法

使用遗传算法求解多航天器并行测试任务调度模型时，其运算步骤如下：

(1)编码方式

编码采用基于任务列表的带有优先级的编码方式；

(2)解码方式

解码采用串行解码方法：将染色体按照从左到右的顺序，依次确定每个测试任务的最早开始时刻，该最早开始时刻满足：①不小于其紧前任务结束时刻；②该任务在整个执行阶段满足资源约束；③满足条件①和②的最小时刻；按照上述解码方式，每个测试任务的最早开始时刻，为某个已确定开始时刻的测试任务的结束时刻；

(3)种群初始化

初始种群可以采用随机的方式产生，也可以依据优先规则产生，或者将二者结合以得到更加分散的个体；

(4)选择算子

选择算子采用2-联赛选择机制；

根据2-联赛选择机制，每次随机选择两条染色体比较其测试总工期，选择测试总工期较短的那条染色体，若两条染色体的测试总工期相同则随机选择其中一条，直到满足种群规模；

(5)交叉算子

交叉算子采用一点交叉方式；

(6)变异算子

变异算法采用插入变异方式；

第二步 获得基线调度方案

根据已知的经验或估计数据，使用第一步提出的遗传算法求解多航天器并行测试任务调度模型以获得基线调度方案；

第三步 重调度

重调度采用三种方式：

(1)右移策略；

(2)完全重调度；

(3)部分重调度；

第四步 设计结束

当调度方案一旦被打破时，则立即进行重新调度，直到测试结束；决策者可以根据具体情况，综合选择重调度方式。

2.根据权利要求1所述的一种基于遗传算法的多航天器并行测试任务调度方法，其特征在于：所述第一步骤(1)中，编码方式设计如下：

假设N艘航天器的优先级是递减的，那么所有航天器的测试任务可以按如下方式编码：

$$\pi=[A_{10},\cdots,A_{1(1_J-1)},A_{20},\cdots A_{2(2_J-1)},\cdots,A_{N0},\cdots,A_{N(N_J-1)}]$$

其中，$A_{i0},\cdots,A_{i(i_J-1)}$表示航天器i的全部测试任务，i越大则航天器的优先级越小；A_{i0}和

$A_{i(j-1)}$的位置始终保持不变，设共有J个测试任务，其余任务在满足时序约束的前提下可以任意改变位置；这种编码方式是按照航天器的优先级分段编码的，即将优先级高的航天器的测试任务排在前面。

一种基于遗传算法的多航天器并行测试任务调度方法

技术领域

[0001] 本发明涉及一种基于遗传算法的多航天器并行测试任务调度方法，特别是一种在不确定环境下的基于遗传算法的多航天器并行测试任务调度方法。本发明属于并行测试技术领域。

背景技术

[0002] 近年来，我国航天技术发展迅猛，航天技术开始从国防军事领域迅速向民用航天领域扩展，我国拥有的航天器数目急剧增多，航天器的研制和生产开始步入批量化阶段。而传统的航天器测试采用的是串行测试方式——测试效率低，资源利用率低，测试成本高，这种测试方式已不能适应航天器批量化生产的要求。在这种情况下，迫切需要改变现有的航天器测试模式，实现航天器的并行测试，以提高测试效率和资源利用率，同时降低测试成本。航天器并行测试是指自动测试系统在同一时间内完成多项航天器测试任务，包括在同一时间内完成对多艘航天器的测试，或者在单艘航天器上异步或同步执行多个测试任务并同时完成对航天器多项参数的测量。在航天器测试设备、测试人员等资源有限的条件下，航天器的并行测试是航天器综合测试的必然趋势，也是航天器批量化生产的迫切需要。实现航天器的并行测试可以大大精简测试队伍，提高测试质量，缩短测试周期。

[0003] 任务调度是并行测试技术的核心，它要求在给定资源约束和任务时序约束的前提下寻求满足目标要求的调度方案。多航天器并行测试任务调度可以描述为：有N艘航天器同时进行测试，第i艘航天器具有i_J个测试任务，其中包括两个虚拟任务，分别用来标志测试的开始和结束，它们不占用任何资源，测试时间为零。这些测试任务满足如下的性质：

[0004] 1、同艘航天器的测试任务之间具有时序约束关系，即每个测试任务只有当其紧前任务完成后才可以开始；

[0005] 2、不同航天器的测试任务之间没有时序约束关系，但是不同的航天器具有不同的优先级，即优先级高的航天器的测试任务具有优先使用测试资源的权利；

[0006] 3、测试任务一旦开始不能被中断；

[0007] 4、全部测试任务共需要K种可更新的测试资源(可更新的测试资源是指测试资源一旦被释放可以立即被其他测试任务使用，在整个并行测试过程中其可用量不变)，每种测试资源的总量为R_k；

[0008] 5、第i艘航天器的第j个测试任务A_{ij}的测试时间为d_{ij}，对第k种资源的需求量为r_{ijk}；

[0009] 6、并行测试任务调度的目的是在满足资源约束、时序约束、优先级关系和优化指标(通常是总工期最短)的前提下，确定各个测试任务的开始时间和结束时间。

[0010] 根据以上要求，多航天器并行测试任务调度问题可以用以下数学模型描述：

[0011] $$\min\{\max_{1\le i\le N} f_{ii_J}\} \tag{1}$$

[0012] s.t

[0013] $$S_{ij}\ge f_{uv}, i=1,\cdots,N, j=0,\cdots,i_J-1, \forall A_{uv}\in P_{ij} \tag{2}$$

[0014] $$\sum_{A_{ij}\in I_t} r_{ijk} \le R_k, i=1,\cdots,N, j=0,\cdots,i_J-1, k=1,\cdots,K \quad (3)$$

[0015] $f_{ij}=S_{ij}+d_{ij}, i=1,\cdots,N, j=0,\cdots,i_J-1$ (4)

[0016] $S_{ij}\ge 0, i=1,\cdots,N, j=0,\cdots,i_J-1, k=1,\cdots,K$ (5)

[0017] 其中，S_{ij}和f_{ij}分别表示测试任务A_{ij}的开始和结束时间，P_{ij}表示A_{ij}的紧前任务集合，I_t表示在t时刻正在执行的测试任务集合。

[0018] 在该模型中，目标函数(1)要求总工期最短，式(2)表示任务的时序约束，一个测试任务的开始时间不小于其紧前任务完成时间，式(3)表示资源约束，任何时刻所使用的资源量小于其总量，式(5)表示表示非负约束，即调度在0时刻开始。

[0019] 传统的并行测试任务调度方法是一种确定环境下的任务调度方法，即认为测试是在理想情况下进行的——模型中的所有参数是确定且不变的，那么对该确定型模型进行求解，即可得到最优或较优的调度方案。但是，实际的测试却是在一个动态的不确定的环境中进行的，往往存在大量的不确定性因素，如任务测试时间变化、任务增加或取消、任务提前或延迟和资源可用量变化等，这些不确定性因素经常导致事先确定的调度方案无法按照计划进行，进而导致测试中断或测试延期。这说明，在实际测试时模型的参数是不断变化的，而模型参数的变化往往伴随着解(调度方案)的改变，导致原来的可行解可能变得不可行，原来的最优解可能变成较差解，这使传统确定环境下的并行测试任务调度方法具有很大的局限性。因此，调度方案应该根据模型的参数变化进行不断的调整，以保持调度方案的可行性和较短的测试总工期。因此，考虑不确定环境下的多航天器并行测试任务调度方法才更有实际的意义，但这也极大地增加了求解的难度，本发明即是为解决这个问题而提出的。

[0020] 多航天器并行测试任务调度的特点突出表现为航天器具有优先级关系，测试任务多(规模大)，资源约束复杂和不确定性大等，这对航天器并行测试任务调度提出了相当大的挑战。为解决该问题并克服传统并行测试任务调度方法的局限性，针对多航天器并行测试任务调度的特点，本发明提出一种不确定环境下的基于遗传算法的多航天器并行测试任务调度方法。

发明内容

[0021] 本发明一种基于遗传算法的多航天器并行测试任务调度方法，其目的是：针对多航天器并行测试任务调度具有优先级关系，规模大，资源约束复杂和不确定大的特点，提出一种不确定环境下的基于遗传算法的并行测试任务调度方法以获得实际可行的调度方案，从而克服传统并行测试任务调度方法的局限性，达到航天器并行测试资源优化配置的目的，进而提高测试效率和降低测试成本。

[0022] 本发明一种基于遗传算法的多航天器并行测试任务调度方法，其设计思想是：首先使用遗传算法根据经验或估计数据求解模型以获得一个基线调度方案，在实际测试中调度方案一旦被打破立即对剩余的测试任务进行重新调度从而获得一个新的调度方案，如此不断更新调度方案以保证调度方案的可行性和较短的测试总工期，直到测试完成，如图1所示。

[0023] 基于上述思想，下面具体介绍本发明的技术方案，具体设计步骤如下：

[0024] 第一步遗传算法

[0025] 本发明使用遗传算法求解多航天器并行测试任务调度模型。使用遗传算法求解该问题的关键在于设计编码方式、解码方式、种群初始化、遗传算子(选择算子、交叉算子和变异算子)等,遗传算法的基本流程如图2所示,具体设计如下:

[0026] 1、编码方式

[0027] 编码采用基于任务列表的带有优先级的编码方式,它将一个调度方案表示为一个任务序列。假设N艘航天器的优先级是递减的,那么所有航天器的测试任务可以按如下方式编码:

[0028] $\pi=[A_{10},\cdots,A_{1(1_J-1)},A_{20},\cdots A_{2(2_J-1)},\cdots,A_{N0},\cdots,A_{N(N_J-1)}]$

[0029] 其中,$A_{i0},\cdots,A_{i(i_J-1)}$表示航天器i的全部测试任务,i越大则航天器的优先级越小。A_{i0}和$A_{i(i_J-1)}$(虚拟任务)的位置始终保持不变,其余任务在满足时序约束的前提下可以任意改变位置。

[0030] 可见,这种编码方式是按照航天器的优先级分段编码的,即将优先级高的航天器的测试任务排在前面。只要保证每艘航天器首尾两个虚拟测试任务的编码位置不变,无论其他测试任务的位置如何变化(在符合时序约束的前提下),当从左到右地调度任务时,总能保证优先级高的航天器的测试任务首先被安排执行,这样就保证了航天器的优先级关系不会被打破。这种分级的编码方式除了可以保证航天器的优先级关系外,还极大的缩小到了搜索空间,因为它限制了每个任务位置变化的范围。

[0031] 2、解码方式

[0032] 解码是编码的逆过程,它将染色体转化为调度方案,从而计算出目标函数值。根据编码特点,本发明采用串行解码方法,它可以得到积极的调度方案。所谓积极的调度方案是指,任何测试任务都不可能在不改变其它测试任务开始时刻的前提下更早的开始。

[0033] 将一条染色体解码实际上就是确定各个测试任务的开始时刻。由于每条染色体编码都已满足航天器的优先级关系和测试任务的时序约束,那么解码依据的规则主要是资源约束关系。将染色体按照从左到右的顺序,依次确定每个测试任务的最早开始时刻,该最早开始时刻满足:①不小于其紧前任务结束时刻;②该任务在整个执行阶段满足资源约束;③满足条件①和②的最小时刻。按照上述解码方式,每个测试任务的最早开始时刻,为某个已确定开始时刻的测试任务的结束时刻。

[0034] 设共有J个测试任务,π为由J个测试任务编码而成的染色体,π_g($0\le g<J$)表示其第g个基因,S_g和f_g分别表示第g个基因的开始和结束时刻,d_g表示第g个基因的测试时间,L为由所有当前已确定开始时刻的测试任务的结束时刻组成的递增时间序列,L_i为其第i个元素。串行解码的流程如下:

[0035] (1)令$g=1$,$S_0=0$,$f_0=0$,$L=\{0\}$;

[0036] (2)计算π_g所有紧前任务最晚结束时刻t_g,并确定t_g在L中的位置i;

[0037] (3)若当$t\in[L_i,L_i+d_g]$时π_g满足资源约束,则令$S_g=L_i$,$f_g=S_g+d_g$,$L=L\cup\{f_g\}$,并对T进行递增排序,转(4),否则令$i=i+1$,转(3);

[0038] (4)$g=g+1$,若$g<J$,则转(2),否则转(5);

[0039] (5)结束。

[0040] 3、种群初始化

[0041] 初始种群应该保证充分的多样性，以减小计算陷入局部极小的可能性。初始种群可以采用随机的方式产生，也可以依据优先规则产生，或者将二者结合以得到更加分散的个体。无论哪种产生方式，都必须保证航天器优先级关系和测试任务时序约束不被打破。

[0042] 假设共有J个任务$\{A_0, A_1, \cdots, A_{J-1}\}$，则编码可以分为J-1个阶段，每个阶段g对应一个未完成编码的染色体π和一个可行任务集合D_g，D_g包含在当前阶段所有未被安排且其所有紧前任务已包含在π中的任务。当一艘航天器的测试任务编码完毕后，其末任务看作是下一艘应被编码的航天器的首任务的紧前任务任务，这样就能将各航天器按优先级顺序依次编码。在每一阶段，随机或根据一定的优先规则从D_g中选择一个任务，在满足资源约束和其它约束的情况下加入π中。编码流程如下：

[0043] (1)令g＝0，π＝$[A_0]$；

[0044] (2)计算可行活动集合D_g，在D_g中选择(随机或依优先规则)一个任务A_j，令π＝$[A_0, \cdots, A_j]$；

[0045] (3)g＝g+1，若g＜J-1，则转(2)，否则转(4)；

[0046] (4)结束。

[0047] 另外，虽然较大的种群规模可以增加种群的多样性，但是过大的种群规模可能使较优个体过早占据种群从而造成过早收敛，而且会增加计算负担。因此，种群规模应该适中。

[0048] 4、选择算子

[0049] 选择算子采用2-联赛选择机制，即每次随机选择两条染色体选取其中测试总工期较短的一条，直到满足种群规模。

[0050] 5、交叉算子

[0051] 交叉算子采用一点交叉方式。设两条交叉的染色体分别为：

[0052] $\pi^F = [A_0^F, A_1^F, A_2^F, \cdots A_M^F]$

[0053] $\pi^G = [A_0^G, A_1^G, A_2^G, \cdots A_M^G]$

[0054] 首先，随机产生一个正整数r(0≤r＜M)作为交叉点，交叉产生的子代染色体π^D的前r+1个基因继承自π^F，后M-r-1个基因继承自π^G并保持在π^G中的相对顺序不变，即

[0055] $\pi^D = [A_0^F, A_1^F, \cdots, A_r^F, A_{(r+1)'}^G \cdots, A_{M'}^G]$

[0056] 其中，$A_{k'}^G \notin \{A_0^F, A_1^F, \cdots, A_r^F\}$，k＝r+1，…，M。

[0057] 另一个子代染色体π^S的产生方式与此相似，即

[0058] $\pi^S = [A_0^G, A_1^G, \cdots, A_r^G, A_{(r+1)'}^F \cdots, A_{M'}^F]$

[0059] 其中，$A_{k'}^F \notin \{A_0^G, A_1^G, \cdots, A_r^G\}$，k＝r+1，…，M。

[0060] 这种交叉方式不会改变时序约束和各艘航天器的优先级。

[0061] 6、变异算子

[0062] 变异算子采用插入变异方式。对某个任务进行变异操作时，首先找到其所有紧前任务最后位置r_1和所有紧后任务的最前位置r_2，然后随机产生一个正整数r(r_1＜r≤r_2)，将该任务插入位置r上。每艘航天器的首任务的紧前任务认为是其自身，而末任务的紧后任务也认为是其自身。这种插入变异方式也不会改变任务的时序约束和各艘航天器的优先级。

[0063] 第二步获得基线调度方案

[0064] 基线调度方案是在测试开始前根据经验或估计数据获得的，它可以视为对整个测试过程的一个初步计划。显然，经验或估计数据越准确，基线调度方案的可行性就越好；反之，经验或估计数据越不准确，基线调度方案的可行性就越差，其被打破的可能性就越大。因此，模型中参数的准确估计对保证调度方案的可行性是非常重要的，而合适的求解算法则是保证调度方案可行性的另一个重要方面。

[0065] 在测试前取得多航天器并行测试的相关数据后，如测试时序约束、航天器优先级、资源约束和每个任务的测试时间等，就可以采用本发明第一步提出的遗传算法求解背景技术中介绍的多航天器并行测试任务调度模型了。首先，根据测试时序约束和航天器的优先级进行染色体的编码，此时就将任务调度问题转化为了遗传算法可以处理的形式；经过遗传算法的遗传算子等的操作和足够的迭代次数便会产生一个最优的染色体；最后通过解码将该染色体解码为调度方案，此时就将遗传算法的求解结果还原成了调度问题的解，这个解就被称为基线调度方案，因为它是在测试开始前确定的，需要在测试开始后不断地进行调整以保持可行性。

[0066] 第三步重调度

[0067] 为了对实际测试过程中遇到的各种干扰因素，如任务测试时间变化、任务增加或取消、任务提前或延迟和可用资源量变化等，及时地做出反应以调整调度方案，使调度方案始终保持可行性以及较短的测试总工期，需要对其进行重调度。重调度可以采用三种方式：

[0068] 1、右移策略

[0069] 右移策略是指当干扰一旦发生且调度方案无法执行时，将属于同一航天器的未开始的任务按照其开始时刻进行升序排列，然后按优先级从高到低的顺序依次重新安排（重新解码）每艘航天器的测试任务的开始时刻。右移策略是最简单的重调度方法，由于没有进行优化，故得到的调度方案不一定是最优的或较优的，可能会使总工期增加较多。右移策略的好处是，一般情况下越是接近重调度时刻的测试任务其开始时刻的改变越小，而且各个测试任务的执行顺序变化也较小，这有利于原来的测试准备工作（如准备测试任务的场地、仪器设备和人员等）不被打破。当干扰因素较少或其影响较小时可以采用这种方式。

[0070] 2、完全重调度

[0071] 完全重调度是指当干扰一旦发生且调度方案无法执行时，将所有未开始的测试任务使用第一步提出的遗传算法进行重新调度。完全重调度可以保证重调度后得到的调度方案是最优的或较优的，即具有最短或较短的测试总工期。但是完全重调度可能会造成部分测试任务（尤其是靠近重调度时刻的测试任务）的开始时刻变化较大，而且测试任务的执行顺序也可能发生较大的变化，这就增大了调度方案的调整成本。当测试对总工期的要求较为严格时，可以采用这种方式。

[0072] 3、部分重调度

[0073] 部分重调度是指当干扰一旦发生且调度方案无法执行时（设该时刻为T），只对开始时刻在[T，T+ΔT)内的测试任务使用右移策略进行重新调度，对开始时刻不小于T+ΔT的测试任务则使用第一步提出的遗传算法进行重新调度。部分重调度相当于加了一个时间窗口ΔT，对时间窗口内的测试任务进行右移操作以使靠近重调度时刻T的一段时间内测试任务的开始时刻和执行顺序变化较小，对剩余的测试任务使用遗传算法进行优化则是为了保

证调度方案有较短的测试总工期。部分重调度在右移策略和完全重调度之间做出了折中，兼具二者的优点，因此它得到的调度方案有较短的测试总工期和较低的调整成本。

[0074] 需要注意的是，在重调度时应该保持测试任务的不可回溯性，即还未开始执行的测试任务的开始时刻在重调度后不能早于重调度时刻。

[0075] 第四步设计结束

[0076] 整个设计过程的第一步和第二步是在测试开始前完成的，第三步其实是一个动态的反复进行的过程，一直持续到测试的完成。围绕保持调度方案的可行性和较短的测试总工期这两个指标，本发明首先在第一步提出了多航天器并行测试任务调度模型的遗传算法求解步骤，这是本发明的核心算法，无论是基线调度方案还是重调度方案都是使用它产生的；第二步给出了基线调度方案的产生方法，如果在测试时没有发生干扰那么它其实就是最终的调度方案，如果发生了干扰那么将对调度方案进行重调度；第三步给了三种重调度方法以应对不同的测试要求，可以灵活选择，重调度伴随着整个测试过程。通过以上三个步骤能够使测试工作顺利完成并较好地达到指标要求，设计结束。

[0077] 本发明一种基于遗传算法的多航天器并行测试任务调度方法，其优点及功效是：本发明能够有效解决具有优先级关系、规模大、资源约束复杂和不确定性大等特点的多航天器并行测试任务调度问题，由于充分考虑了不确定性因素的影响，使在出现干扰的情况下也能始终保持调度方案的可行性和较短的测试总工期，从而克服了传统并行测试任务调度方法的局限性，达到了提高资源利用率和测试效率及降低测试成本的效果。其优点具体包括：①考虑了不确定性因素的影响，调度方案能够始终保持可行性；②生成的调度方案具有较短的测试总工期；③能够有效处理大规模的航天器并行测试任务调度问题。

附图说明

[0078] 图1本发明航天器并行测试任务调度方法的基本流程；

[0079] 图2本发明遗传算法流程图；

[0080] 图3本发明实施例的每艘航天器的测试时序约束图；

[0081] 图4本发明实施例的基线调度方案；

[0082] 图5本发明实施例的使用右移策略得到的重调度方案；

[0083] 图6本发明实施例的完全重调度时航天器的测试时序约束图；

[0084] 图7本发明实施例的使用完全重调度得到的重调度方案；

[0085] 图8本发明实施例的部分重调度进行优化时航天器的测试时序约束图；

[0086] 图9本发明实施例的使用部分重调度得到的重调度方案；

[0087] 对图中符号说明如下：

[0088] 图3中圆圈代表测试任务，箭头代表时序约束；

[0089] 图4中横坐标为测试时间，纵坐标为航天器的编号，方块代表测试任务，方块的长度代表测试时间，每个方块的左边界为其开始时刻，右边界为其结束时刻；

[0090] 图5中横坐标为测试时间，纵坐标为航天器的编号，方块代表测试任务，方块的长度代表测试时间，每个方块的左边界为其开始时刻，右边界为其结束时刻；

[0091] 图6中圆圈代表测试任务，箭头代表时序约束；

[0092] 图7中横坐标为测试时间，纵坐标为航天器的编号，方块代表测试任务，方块的长

度代表测试时间，每个方块的左边界为其开始时刻，右边界为其结束时刻；

[0093] 图8中圆圈代表测试任务，箭头代表时序约束；

[0094] 图9中横坐标为测试时间，纵坐标为航天器的编号，方块代表测试任务，方块的长度代表测试时间，每个方块的左边界为其开始时刻，右边界为其结束时刻。

具体实施方式

[0095] 本发明通过3艘相同航天器的并行测试任务调度问题的简单实例来说明其具体实施方式。每艘航天器具有8个测试任务，其第0个和第7个任务为虚拟测试任务，每艘航天器的测试时序约束如图3所示，每艘航天器的测试任务的测试时间、资源需求及资源总量如表1所示。航天器1的优先级最高，航天器2的优先级次之，航天器3的优先级最低。

[0096]

任务编号	测试时间	资源 1 需求量	资源 2 需求量	资源 3 需求量	资源 4 需求量
0	0	0	0	0	0
1	5	2	4	4	2
2	5	4	0	4	0
3	9	2	0	5	2
4	4	2	2	6	4
5	8	6	4	0	2
6	7	4	6	3	5
7	0	0	0	0	0
资源总量		10	8	12	8

[0097] 表1

[0098] 根据以上数据，本发明的具体实施步骤如下：

[0099] 第一步遗传算法

[0100] 根据发明内容中第一步提出的遗传算法，实现对多航天器并行测试任务调度模型的求解。第二步和第三步中的基线调度和重调度都是使用该算法进行求解的。其具体实现步骤如下：

[0101] (1)编码方式

[0102] 编码需要符合测试任务的时序约束和航天器的优先级关系。按照本发明提出的编码规则，在该实例中染色体可以编码为如下形式：

[0103] π=[10,11,12,14,13,15,16,17,20,21,22,24,23,25,26,27,30,31,32,34,33,35,36,37]

[0104] 其中，染色体的每个基因的第1位表示航天器编号，第2位表示任务编号。

[0105] (2)解码方式

[0106] 按照发明内容第一步(2)中提出的串行解码步骤，确定所有染色体的测试任务的开始时刻，并计算其测试总工期(航天器3的最后一个虚拟任务的开始时刻即为测试总工

期)。下面给出其具体计算步骤,设某条染色体为:

[0107] π=[10,11,12,14,13,15,16,17,20,21,23,22,24,25,26,27,30,31,33,32,35,34,36,37]

[0108] 首先,令基因10的开始时间$S_{10}=0$,结束时间$f_{10}=0$,时间序列L={0};基因11的紧前任务的最晚结束时刻为基因10的结束时刻$L_0=0$,当$t\in[L_0,L_0+d_{11}]$时满足资源约束,基因11可以执行,$S_{11}=0$,$f_{11}=5$,L={0,5};基因12的紧前任务的最晚结束时刻为基因11的结束时刻$L_1=5$,当$t\in[L_1,L_1+d_{12}]$时满足资源约束,基因12可以执行,$S_{12}=5$,$f_{12}=10$,L={0,5,10};基因14的紧前任务的最晚结束时刻为基因12的结束时刻$L_2=10$,当$t\in[L_2,L_2+d_{14}]$时满足资源约束,基因14可以执行,$S_{14}=10$,$f_{14}=14$,L={0,5,10,14};基因13的紧前任务的最晚结束时刻为基因11的结束时刻$L_1=5$,当$t\in[L_1,L_1+d_{13}]$时满足资源约束,基因13可以执行,$S_{13}=5$,$f_{13}=14$,L={0,5,10,14};基因15的紧前任务的最晚结束时刻为基因13的结束时刻$L_3=14$,当$t\in[L_3,L_3+d_{15}]$时满足资源约束,基因15可以执行,$S_{15}=14$,$f_{15}=22$,L={0,5,10,14,22};基因16的紧前任务的最晚结束时刻为基因15的结束时刻$L_4=22$,当$t\in[L_4,L_4+d_{16}]$时满足资源约束,基因16可以执行,$S_{16}=22$,$f_{16}=29$,L={0,5,10,14,22,29};基因17的紧前任务的最晚结束时刻为基因16的结束时刻$L_5=29$,当$t\in[L_5,L_5+d_{17}]$时满足资源约束,基因17可以执行,$S_{17}=29$,$f_{17}=29$,L={0,5,10,14,22,29};基因20的开始时间$S_{20}=0$,结束时间$f_{20}=0$,L={0,5,10,14,22,29};基因21的紧前任务的最晚结束时刻为基因20的结束时刻$L_0=0$,当$t\in[L_0,L_0+d_{21}]$时满足资源约束,基因21可以执行,$S_{21}=0$,$f_{21}=5$,L={0,5,10,14,22,29};基因23的紧前任务的最晚结束时刻为基因21的结束时刻$L_1=5$,当$t\in[L_3,L_3+d_{23}]$时最早满足资源约束,这时基因23可以执行,$S_{23}=14$,$f_{23}=23$,L={0,5,10,14,22,23,29};基因22的紧前任务的最晚结束时刻为基因21的结束时刻$L_1=5$,当$t\in[L_4,L_4+d_{22}]$时最早满足资源约束,这时基因22可以执行,$S_{22}=22$,$f_{22}=27$,L={0,5,10,14,22,23,27,29};基因24的紧前任务的最晚结束时刻为基因22的结束时刻$L_6=27$,当$t\in[L_7,L_7+d_{24}]$时最早满足资源约束,这时基因24可以执行,$S_{24}=29$,$f_{24}=33$,L={0,5,10,14,22,23,27,29,33};基因25的紧前任务最晚结束时刻为基因23的结束时刻$L_5=23$,当$t\in[L_7,L_7+d_{25}]$时最早满足资源约束,这时基因25可以执行,$S_{25}=29$,$f_{25}=37$,L={0,5,10,14,22,23,27,29,33,37};基因26的紧前任务最晚结束时刻为基因25的结束时刻$L_9=37$,当$t\in[L_9,L_9+d_{26}]$时满足资源约束,这时基因26可以执行,$S_{26}=37$,$f_{26}=44$,L={0,5,10,14,22,23,27,29,33,37,44};基因27的紧前任务最晚结束时刻为基因26的结束时刻$L_{10}=44$,当$t\in[L_{10},L_{10}+d_{27}]$时满足资源约束,这时基因27可以执行,$S_{27}=44$,$f_{27}=44$,L={0,5,10,14,22,23,27,29,33,37,44};同理可计算出其余基因(测试任务)的开始和结束时间。

[0109] (3)种群初始化

[0110] 按照发明内容第一步(3)中提出的步骤随机产生100条满足测试任务时序约束和航天器优先级关系的染色体,这里种群的规模设为100。下面给出其具体计算步骤:

[0111] 首先,令π=[10],则D_g={11};在D_g中随机选择一个任务11,则π=[10,11],D_g={12,13};在D_g中随机选择一个任务13,则π=[10,11,13],D_g={12,15};在D_g中随机选择一个任务12,则π=[10,11,13,12],D_g={14,15};在D_g中随机选择一个任务14,则π=[10,11,13,12,14],D_g={15};在D_g中随机选择一个任务15,则π=[10,11,13,12,14,15],D_g={16};在D_g中随机选择一个任务16,则π=[10,11,13,12,14,15,16],D_g={17};在D_g中随机选择一

个任务17，则π＝[10,11,13,12,14,15,16,17]，D_g＝{20}；同理可对航天器2和航天器3的测试任务进行编码，最后得到染色体：

[0112] π＝[10,11,13,12,14,15,16,17,20,21,22,23,24,25,26,27,30,31,33,32,35,34,36,37,]。

[0113] 重复上述过程100次，即可得到规模为100的初始种群。

[0114] (4)选择算子

[0115] 根据2-联赛选择机制，每次随机选择两条染色体比较其测试总工期，选择测试总工期较短的那条染色体，若两条染色体的测试总工期相同则随机选择其中一条，直到满足种群规模。

[0116] (5)交叉算子

[0117] 设两条交叉的染色体分别为：

[0118] π^F＝[10,11,12,14,13,15,16,17,20,21,22,24,23,25,26,27,30,31,33,35,32,34,36,37]

[0119] π^G＝[10,11,13,12,15,14,16,17,20,21,22,24,23,25,26,27,30,31,32,34,33,35,36,37]

[0120] 若随机产生的交叉点为2，那么子代染色体π^S的前3个基因继承自π^F，其余基因继承自π^G并保持顺序不变；子代染色体π^D的前3个基因继承自π^G，其余基因继承自π^F并保持顺序不变，由此可以得到：

[0121] π^S＝[10,11,12,13,15,14,16,17,20,21,22,24,23,25,26,27,30,31,32,34,33,35,36,37]

[0122] π^D＝[10,11,13,12,14,15,16,17,20,21,22,24,23,25,26,27,30,31,33,35,32,34,36,37]

[0123] (6)变异算子

[0124] 设某条染色体为：

[0125] π＝[10,11,12,13,14,15,16,17,20,21,22,24,23,25,26,27,30,31,32,34,33,35,36,37]要对基因15进行变异，那么其所有紧前任务最后位置r_1＝3，所有紧后任务的最前位置r_2＝6。若随机产生正整数r＝4，那么将基因15插入到位置4上，可以得到：

[0126] π＝[10,11,12,13,15,14,16,17,20,21,22,24,23,25,26,27,30,31,32,34,33,35,36,37]

[0127] (7)其它参数

[0128] 算法的其它参数设置如下：交叉率0.9，变异率0.1，最大运行代数(停止条件)500。

[0129] 第二步获得基线调度方案

[0130] 根据图3的时序约束、表1中的数据以及航天器的优先级关系，按照第一步描述的遗传算法的各步骤进行计算，得到的调度方案即为基线调度方案，如图4所示。

[0131] 第三步重调度

[0132] 假设在基线调度方案的执行过程中，航天器1的第4个测试任务的测试时间发生变化，其测试时间增加4，即在T＝14时基线调度方案被破坏，通过三种重调度方式产生的新调度方案如下：

[0133] (1)右移策略

[0134] 航天器1的第4个测试任务的测试时间增加4后，由图4的基线调度方案可以看出需要重新调度的任务序列为(每艘航天器的测试任务按其开始时间进行升序排列)：

[0135] [15,16,17,23,22,24,25,26,27,31,33,32,34,35,36,37]。

[0136] 根据解码方式对上述任务序列进行重新解码。首先，令基因14的结束时间$f_{14}=18$，再将重调度时刻和基因14的结束时刻加入到时间序列L中得到L={14,18}；基因15的紧前任务的最晚结束时刻为基因13的结束时刻$L_0=14$且不小于14，当$t\in[L_0,L_0+d_{15}]$时满足资源约束，基因15可以执行，$S_{15}=14$，$f_{15}=22$，L={14,18,22}；基因16的紧前任务的最晚结束时刻为基因15的结束时刻$L_2=22$且不小于14，当$t\in[L_2,L_2+d_{16}]$时满足资源约束，基因16可以执行，$S_{16}=22$，$f_{16}=29$，L={14,18,22,29}；基因17的紧前任务的最晚结束时刻为基因16的结束时刻$L_3=29$且不小于14，当$t\in[L_3,L_3+d_{17}]$时满足资源约束，基因17可以执行，$S_{17}=29$，$f_{17}=29$，L={14,18,22,29}；基因23的紧前任务的最晚结束时刻为基因21的结束时刻5，小于14，那么基因23开始调度的时刻应变为$L_0=14$，当$t\in[L_0,L_0+d_{23}]$时满足资源约束，基因23可以执行，$S_{23}=14$，$f_{23}=23$，L={14,18,22,23,29}；同理可以确定其余基因(测试任务)的开始和结束时刻。最后重调度生成的调度方案如图5所示。

[0137] (2)完全重调度

[0138] 航天器1的第4个测试任务的测试时间增加4后，由图4的基线调度方案可以看出三艘航天器的测试时序约束变为了如图6所示的关系。根据图6的时序约束、表1中的数据以及航天器的优先级关系，使用在发明内容第一步中提出的遗传算法进行求解，得到的完全重调度方案如图7所示。

[0139] 需要注意的是，进行重新调度的开始时刻应为14，且每个测试任务的开始时刻不能小于其紧前任务(即使该紧前任务不在新的测试时序约束图中)的开始时刻，当$t\in[14,18]$时，每种资源的总量应该为原来的资源总量减去遗传算法优化前已安排完但是还没有执行或没有执行完的测试任务对每种资源的需求量。

[0140] (3)部分重调度

[0141] 航天器1的第4个测试任务的测试时间增加4后，令$\Delta T=10$，由图4的基线调度方案可以看出此时需要使用右移策略进行重新调度的任务序列为(每艘航天器的测试任务按其开始时间进行升序排列)：

[0142] [15,16,23,22,31,33]。

[0143] 首先，令基因14的结束时间$f_{14}=18$，再将重调度时刻和基因14的结束时刻加入到时间序列L中得到L={14,18}；基因15的紧前任务的最晚结束时刻为基因13的结束时刻$L_0=14$且不小于14，当$t\in[L_0,L_0+d_{15}]$时满足资源约束，基因15可以执行，$S_{15}=14$，$f_{15}=22$，L={14,18,22}；基因16的紧前任务的最晚结束时刻为基因15的结束时刻$L_2=22$且不小于14，当$t\in[L_2,L_2+d_{16}]$时满足资源约束，基因16可以执行，$S_{16}=22$，$f_{16}=29$，L={14,18,22,29}；基因23的紧前任务的最晚结束时刻为基因21的结束时刻5，小于14，那么基因23开始调度的时刻应变为$L_0=14$，当$t\in[L_0,L_0+d_{23}]$时满足资源约束，基因23可以执行，$S_{23}=14$，$f_{23}=23$，L={14,18,22,23,29}；基因22的紧前任务的最晚结束时刻为基因21的结束时刻5，小于14，那么基因22开始调度的时刻应变为$L_0=14$，当$t\in[L_2,L_2+d_{22}]$时最早满足资源约束，此时基因22可以执行，$S_{22}=22$，$f_{22}=27$，L={14,18,22,23,27,29}；基因31的紧前任务的最晚结束时刻为基因30的结束时刻0，小于14，那么基因31开始调度的时刻应变为$L_0=14$，当$t\in[L_5,$

L_5+d_{31}]时最早满足资源约束，基因31可以执行，$S_{31}=29$，$f_{31}=34$，$L=\{14,18,22,23,27,29,34\}$；基因33的紧前任务的最晚结束时刻为基因31的结束时刻$L_6=34$且不小于14，当$t\in[L_6, L_6+d_{33}]$时满足资源约束，基因33可以执行，$S_{33}=34$，$f_{33}=43$，$L=\{14,18,22,23,27,29,34,43\}$。然后使用遗传算法对剩余的测试任务进行优化，剩余测试任务的时序约束如图8所示。最后得到的使用部分重调度生成的调度方案如图9所示。

[0144] 需要注意的是，使用遗传算法对剩余测试任务重新调度的开始时刻应为14，且每个测试任务的开始时刻不能小于其紧前任务(即使该紧前任务不在新的测试时序约束图中)的开始时刻，当$t\in[14,46]$时，每种资源的总量是变化的，其应该为原来的资源总量减去遗传算法优化前已安排完但是还没有执行或没有执行完的测试任务对每种资源的需求量。

[0145] 第四步设计结束

[0146] 当调度方案一旦被打破时，则立即进行重新调度，以保持调度方案的可行性，直到测试结束。至于重调度时采用何种重调度方式，应该根据具体情况，如关心的指标、每种重调度方式调度结果的差异等，由决策者综合选择。

[0147] 将以上步骤在VC2008中使用C#编程实现，仿真结果如图4、图5、图7和图9所示。

[0148] 图4是在经验或估计数据的基础上通过使用遗传算法优化得到的基线调度方案，其总工期为59。

[0149] 由图5可以看出，使用右移策略进行重调度，当航天器1的第4个测试任务的测试时间增加4时，调度方案的总工期由59变为73，其总工期的增加量大于测试任务测试时间的增加量。但是靠近重调度时刻的一段时间内的测试任务的开始时刻和执行顺序变化较小，如航天器1的第5个测试任务和航天器2的第3个测试任务，这有利于原来的准备工作尽量不被打破。

[0150] 由图7可以看出，使用完全重调度，当航天器1的第4个测试任务的测试时间增加4时，调度方案的总工期由59变为62。此时，不仅总工期的增加量小于测试任务测试时间的增加量，而且总工期明显小于使用右移策略得到的总工期。但是也可以看出，靠近重调度时刻的一段时间内的测试任务的开始时刻和执行顺序变化较大，如航天器2的第2、3个测试任务和航天器3的第1个测试任务，这可能会使原来的准备工作失效，造成较大的调整成本。

[0151] 由图9可以看出，使用部分重调度，当航天器1的第4个测试任务的测试时间增加4时，调度方案的总工期由59变为63。该调度方案保证了靠近重调度时刻的一段时间内的测试任务的开始时刻和执行顺序变化较小，如航天器1的第5个和第6个测试任务和航天器2的第3个和第2个测试任务等，同时对开始时刻不小于$T+\Delta T=24$的测试任务进行重新优化保证了调度方案具有较小的测试总工期。一般情况下，使用部分重调度得到的调度方案的总工期会介于完全重调度和右移策略之间，调整成本也较低。

[0152] 综上，本发明一种基于遗传算法的多航天器并行测试任务调度方法，通过考虑调度方案执行时遇到的不确定性因素的干扰，在基线调度方案的基础上使用遗传算法或右移策略不断地更新调度方案，能够始终保持调度方案的可行性和较短的总工期，从而克服了传统并行测试任务调度方法的局限性，达到了航天器并行测试资源优化配置的目的，提高了测试效率，降低了测试成本。

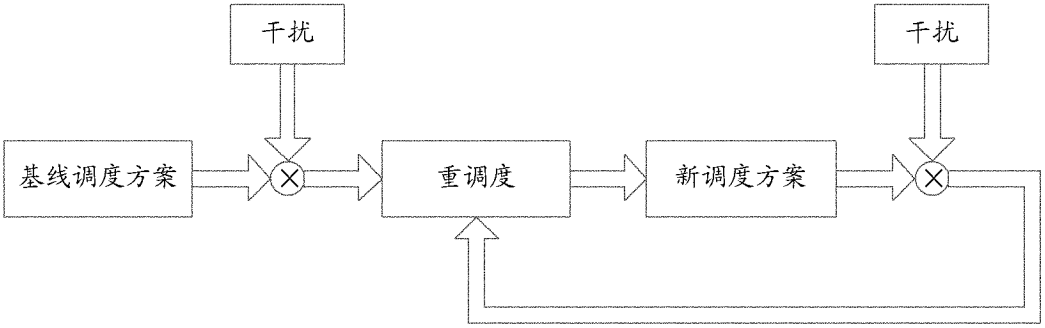

干扰
干扰
基线调度方案
重调度
新调度方案

图1

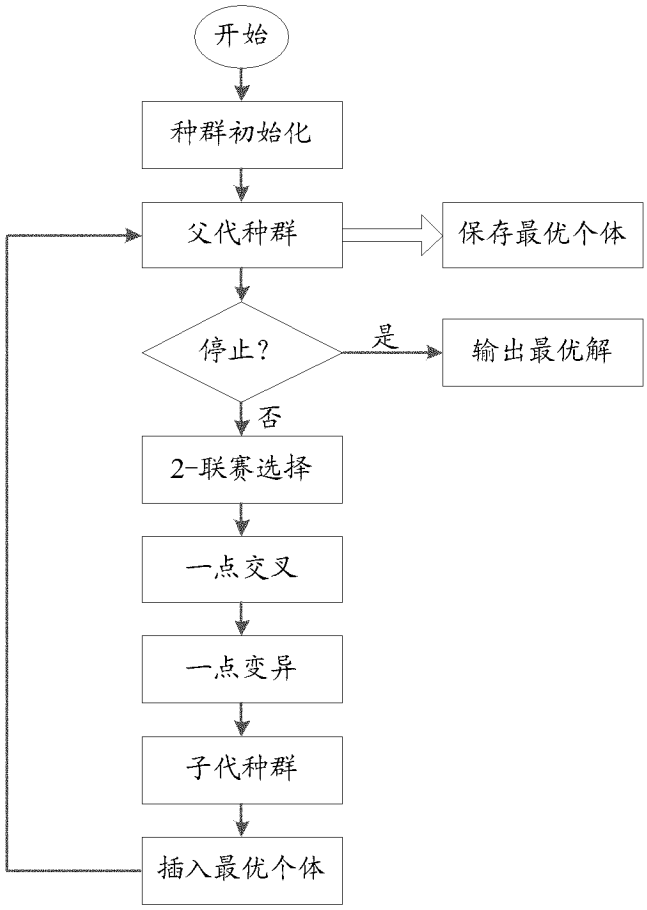

开始
种群初始化
父代种群
保存最优个体
停止？
是
输出最优解
否
2-联赛选择
一点交叉
一点变异
子代种群
插入最优个体

图2

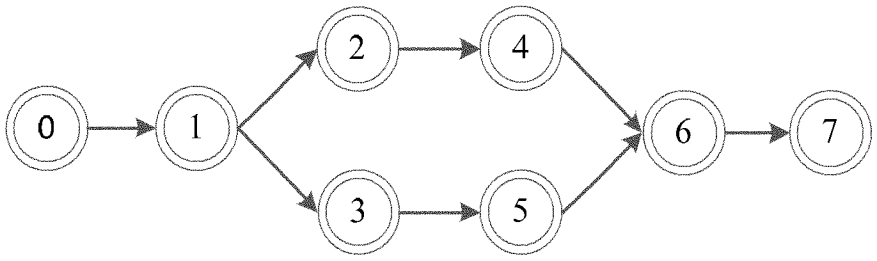

0
1
2
3
4
5
6
7

图3

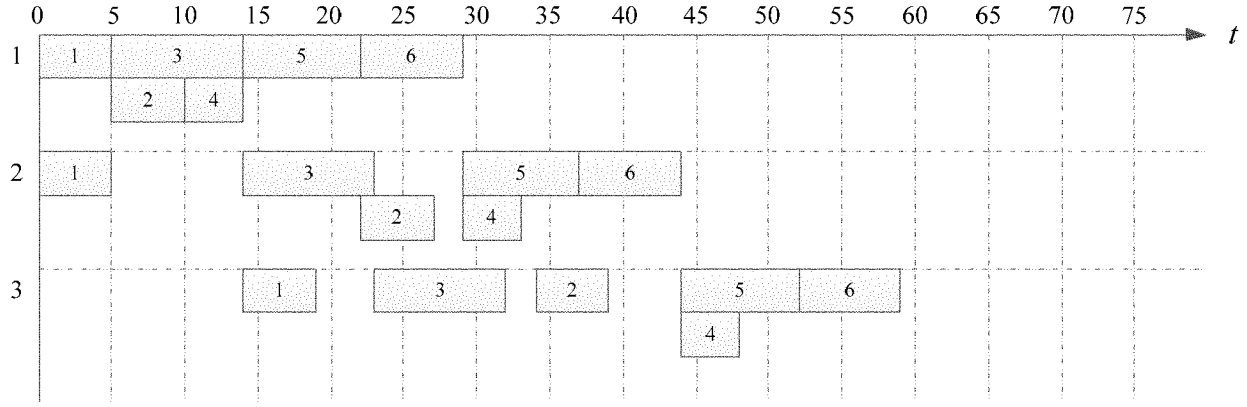

0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
t
1
2
3

图4

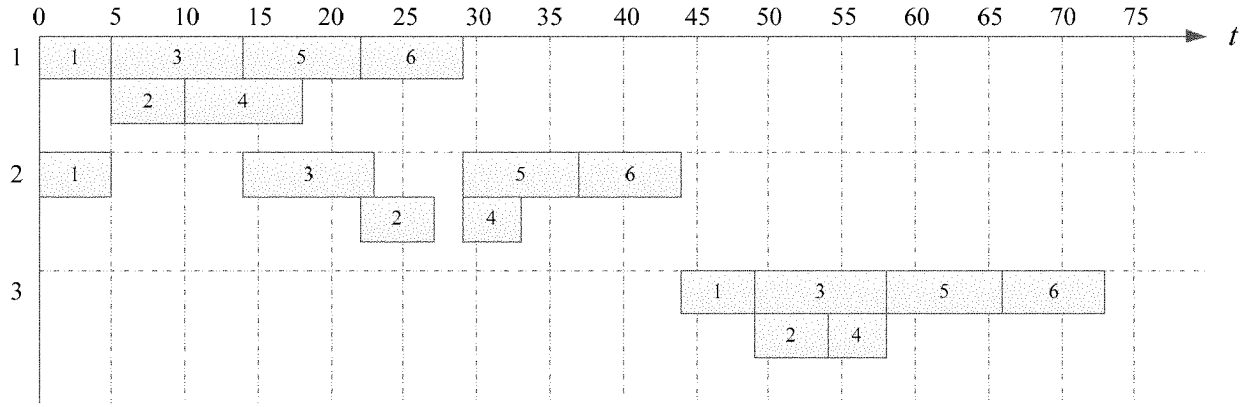

0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
t
1
2
3

图5

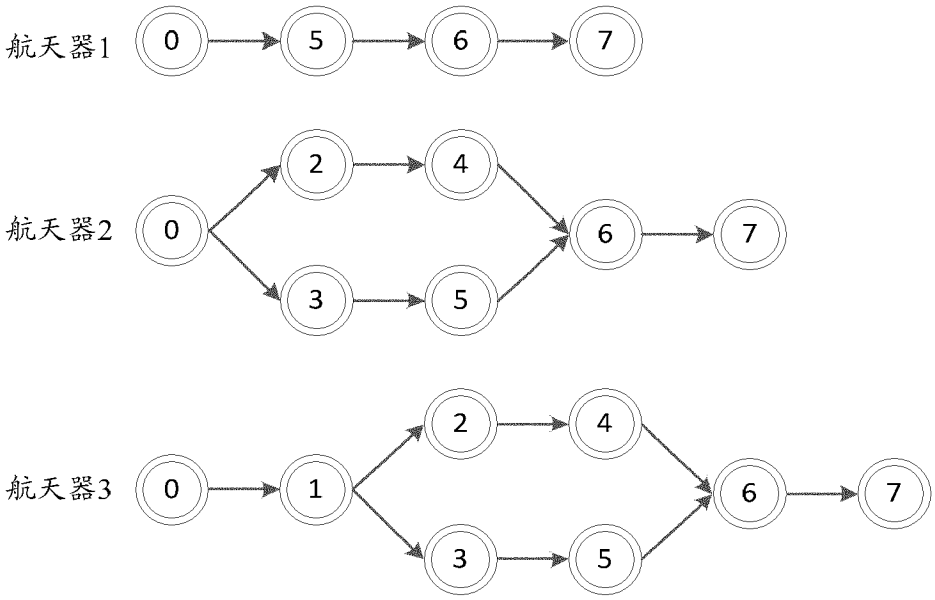

航天器1
0
5
6
7
航天器2
0
2
4
3
5
6
7
航天器3
0
1
2
4
3
5
6
7

图6

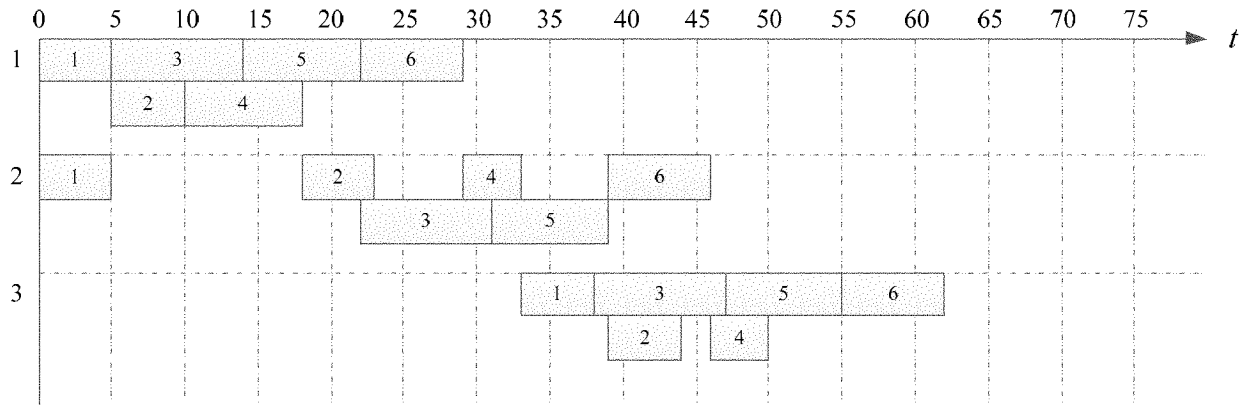

0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
t
1
2
3
1
3
5
6
2
4
1
2
4
6
3
5
1
3
5
6
2
4

图7

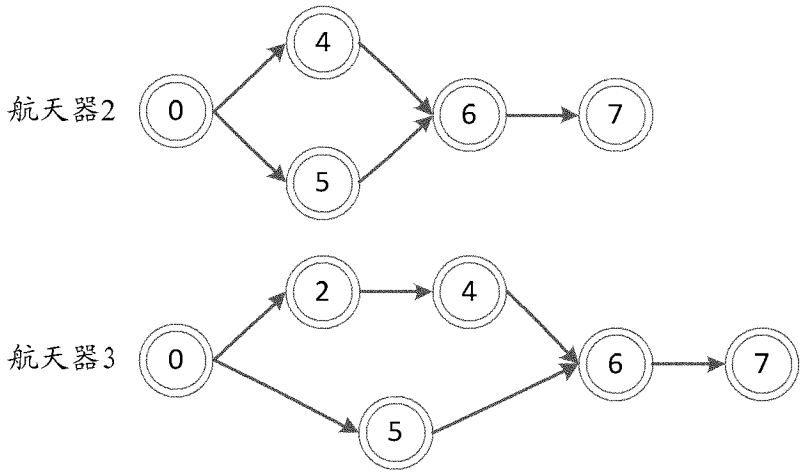

航天器2
0
4
5
6
7
航天器3
0
2
4
5
6
7

图8

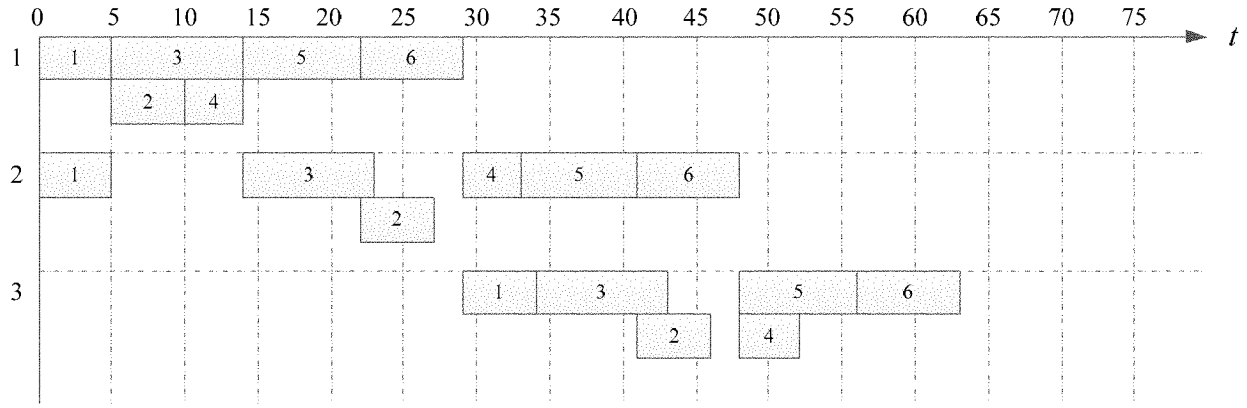

0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 t
1
2
3

图9