间歇式化工系统的Petri网模型预测控制实时调度方法

**摘要**：为了解决复杂间歇式化工系统的实时调度问题，提出一种基于Petri网的模型预测控制实时调度方法：首先，根据生产工序，建立生产过程的赋时Petri网模型；其次根据生产对象的拓扑结构，在该赋时Petri网模型中引入阀门系统的网结构，从而获得该系统的受控Petri网模型；再次，通过赋时Petri网拓扑结构设计启发式函数，构建赋时Petri网的启发式A\*算法；最后，利用模型预测控制的优势，并构建赋时Petri网的模型预测控制启发式A\*算法，实时获得系统的调度策略。本文以啤酒过滤系统为例验证该方法。

## 0.引言

间歇式化工系统的调度与控制[1-3]问题，在学界已经取得了许多的研究成果。文献[4]设计了间歇式化工生产设备的微机控制系统，该系统实现了从简单的报警监测和过程记录到在线闭环控制。文献[5]利用赋时Petri网对多产品批处理工厂进行建模，并且研究短期调度问题，设计算法解决优化调度问题。文献[6]针对间歇式化工系统的调度使用赋时Petri网对系统建模，并且使用A\*搜索算法找到最优策略。目前对间歇式化工系统的调度与控制的调度策略往往都是离线获取，通常都是在任务还未开始执行之前通过预先制定的算法、规则或优化模型，对任务进行规划和调度，其最大的不足是无法适应任务执行过程中的实时变化，随着任务量增加的时候，通过离线计算获取调度策略无法快速得到执行的操作从而导致执行的滞后。

模型预测控制（MPC）[7]可以很好的解决实时获得调度策略的问题，它是通过已有的数据或构建好的数学模型来进行预测，并且将这些预测值进行评估选择一个最好的值执行。许多学者使用模型预测控制在调度与控制等方面进行大量研究[8-10]。文献[11]利用模型预测，实现可再生能源电网与混合能源储存系统的最优控制。文献[12]开发一种负荷预测技术和管理算法，用于通过模型预测控制调度电池和暖通空调系统的建筑，对建筑物的负荷进行调整，并最小化需求费用。

Petri网是一种数学工具，用于建立表示动态行为的状态方程[13,14]，它们推断未知的状态信息，并以图形形式显示动态行为。间歇式化工系统由阀门开启或关闭等事件驱动，因此可以使用Petri网进行建模，当前的建模方法通常涉及模块化建模[15,16]。间歇式化工系统中存在大量的阀门，其生产过程是通过对阀门控制的管道将流体从一个储罐传输到另一个储罐中。它具有许多资源元素，并且涉及共享资源的分配。间歇式化工系统的Petri网建立需要对阀门的操作进行整理，分析各个操作之间的关系，由于涉及到共享资源的表示与分配，即便是小型管道运输系统也可能出现复杂的建模。

本文提出一种间歇式化工系统的Petri网模型预测控制实时调度方法。（1）对间歇式化工系统的生产流程与阀门的控制操作进行赋时Petri网建模；（2）为相应库所设计到目标库所最短距离，通过赋时Petri网的拓扑结构设计模型的启发式，以加快搜索过程。（3）设计启发式A\*搜索算法，结合赋时Petri网模型的特点，以高效地搜索最优调度策略，确保生产任务在最短时间内得到执行；（3）设计赋时Petri网模型预测算法，结合A\*搜索算法，实现实时调度，在预测的过程中，对每一步策略计算所消耗的时间是毫秒级别，以保证实时性；（4）随着对预测的步长逐渐增加，该方法地调度策略越接近Dijkstra搜索的调度策略，从而提高预测准确性。该方法通过啤酒过滤装配系统进行说明，为间歇式化工系统的实时调度提供了一种高效可行的解决方案。

本文的其余部分由以下几部分组成。第二节描述间歇式化工系统中的并行控制问题。第三节提出利用Petri网对间歇式化工系统进行建模。第四节设计管间歇式化工系统的A\*搜索算法。第五节设计间歇式化工系统的模型预测算法并且将A\*搜索算法进行结合。第六节实验验证。第七节对这项工作的总结。

## 1.问题描述



图1 啤酒过滤系统示意图

如图1所示，这是某啤酒厂过滤装配系统，该系统分别执行填充、过滤、灌装和清洗四个任务。其中Ts、T1、T2分别是啤酒原液罐和两个缓冲罐，是三个过滤器，CIP是清洁剂的供应和回收系统，Bottling System是灌装系统，是双盘式活塞阀。需要注意的是，每个阀门都可以切换到“开”或“关”位置。打开阀门时，来自其控制下的垂直和水平管道的流体会混合，然后通过所有出口管道流出；阀门关闭时，垂直和水平管道中的流体分别流动。该系统共有12个操作，如表1所示。其中填充操作是将啤酒原液通过管道从Ts运送到缓冲罐T1。有两种传输途径，对应的阀门操作为：打开和或打开阀门和。将这两种路径的填充操作分别用和表示。当使用操作时，流体运输5s后停止填充；当使用操作时，流体运输6s后停止填充。过滤操作是将缓冲罐中的液体，经过过滤器传输到缓冲罐。有三种传输路径，对应的阀门操作为：打开阀门和或打开阀门和或打开阀门和。将这三种路径的填充操作分别用表示。当使用操作时，流体运输11s后停止过滤；当使用操作时，流体运输12s后停止过滤；当使用操作时，流体运输13s后停止过滤。灌装操作是将缓冲罐T2中的液体传输到灌装系统进行啤酒液装瓶。有两种传输路径，对应的阀门操作为：打开阀门和或打开阀门。将这两种路径的填充操作分别用表示。当使用操作时，流体运输7s后停止灌装操作；当使用操作时，流体运输8s后停止灌装操作。此外，还有五种不同的清洗任务，分别如下：

（1）（清洗缓冲罐）：打开阀门和，清洗剂从流入缓冲罐；

（2）（清洗缓冲罐）：打开阀门，清洗剂从流入缓冲罐；

（3）（清洗过滤器）：打开阀门，清洗剂从流入过滤器；

（4）（清洗过滤器）：打开阀门和，清洗剂从流入过滤器；

（5）（清洗过滤器）：打开阀门和，清洗剂从流入过滤器；

需要注意的是缓冲罐使用两次就必须清洗一次，缓冲罐使用三次就必须清洗一次；过滤器使用一次就必须清洗一次；过滤器使用两次就必须清洗一次；使用三次就必须清洗一次；此外由于啤酒和清洁剂不允许在该系统中混合，应始终保持两种材料之间的屏障完好无损。

因此，啤酒厂过滤装配系统需要满足以下控制要求：

（1）当有生产任务时，各个阀门的操作必须按照调度策略中的指令执行，以确保生产任务的顺利进行。

（2）必须合理分配设备的任务，以平衡负载，避免某些设备过载而导致生产效率下降。

（3）鉴于给定生产任务时，对各个阀门的操作和设备任务分配可能会涉及资源的共享和竞争，需要研究如何合理地分配设备，以实现负载平衡。

（4）实现对阀门操作的实时调度，确保操作在适当的时机进行，避免生产任务的延误或冲突。

为了达到上述控制要求，接下来的工作将包括以下内容：

（1）建立系统的Petri网模型，对啤酒厂过滤装配系统进行建模，以便进行后续的分析和控制策略设计。

（2）设计实时调度策略的相关算法，考虑设备的负载情况和生产任务的优先级，以确保任务的高效执行。

（3）使用啤酒厂过滤装配系统作为示例，验证所提出的建模和调度策略的有效性和可行性。

表1 啤酒过滤系统的操作表单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operations** | **Open valves** | **Closed valves** | **Tanks** | **Time** |
| *o*1 | 2,3 | 8,9,12,1 | *T*1 | 4s |
| *o*2 | 12,3 | 8,9,3,11 | *T*1 | 4s |
| *o*3 | 3,4 | 8,9,7,10,1,2 | *T*1,*T*2,*M*1 | 4s |
| *o*4 | 13,14 | 8,9,7,10,11,12,3,4 | *T*1,*T*2,*M*2 | 4s |
| *o*5 | 18,19 | 8,9,7,10,3,4,13,14,17 | *T*1,*T*2,*M*3 | 4s |
| *o*6 | 4,5 | 7,10,1,6 | *T*2 | 4s |
| *o*7 | 14,15 | 7,10,11,16,5,4 | *T*2 | 4s |
| *o*8 | 8,9 | 3,13,10,7 | *T*1 | 2*min* |
| *o*9 | 7,10 | 4,14 | *T*2 | 2*min* |
| *o*10 | 1,6 | 2,3,4,5,7,8,11,17 | *M*1 | 2*min* |
| *o*11 | 11,16 | 12,13,14,15,7,8,17 | *M*2 | 2*min* |
| *o*12 | 17,20 | 18,19,7,8 | *M*3 | 2*min* |

## 2.间歇式化工系统的Petri网建模

本节将介绍使用Petri网概念[15,16]来建立系统的加工过程模型和预防资源冲突单元模型。我们定义为严格正整数集。

在间歇式化工系统的生产工序中，我们用D = V ∪ R 来表示资源的集合。其中，V是阀门资源的集合，即V={，…，，，…，}，其中***i***∈N+，，…，是阀门为打开的状态，，…，是阀门为关闭的状态；R则包括除阀门资源外的其他资源，例如缓冲罐、过滤器和管道等，即R={…,,…,,,…,，,…,}，其中，…是表示缓冲罐为空的状态，,…,是表示缓冲罐为满的状态，,…,是表示过滤器使用中，,…,是表示过滤器使用完成。因此该系统所有的资源D = {，…，，，…，，…,,…,,,…,，,…,}。对于一个操作，完成该操作所需的资源集合记为D（）⊆ D。

**定义1：**当系统有生产任务时，阀门操作集合用标识，其中，。

在执行生产任务时，系统需要使用一些资源，例如：阀门、缓冲罐。我们用表示操作执行次数，其中∈N+。

**定义2：**对于给定的任务中，如果一个操作的执行需要在另一个操作执行完成之后才能进行，我们可以称这个前一个操作（）为后一个操作（）的前提操作。

例如，在表1中，如果或的操作需要缓冲罐有液体，那么或的操作就成为了或的前提操作。换句话说，为了执行或，首先要确保或已经完成，以使得T1中有液体可以通过过滤器并填充到中。

**定义3：**若前一个操作（）作为后一个操作（）的前提操作，为了确保前一个操作成功执行，我们添加作为它们的缓冲操作。这样的缓冲操作确保了在执行之前，先执行，并在成功执行后，再执行，从而满足操作之间的前后顺序关系。

例如，的操作分别是的前提操作。我们在与之间添加缓冲操作；在与之间添加缓冲操作；在与之间添加缓冲操作；在与之间添加缓冲操作；通过上面的描述我们只需在操作和操作与操作和操作之间添加一个共享的缓冲罐操作。这样一来，操作就能够同时为和的后续操作提供前提，并且也能同时为和的后续操作提供前提。这种方式可以避免在每一对操作之间都添加单独的缓冲操作，并且可以保证或能够成功执行。

**定义4**：如果有一个缓冲操作在任务初始时刻需要被执行,则称其为初始缓冲操作。

例如，操作是操作的前提操作，我们在操作和操作之间添加缓冲操作。其中的实际物理含义打开阀门1,6将清洗剂从流入过滤器进行清洗,的实际物理含义是打开阀门3,4将缓冲罐中的流体运输到缓冲罐。任务开始执行之前，所有资源都是处于初始的状态，不需要执行操作完成后才能执行操作，即过滤器不需要清洗，那么我们称缓冲操作称为初始缓冲操作。

**定义5**：如果存在，例如∈D(o)&∈D(o’)；或者存在∈D(o)∩D(o’)（其中*i*∈N+），例如∈D(o)&∈D(o’)；或者存在∈D(o)∩D(o’) （其中*i*∈N+），例如∈D(o)&∈D(o’)。那么我们就称o与o’是存在冲突的，表示为oo’。反之o与o’是不冲突的，表示为oo’。我们将具有冲突的操作整理成集合称为最大冲突集合，表示为Omax。如果在一个冲突集合中的两两操作都是冲突，即oo’，并且存在o∈Omax，o’Omax满足oo’，则称该冲突集合是最大冲突集合，即Omax。

根据定义1-5结合Petri网的概念给出图1赋时Petri网如图2所示。

**Step1:**分别创建开始与结束库所，，，,其中 ，表示任务量；

**Step2:**根据表1的操作o*i*，创建相应的库所与变迁，与，、分别表示动作开始与结束，其中*i*=1，2，3…12。并且创建有向弧（，）和（，）。

**Step3:**根据定义2-4和表1的操作o， o1，o2与o 3，o 4，o 5之间有缓冲操作，记为o b1，将其创建缓冲库所用表示；o 3与o 10之间有缓冲操作，记为o b2，将其创建缓冲库所用表示；o 10与o 3之间有缓冲操作，记为o b3，并且该缓冲操作在任务的初始时候需要执行，因此该缓冲操作为初始缓冲库所，将其创建初始缓冲库所用表示，其m（）=1；o 4与o 11之间有缓冲操作，记为o b4，将其创建缓冲库所用表示；o 11与o 4之间有缓冲操作，记为o b5，并且该缓冲操作在任务的初始时候需要执行，因此该缓冲操作为初始缓冲库所，将其创建初始缓冲库所用表示，其m（）=2；o 5与o 12之间有缓冲操作，记为o b6，将其创建缓冲库所用表示；o 12与o 5之间有缓冲操作，记为o b7，并且该缓冲操作在任务的初始时候需要执行，因此该缓冲操作为初始缓冲库所，将其创建初始缓冲库所用表示，其m（）=3；o 3，o 4，o 5与o 8之间有缓冲操作，记为o b8，将其创建缓冲库所用表示；o 8与o 1，o 2之间有缓冲操作，记为o b9，并且该缓冲操作在任务的初始时候需要执行，因此该缓冲操作为初始缓冲库所，将其创建初始缓冲库所用表示，其m（）=2；o 3，o 4，o 5与o 6，o 7之间有缓冲操作，记为o b10，将其创建缓冲库所用表示；o 6，o 7与o 9之间有缓冲操作，记为o b11,将其创建缓冲库所用表示；o 9与o 3，o 4，o 5之间存在缓冲操作，记为o b12，并且该缓冲操作在任务的初始时候需要执行，因此该缓冲操作为初始缓冲库所，将其创建初始缓冲库所用,m（）=3；因此缓冲库所分别是：,,,,,,,,,,,。分别创建前提动作的结束变迁到缓冲库所的有向弧，以及缓冲库所到下一个执行操作的开始变迁。

**Step4:**根据定义5可以列出最大冲突操作集：{ o1, o10, o11},{ o3, o4, o5, o8, o9},{ o6, o10},{ o2, o11, o12},{ o8, o10, o11, o12},{ o6, o7, o9},{ o7, o11},{ o1, o2, o8},{ o3, o4, o5, o9}。通过最大冲突操作集分别创建监控库所,,…,，其中m0()= m0()=…= m0()=1。并且设计了监控地点(i=1,2,…,8)和相应的弧[17-22]。

## 3.间歇式化工系统的启发式A\*搜索算法

3.1 间歇式化工的启发式函数设计

对于Petri网启发式函数路径的搜索已经取得许多研究 [23-24]。本文给出间歇式化工系统Petri网的启发式函数:

其中是Petri网中各个库所到目标库所(除控制库所外)最短距离，是指取出非空库所的最短距离。通过取出当前状态所有非空库所的最短距离进行比较，最大的最短距离就是当前状态的启发式函数值。

**定理1** 如果一个启发式函数是可采纳的，那么它的A\*算法是最优的[25]

**定理2** 给定的间歇式化工系统Petri网的启发式函数h是可采纳的.

**证明：**由于是取有托肯的库所到达目标库所的最短距离值的最大值，系统中速度是一定的，即所花费的时间。是从当前状态到目标状态实际所耗费的时间，由于系统中是存在共享资源情况，因此所耗费的时间会远大于预计耗费的时间，因此是可容许的.证毕.

**3.2** 基于赋时Petri网的运行规则，给出间歇式化工的赋时Petri网的A\*算法架构。

**算法1** 间歇式化工系统的启发式A\*搜索算法

输入：Petri网初始标识,启发式函数,目标标识

输出：最优调度策略的变迁激发序列.

1 创建根节点,其中为初始标识,是一组向量,即,,,,discard为初始值为false的布尔变量; OPEN表存储待扩展结点,OPEN = ,其中OPEN表的数据结构是优先队列；CLOSE表存储已扩展结点,CLOSE = ,其中CLOSE表的数据结构是哈希表；将存入OPEN表中.

2 如果OPEN表为空,则算法退出.

3 从OPEN表弹出第一个结点



**图1 啤酒过滤装配系统赋时Petri网**

4 判断第一个结点中的discard是否等于true，如果等于true删除结点返回第2步，否则程序继续执行.

5 如果的标识是目标标识,则输出到的变迁激发序列,算法退出.

6 计算的标识的使能的变迁集合.

7 ,计算使能变迁t激发的子节点的信息：;绘制一条由指向的有向弧，并在弧上附上;将存入COLSE表中；

8 对进行新旧结点判断:

8.1 如果 的标识与CLOSE表中的标识相同；

8.1.1 将中的和中的放入公式：,进行判断如果的标识的非空库所的都不满足或不满足这个公式，则是新节点存入OPEN表中，并且给中的discard标记为false,给中的discard标记为true.

8.1.2 将看成新扩展结点，将看成已扩展结点将它们的相关信息放入公式(1)，如果的标识的非空库所的已等待时间都满足这个公式，则是旧结点，并且给中的标记为true.

8.2 如果 的标识CLOSE表不存在，则将存入OPEN表，并且给中的discard标记为false.

9 OPEN表中按f值进行升序排列.

10 返回2.

算法1给出了间歇式化工系统的赋时Petri网A\*算法的架构。根据赋时Petri网运行规则，每次扩展OPEN表中f值最小的状态结点，指到搜索到目标，从而得到从初始结点到目标结点的变迁序列，即间歇式化工系统的最优调度策略。

步骤6-7计算所有子节点的，并且利用3.1得出启发式函数值和评估函数;

步骤8-9是判断子节点的新旧问题，我们提出一种新的判断结点新旧的计算公式：

,其中。 (1)

假设是新扩展结点，是CLOSE表中已扩展结点，它们的标识是相同的，它们从初始状态到当前状态所消耗时间分别是与，通常我们认为如果，就判断所对应的是旧节点，然而这对新旧节点判断是非常严格的，也可能会出现是新节点的情况，因此我们对公式(1)进行证明：

**证明：**假设，我们将中的的维度拉到与中的的维度一样，即,那我们分别定义两个函数与*k*,其中*i*=1,…,35，是取当前状态非空库所下标; 此时我们判断新扩展结点与已扩展结点在标识是相同的情况下，新旧的依据就是看它们的已等待时间。假设已扩展结点与新扩展结点的所有非空库所都满足*k*，就判断新扩展结点就是旧节点，反之就是新节点。证毕。

步骤9 是OPEN表中按f值进行升序排列；

为了检验间歇式化工系统启发式A\*搜索算法的性能，我们根据算法1编写C++程序，其性能结果通过以下规格获得：CPU i7-11700k 3.60GHz，RAM 32.0GB.以图1所示的啤酒过滤装配系统为例，通过改变任务量设计了6组实验，对比Dijkstra算法和间歇式化工系统赋时Petri网A\*算法.实验安排和结果如表2所示.

**表2 Dijkstra算法和****间歇式化工系统启发式A\*搜索算法对比实验**

|  |  |  |
| --- | --- | --- |
| 实验 | Dijkstra算法 | 间歇式化工系统赋时Petri网A\*算法 |
| 序号 | 加工时间(min) | 加工时间(min) |
| 2 | 0.7 | 0.7 |
| 5 | 7.61 | 7.61 |
| 10 | 19.95 | 19.95 |
| 15 | 31.6 | 31.6 |
| 20 | 44.3 | 44.3 |
| 30 | 69.6 | 69.6 |

通过表2实验结果可知间歇式化工系统赋时Petri网A\*算法能够得到最优策略。

## 4.间歇式化工系统的模型预测算法

本节将仿照模型预测控制[7]的思路使用间歇式化工系统的赋时Petri网模型进行对未来的预演和优化。

（1）预测模型

这里使用的预测模型就是库所赋时Petri网。我们都知道赋时Petri网是可以演化离散事件动态系统模型，保持赋时Petri网和实际系统始终一致，将其展开成可达图就可以对未来状态进行预演。

（2）滚动优化

在动作执行过程中进行优化。优化的目的是从动作完成后的状态出发根据设置预测步数对状态空间（可达图）进行搜索，寻找一个状态。当状态空间扩展到达所设置预测步数后，当前步数所有状态中最大状态就是。其中是A\*算法中的启发式函数，。这种优化过程等效于最小的状态，与A\*算法的搜索方式类似，但不一定要搜索到目标结点。在找到后，只执行到的第一段局部动作序列，并在执行期间继续滚动优化。其中每次搜索并不是重新创建新的搜索树，整个流程建立在同一个搜索树上，每当最新状态更新时自动剪除无意义分支。

（3）反馈矫正

每次A\*搜索到所设置的预测步数后，算法会生成一段局部动作序列，并且根据赋时Petri网模型进行预测，得到预测执行后的状态。随后，当系统实际执行这段局部动作序列后，会对比预测得到的状态与真实的系统状态，如果不一致，将实际状态更新到赋时Petri网模型，重新开始搜索。

在模型预测过程中，利用滚动优化策略，系统能够根据预测步数有选择地搜索状态空间，选择在当前步数中具有最大优先级的状态进行执行。这样智能的优化策略提高了任务执行的效率，并在保持搜索树的实时性的同时降低了计算成本，使得动作执行过程更加高效和智能化。另外，利用反馈矫正不断优化动作序列的选择，可以适应系统动态变化和不确定性，从而实现更高效、更智能的任务执行。

**算法2** 基于间歇式化工系统的赋时Petri网模型预测控制实时调度算法

**输入：**Petri网初始标识,启发式函数,预测步长,目标标识.

**输出：**实时调度指令.

1创建根节点,其中为初始标识;是一组向量,即;;;;discard为初始值为false的布尔变量; ;

2 初始化f值小先出的优先队列OPEN = {}；

3 初始化哈希表CLOSE ={}；

4 将存入OPEN表中.

5 Step = x 设置模型预测步长.

6 Step\_num = 0 设置模型预测后续步长

7 初始化容器Open\_temp;

8

9

10 将OPEN表中的所有元素弹给Open\_temp

11

12 Open\_temp\_size=Open\_temp.size();

13 初始化f值小先出的优先队列best\_Son\_node={}

14 ;

15

16 初始化f值小先出的优先队列 best\_f\_list;

17 从Open\_temp表弹出第一个结点

18

19

20

21

22

23

24 计算的使能的变迁集合.

25

26计算使能变迁t激发的的信息： 绘制一条指向的有向弧，并在弧上附上,将best\_f\_list;;

27

28 COLSE[hash()] ← ；

29

30 对进行新旧结点判断:

30.1 如果 的标识与CLOSE表中的标识相同；

30.1.1 将中的和中的放入公式(1)进行判断如果的标识的非空库所的都不满足或不满足这个公式，给中的discard标记为false,将是新节点存入Open\_temp中;

30.2.2 将看成新扩展结点，将看成已扩展结点将它们的相关信息放入公式(1)，如果的标识的非空库所的已等待时间都满足这个公式，则是旧结点，并且给中的标记为true.

30.2 如果 的标识CLOSE表不存在，则将存入 Open\_temp中，并且给中的discard标记为false.

31

32 Open\_temp\_size--；

33

34 ;

35 Step--;

36

37

38 将存入best\_Son\_node;

39

40

41

42

43

44

45 删除赋值给的结点

46

47

48 对进行降序排序;

49

50 给执行器发送执行从root到动作序列指令，并且将root为根节点的树除子树外的其他结点discard置true

51

break;

52

53 ;

54

55 寻找的子节点为空为空的子结点,将这些结点放入OPEN表中；

56 ；

57

该算法总体结构是一个while循环，循环的终止条件是OPEN表为空或搜索到目标节点。循环内包括两个主要部分，分别是扩展搜索树和动作选择。

在扩展部分(步骤15-43)，使用有限预测步长为的A\*算法进行扩展状态。除了最开始设置的步长，后续算法运行过程中每次只扩展一步，这能够避免计算的重复性。如果只将根结点中最优的待扩展结点存入OPEN中，会造成计算的重复性，从而无法提高计算速度。

在动作选择部分（步骤47-55），首先从弹出值最小的子结点，即，然后找到在根结点中待扩展的父节点，并将除了已选择的待扩展结点的子树外，由根结点产生的其他子树中的结点置为discard = true，接着执行待扩展结点的子树的调度指令；其次，通过已选择的待扩展结点寻找子结点为空的结点；最后，将子结点为空的结点全部存入OPEN表中，并且让，以确保下一次预测只扩展一步。

该算法结合模型预测控制与A\*搜索算法的优势，在搜索的过程中按照A\*算法的规则进行扩展状态，在调度过程中利用模型预测的规则实时给出调度指令，以确保调度的实时性。通过这种结合，能够在实时搜索过程中灵活地根据实际情况调整搜索树的扩展，并通过模型预测进行实时调度，从而实现更高效、更智能的任务执行。

参考文献

[1] A. Mastoras and T. R. Gross, "Unifying Fixed Code Mapping, Communication, Synchronization and Scheduling Algorithms for Efficient and Scalable Loop Pipelining," in IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 9, pp. 2136-2149, 1 Sept. 2018, doi: 10.1109/TPDS.2018.2817207.

[2] F. Bendali, A. Quilliot, E. M. Kamga, J. Mailfert and H. Toussaint, "Pipe-lining Dynamic Programming Processes in Order to Synchronize Energy Production and Consumption," 2020 15th Conference on Computer Science and Information Systems (FedCSIS), Sofia, Bulgaria, 2020, pp. 303-306, doi: 10.15439/2020F13.

[3] Aristeidis Mastoras and Thomas R. Gross. 2019. Chunking for Dynamic Linear Pipelines. ACM Trans. Archit. Code Optim. 16, 4, Article 44 (December 2019), 25 pages. https://doi.org/10.1145/3363815

[4]R. Young and D. E. Svoboda, "A Minicomputer Control System for a Batch Chemical Production Facility," in IEEE Transactions on Industrial Electronics and Control Instrumentation, vol. IECI-17, no. 4, pp. 282-284, June 1970, doi: 10.1109/TIECI.1970.229605.

[5] Tianlong Gu and P. A. Bahri, "Timed Petri-net representation for short term scheduling of multiproduct batch plants," Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251), San Diego, CA, USA, 1999, pp. 4092-4096 vol.6, doi: 10.1109/ACC.1999.786313.

[6] W. Lin, J. L. Luo, J. Z. Zhou, Y. S. Huang and M. C. Zhou, "Scheduling and control of batch chemical processes with timed Petri nets," 2016 IEEE International Conference on Automation Science and Engineering (CASE), Fort Worth, TX, USA, 2016, pp. 421-426, doi: 10.1109/COASE.2016.7743436.

[7] E. F. Camacho and C. Bordons, Model Predictive Control. New York: Springer-Verlag, 1999.

[8] X. Du, X. -M. Sun, Z. -M. Wang and A. -N. Dai, "A Scheduling Scheme of Linear Model Predictive Controllers for Turbofan Engines," in IEEE Access, vol. 5, pp. 24533-24541, 2017, doi: 10.1109/ACCESS.2017.2764076.

[9]P. Li, Y. -B. Zhao and Y. Kang, "Integrated Channel-Aware Scheduling and Packet-Based Predictive Control for Wireless Cloud Control Systems," in IEEE Transactions on Cybernetics, vol. 52, no. 5, pp. 2735-2749, May 2022, doi: 10.1109/TCYB.2020.3019179.

[10] J. Bertoncini, V. Nikitina and M. Gerdts, "Multi-Agent Dynamic Scheduling With a Posteriori Path Tracking and Collision Avoidance Using Model Predictive Control," in IEEE Control Systems Letters, vol. 7, pp. 2101-2106, 2023, doi: 10.1109/LCSYS.2023.3284761.

[11] F. Garcia-Torres and C. Bordons, "Optimal Economical Schedule of Hydrogen-Based Microgrids With Hybrid Storage Using Model Predictive Control," in IEEE Transactions on Industrial Electronics, vol. 62, no. 8, pp. 5195-5207, Aug. 2015, doi: 10.1109/TIE.2015.2412524.

[12] D. T. Vedullapalli, R. Hadidi and B. Schroeder, "Optimal Demand Response in a building by Battery and HVAC scheduling using Model Predictive Control," 2019 IEEE/IAS 55th Industrial and Commercial Power Systems Technical Conference (I&CPS), Calgary, AB, Canada, 2019, pp. 1-6, doi: 10.1109/ICPS.2019.8733344.

[13] Jiliang Luo, Yi-Sheng Huang, and Yi-Shun Weng, “Design of variable traffic light control systems for preventing two-way grid network traffic jams using timed petri nets,” IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 7, pp. 3117–3127, 2020.[14] Shuo Wang and Jiliang Luo, “Synthesis of mobile robot control system on embedded chips via petri nets,” Journal of the Chinese Institute of Engineers, vol. 41, no. 6, pp. 442–451, 2018.

[15 JiLiang Luo, KaiCheng Tan, HuaiJu Luo, and MengChu Zhou, “Inference approach based on petri nets,” Information Sciences, vol. 547, pp. 1008–1024, 2021.

[16] Jiliang Luo, Yaxin Wan, Weimin Wu, and Zhiwu Li, “Optimal petri-net controller for avoiding collisions in a class of automated guided vehicle systems,” IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 11, pp. 4526–4537, 2020.

[17] Giua, A. et al. “Generalized mutual exclusion contraints on nets with uncontrollable transitions.” [Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics (1992): 974-979 vol.2.

[18] Yamalidou, Katerina et al. “Feedback control of Petri nets based on place invariants.” Proceedings of 1994 33rd IEEE Conference on Decision and Control 3 (1994): 3104-3109 vol.3.

[19] J. Luo, H. Ni, W. Wu, S. Wang and M. Zhou, "Simultaneous Reduction of Petri Nets and Linear Constraints for Efficient Supervisor Synthesis," in IEEE Transactions on Automatic Control, vol. 60, no. 1, pp. 88-103, Jan. 2015, doi: 10.1109/TAC.2014.2336431.

[20] J. Luo and M. Zhou, "Petri-Net Controller Synthesis for Partially Controllable and Observable Discrete Event Systems," in IEEE Transactions on Automatic Control, vol. 62, no. 3, pp. 1301-1313, March 2017, doi: 10.1109/TAC.2016.2586604.

[21] J. Luo, Q. Zhang, X. Chen and M. Zhou, "Modeling and Race Detection of Ladder Diagrams via Ordinary Petri Nets," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 48, no. 7, pp. 1166-1176, July 2018, doi: 10.1109/TSMC.2016.2647219.

[22] J. Luo, W. Wu, M. Zhou, H. Shao, K. Nonami and H. Su, "Structural Controller for Logical Expression of Linear Constraints on Petri Nets," in IEEE Transactions on Automatic Control, vol. 65, no. 1, pp. 397-403, Jan. 2020, doi: 10.1109/TAC.2019.2918052.

[23] Mejía, Gonzalo and Karen Niño. “A new Hybrid Filtered Beam Search algorithm for deadlock-free scheduling of flexible manufacturing systems using Petri Nets.” Comput. Ind. Eng. 108 (2017): 165-176.

[24] Lefebvre, Dimitri and Gonzalo Mejía. “Robust scheduling in uncertain environment with Petri nets and beam search.” IFAC-PapersOnLine 51 (2018): 1077-1082.

[25] Judea Pearl. 1984. Heuristics: intelligent search strategies for computer problem solving. Addison-Wesley Longman Publishing Co., Inc., USA.