运筹学 2017 年春季学期课程研究
——NP 问题：集束型设备群调度优化

白一晟 2014011543

郑洁 2014011553

晏筱雯 2014011459

目录

1 大作业介绍 5
1.1 摘要 5
1.2 关键词 5
1.3 概述 5
2 问题提出 6
2.1 设备描述 7
2.2 工艺介绍 7
2.3 限制条件 8
2.4 研究的问题 9
3 解决方案 9
3.1 串行 9
3.1.1 模型建立 9
3.1.2 方案概述 11
3.1.3 Childnodes 12
3.1.4 Readytime 13
3.1.5 Dominate 15
3.1.6 Makespan 15
3.1.7 Route 15
3.2 并行 16
3.2.1 并行模型 16
3.2.2 Childnodes 16
3.2.3 Readytime 17
3.3 可重入 19
3.3.1 可重入模型 19
3.3.2 Childnodes 19
3.3.3 Readytime 20
4 结果分析 21
4.1 串行 21

4.1.1 动作序列 21
4.1.2 对应的动作时间 21
4.1.3 工序时间 21
4.1.4 CPU 时间 22
4.2 并行 22
4.2.1 动作序列 22
4.2.2 对应的动作时间 22
4.2.3 工序时间 23
4.2.4 CPU 时间 23
4.3 可重入 23
4.3.1 动作序列 23
4.3.2 对应的动作时间 24
4.3.3 工序时间 24
4.3.4 CPU 时间 24
5 作业总结 24
5.1 遇到问题和解决方案 24
5.1.1 模型建立 24
5.1.2 机械双臂的分析 27
5.1.3 重入的模型分析 29
5.1.4 并行工艺 Readytime 的处理 29
5.1.5 Pareto 最优和支配关系的确立 31
5.2 优化尝试 31
5.2.1 层间删除 31
5.2.2 判断旋转 31
5.2.3 生成子代 32
5.3 项目收获感想 32
5.3.1 运筹学习 32
5.3.2 编程算法 33
5.2.3 文献报告 33

6 附录 33
6.1 成员分工 33
6.2 参考文献 34

1 大作业介绍

1.1 摘要

集束型制造设备是目前被广泛应用于半导体生成的自动化设备，主要包含了Load Lock(LL)、加工腔室（Processing Module）、机械手（Robot）、缓存腔室（Buffer Module）几个部分。晶圆（wafer）在集束型设备中经过一系列工艺加工，最终变成芯片。实际生产中，我们需要对机械臂的调度路径进行规划以提高效率，获得最优加工工序。目前，晶圆尺寸越来越大，其上所放置的芯片也越来越多样化，晶圆制造模式从而经常发生改变，所以不能采用周期性调度算法进行线路规划。本文中我们将问题转化为最小化各个设备的完成时间的多物体问题。利用 Pareto 最优动态规划法对路径进行删减，以求得最优路径。分析求解了串行（serial）、并行（parallel）和可重入（reentrant）工艺，并且对有着细微区别的不同算法下的 CPU 处理时间进行了讨论。

1.2 关键词

集束型制造设备、动态规划、pareto 最优

1.3 概述

集束型制造设备在半导体制造中有着非常广泛的应用，机械手可以拥有一个机械臂或者两个机械臂，晶圆需要经过一系列的加工腔室和缓存腔室来完成其工艺，而机械臂主要完成的工作时卸载、搬运和放置晶圆。

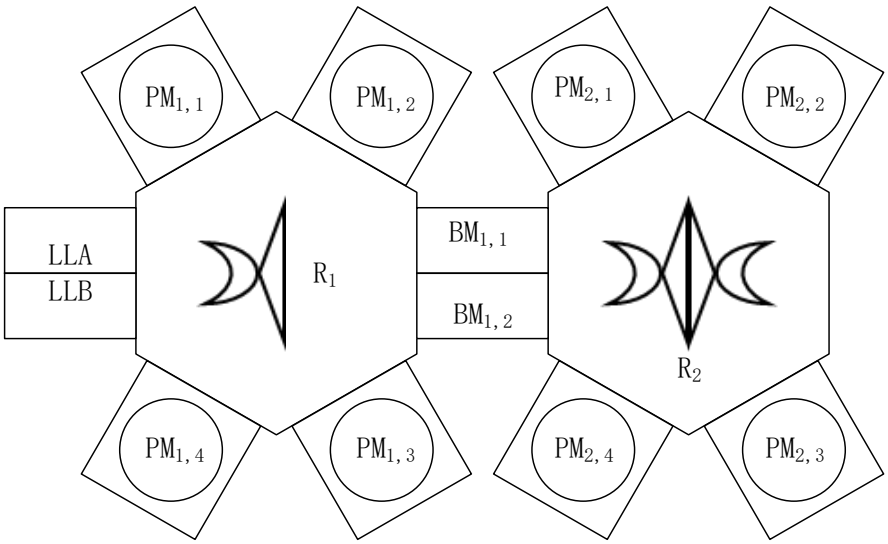
尽管集束型制造平台结构简单，但是它的调度优化问题却比较复杂。首先，集束型工具可能会有多种晶圆工作模式，比如串行、并行和可重入模式等。其次，一部分模式对于晶圆在不同加工腔室中的停留时间都有一定约束，有上下时间限的限制。此外，如上所述，机械手有单臂和双臂之分，双臂可以通过抓取晶圆作为缓存来大大提高加工效率，但是其调度问题也更加复杂。不仅如此，缓存腔室的有无也会影响调度策略的设计。因此，已经有不少研究学者针对集束型制造平台的资源调度问题作出研究，包括针对操作序列、操作控制、工具建模、性能分析

等方面。先前的研究主要是以周期性调度算法（即大量相同的晶圆根据相同的工艺在集束型制造设备上进行制造）为主，其中一种是加工装置以一定的周期重复相同的操作流程，把调度问题转化为求解最小重复周期问题。例如[3]，其使用了基于 Time Petri Net(TPN)的集束型制造装备周期性调度流程模型建立，利用混合整数规划（MIP）方法，求解了最优稳定工作周期工作时间，但是由于这里研究的是非周期性调度问题（晶圆数很少），集束型制造设备很有可能不能进入稳态或者只有极短的时间处于稳态，此外，还需要讨论进入稳态和退出稳态的情况，这样的算法越来越不适用于如今集束型制造设备的发展方向。如今，随着半导体的发展，晶圆制造模式经常发生改变，从而我们需要利用非周期性调度问题来进行研究。[2]中利用了基于 Time Petri Net(TPN)的分支定界算法，其基本思路是如果加工工序中每个晶圆都能无冲突地各自完成加工，那么这样的时间计算是最为简单的，因此其通过化解冲突点来解线性规划问题，从而找到两个不同的加工时间下界，并将更大的下界作为实际完成时间。

本文中，我们采取了基于[2]的方法，通过状态转移图建模方法将问题转化为最小化各个设备的完成时间的多物体问题。利用 Pareto 最优动态规划法对路径进行删减，以求得最优路径。Pareto 最优是指在两个状态之间如果拥有不同路径，那么所有元素的完成时间都较短的路径为 Pareto 最优路径，并且可以删除被支配路径。此外，我们对于[2]的算法进行了一些更改与完善，我们采用了顺序动态规划的方法，并且将该方法拓展到了双臂的机械手上。首选求解了串行工艺的加工时间，并且比较了在该算法下进行不同改动时的算法效率与结果比较，之后，我们求解并分析了并行工艺以及可重入工艺的加工时间。

2 问题提出

本文研究的问题主要基于下图的双平台集束型加工设备。



2.1 设备描述

- LLA：存放待加工的 Wafer
- LLB：存放加工完的 Wafer
- R_1：平台 1 机械手，单臂，可以同时持有一片 wafer
- R_2：平台 2 机械手，双臂，可以同时持有两片 wafer，两臂 180 度耦合，只能同时进行一个动作（取片、放片、旋转）
- $PM_{1,1}, PM_{1,2}, PM_{2,1}, PM_{2,2}, PM_{2,3}, PM_{2,4}, PM_{1,3}, PM_{1,4}$：加工腔室，每次只能加工一片 wafer
- $BM_{1,1}$：缓冲腔室，当平台 1 加工完成，可以先放在 $BM_{1,1}$ 腔室中，R_2 可以从 $BM_{1,1}$ 中取 Wafer，$BM_{1,1}$ 中最多容纳 1 片 Wafer
- $BM_{1,2}$：缓冲腔室，当平台 2 加工完成，可以放入 $BM_{1,2}$ 腔室中，R_1 可以从 $BM_{1,2}$ 中取出 Wafer，$BM_{1,2}$ 中最多容纳 1 片 Wafer

2.2 工艺介绍

待加工的晶圆存放在 LLA 中，通过机械手的取片、放片和旋转的动作，根据工艺顺序依次经过 PM 完成加工，最后放入 LLB 中。在加工的过程中，有时会出现多个动作需要完成，但每个机械手每次只能执行一个动作，如何安排它们的顺

序，会对加工时间产生很大的影响，我们需要找到最优的机械手调度顺序，以使得能在最短的时间内完成加工。

- **串行工艺：**每个工艺在一个加工腔室中进行，晶圆依次经过每个腔室完成加工

$$P_0(LLA) \to P_1(PM_{1,1},60) \to P_2(PM_{1,2},70) \to P_3(BM_{1,1},0) \to P_4(PM_{2,1},40) \to$$
$$P_5(PM_{2,2},50) \to P_6(PM_{2,3},50) \to P_7(PM_{2,4},60) \to P_8(BM_{1,2},0) \to$$
$$P_9(PM_{1,3},50) \to P_{10}(PM_{1,4},48) \to P_{11}(LLB)$$

- **并行工艺：**每个工艺可在多个腔室中进行，晶圆只需进入其中一个加工即可，多个腔室并行工作。

$$P_0(LLA) \to P_1(PM_{1,1},PM_{1,2},139) \to P_2(BM_{1,1},0) \to P_3(PM_{2,1},PM_{2,2},131) \to$$
$$P_4(PM_{2,3},PM_{2,4},130) \to P_5(BM_{1,2},0) \to P_6(PM_{1,3},PM_{1,4},120) \to P_7(LLB)$$

- **可重入工艺：**有的腔室可以进行多种工艺，需要重复进入，如下所示的 $PM_{2,2}$，需要被重复进入。

$$P_0(LLA) \to P_1(PM_{1,1},PM_{1,2},PM_{1,3},200) \to P_2(BM_{1,1},0) \to P_3(PM_{2,1},60) \to$$
$$P_4(PM_{2,2},50) \to P_5(PM_{2,3},PM_{2,4},130) \to P_6(PM_{2,2},50) \to$$
$$P_7(BM_{1,2}) \to P_8(PM_{1,4},110) \to P_9(LLB)$$

2.3 限制条件

- 加工腔室是单晶圆加工设备，每次只能加工一片 wafer
- Wafer 传递到设备中之后，只有当这个 Wafer 的所有加工工艺做完才可以从出口处取下来
- Wafer 传递到某个腔室后，会立即进行相应的工艺，加工结束后可以停留一段时间拿出
- 两机械手在缓冲模块不能发生碰撞，即只有一个机械手在缓存模块卸载 wafer 结束后，另一个机械手才能开始装载 wafer
- 机械手的装载 Wafer 的时间是 tr1(1s)，卸载 Wafer 的时间是 tr2(1s)，机械手旋转的时间是 tr3(1s)
- 每个机械手每个只能执行一个动作，但是两个机械手可以同时执行动作

2.4 研究的问题

- **考虑串行工艺**，进行 5 片 wafer 的调度优化，要求总加工时间最短，给出最优解的总加工时间，机械手动作序列及发生动作的时刻。
- **考虑并行工艺**，进行 5 片 wafer 的调度优化，要求总加工时间最短，给出最优解的总加工时间，机械手动作序列及发生动作的时刻。
- **考虑可重入工艺**，进行 5 片 wafer 的调度优化，要求总加工时间最短，给出最优解的总加工时间，机械手动作序列及发生动作的时刻。

3 解决方案

3.1 串行

3.1.1 模型建立

根据说明文档给出的状态转移图建模方法，对于工艺$P_0(LLA) \to P_1 \to P_2 \to \cdots \to P_{m-1} \to P_m \to P_{m+1}(LLB)$，定义动作$z$和状态$s$如下：

$$z = (a_1, b_1, a_2, b_2, \cdots, a_m, b_m, a_{m+1}, b_{m+1})$$

$$s = (w, c_1, d_1, c_2, d_2, \cdots, c_m, d_m, c_{m+1})$$

其中a_i表示机械手从P_{i-1}取片并旋转到下个动作位置，b_i表示机械手到P_i放片并旋转到下个动作位置；w表示 *LLA* 中待加工的片的个数，c_i表示机械手持有的从P_{i-1}取出的片的个数，d_i表示进行工艺P_i的片的个数。$m = 10$。

机械臂的动作限制条件为：

- 两机械手在缓冲模块不能发生碰撞，即只有一个机械手在缓存模块卸载 wafer 结束后，另一个机械手才能开始装载 wafer；
- 机械手的装载 Wafer 的时间是 trl(ls)，卸载 Wafer 的时间是 tr2(ls)，机械手旋转的时间是 tr3(ls)；
- 每个机械手每个只能执行一个动作，但是两个机械手可以同时执行动作。

我们对上述模型作出如下简化：

① 机械手的动作有两种情况：

Ⅰ 当$d_i = 0$即进行工艺P_i的片的个数为 0 时，机械臂仅需把完成工艺P_{i-1}的片装载、旋转并到进行工艺P_i的腔室内卸载，再旋转到下一个位置；

Ⅱ 在工作台 2 上，当机械臂需要装载但是暂时不卸载时，机械臂仅需完成装载和旋转到下一个位置两个动作。

即我们把机械臂的旋转分别和装载、卸载组合，这样做的依据是当我们设计好了加工流程后，机械臂在装载和卸载之后要旋转到的位置是确定的。

② 我们把状态s重新定义为如下形式：

$$s = (w, d_1, d_2, \dots, d_m, buffer1, buffer2, act1, act2)$$

其中，w表示 LLA 中待加工的片的个数，d_i表示进行工艺P_i的片的个数；当 buffer1 或者 buffer2 不为 0 时，说明机械臂 2 抓取了 buffer，buffer1 和 buffer2 的数值即为该 buffer 应该放的地方；act1 和 act2 分别表示机械臂 1 和机械臂 2 的动作，其表示的含义在下表中列出。

$act1 = i$	$act2 = i$		
	$14 \le i \le 18$	$23 \le i \le 27$	$i < 14$
从进行工艺$\boldsymbol{P_{i-1}}$的腔室到进行工艺$\boldsymbol{P_i}$的腔室	到进行工艺P_{i-10}的腔室卸载 *buffer*	从进行工艺P_{i-20}的腔室装载 *buffer*	从进行工艺P_{i-1}的腔室到进行工艺P_i的腔室

③ 我们把动作z重新定义为如下形式：

$$z = (z_1, z_2, \dots, z_m, buffer1, buffer2)$$

其中，z_i表示从第$i-1$个腔拿起工件放到第i个腔,最后 2 列为 buffer 位，1 表示双臂已经拿了一个工件，0 表示空。

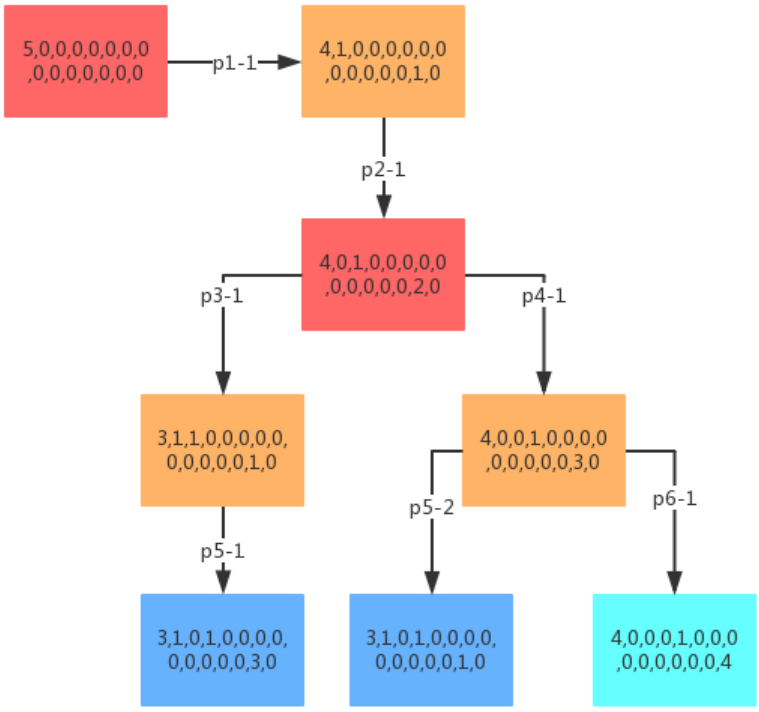
于是，初始状态可以表示为：

$$s_0 = (5,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

最终状态可以表示为：

$$s_{end} = (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

这里给出状态转移图的前几步：



图中 pi-j 同说明文档里的p_i^j。

至此，串行工艺的模型建立完毕。

3.1.2 方案概述

根据上述模型，我们要找到从s_0到s_{end}的最佳路径。如概述中所述，我们采用了[2]中的动态规划法。其基本的流程如下：

Ⅰ创建初始状态s_0，进入第Ⅱ步；

Ⅱ判断当前代是否只有一个节点并且该节点是s_{end}？如果是，进入第Ⅴ步；如果不是，从当前代的第一个节点开始产生子代，并分别计算它们的每一个机械臂和腔室的 Readytime，进入第Ⅲ步；

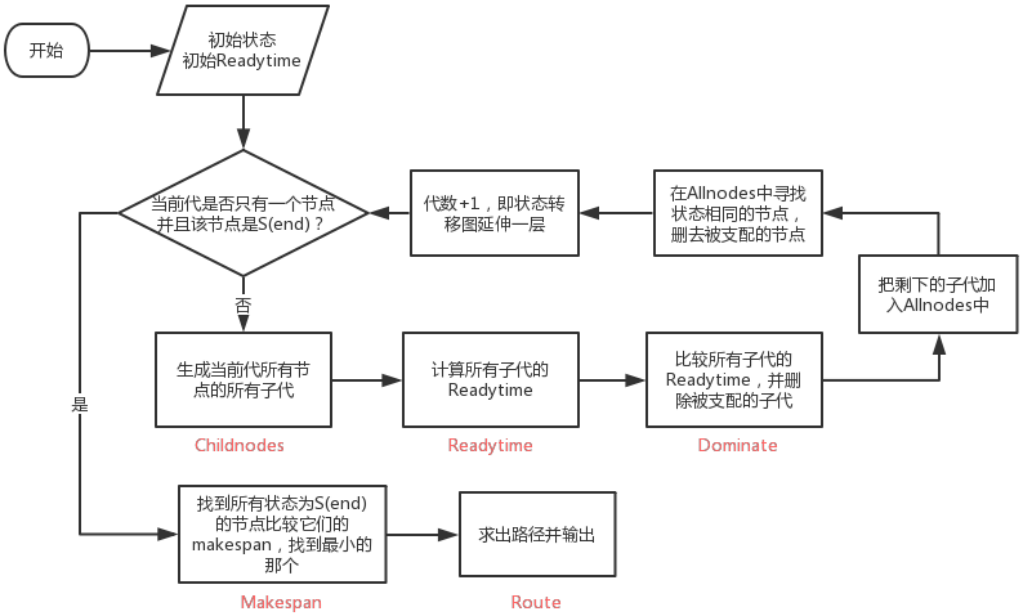
Ⅲ对当前产生的所有子代的 Readytime 进行比较，删去被支配的子代，并把未被删去的子代加入 Allnodes 中，进入第Ⅳ步；

Ⅳ在 Allnodes 中寻找相同的节点（不同代），比较它们的 Readytime 并删去被支配的节点，返回第Ⅱ步；

Ⅴ在 Allnodes 中寻找所有的s_{end}，比较它们的 makespan，找到最小的那个节点，进入第Ⅵ步；

Ⅵ根据这个节点的相关信息找到产生这一最小 makespan 的路径并输出。

流程图如下：



如图 2 是主程序流程图，其中生成子代（3.1.3 Childnodes）、计算 Readytime（3.1.4 Readytime）、删除被支配的子代（3.1.5 Dominate）、找到最小的 Makespan（3.1.6 Makespan）和输出路径（3.1.7 Route）的详细说明见下述各小节。

在上述算法中，值得注意的是在 Allnodes 中寻找相同的节点时，只需要满足

$$s_i(1{:}m+3) = s_j(1{:}m+3)$$

即前 m+3 位表示的是当前所有腔室以及机械臂中晶圆个数的情况，当它们对应相等时即认为时“同一个”状态；而后两位仅仅代表当前节点是如何从上一步转移来的，并不是我们关心的点。

3.1.3 Childnodes

从一个节点生成子代时主要考虑以下规则：

父代	子代	对应操作
s(1)=a(1≤a≤5)&s(2)=0	s(1)=a-1&s(2)=1	**在 LLA 装载晶圆并在 PM1l 卸载**
s(11)=1	s(11)=0	**在 PM14 装载晶圆并在 LLB 中卸载**

s(i-1)=1&s(i)=0	s(i-1)=0&s(i)=1	在工艺 P(i-1)腔室中装载晶圆并在工艺 P(i)腔室中卸载
s(12)≠0	s(s(12)-20)=1	机械臂 2 上 buffer 在工艺 P(s(12)-20)腔室卸载
s(13)≠0	s(s(13)-20)=1	机械臂 2 上 buffer 在工艺 P(s(13)-20)腔室卸载
s(i-1)=1&s(i)=1& Readytime(i-1)<Readytime(i) &(s(12)=0||s(13)=0)&4<i<10	s(i-1)=0 s(12)=i||s(13)=i	在工作台 2 上，如果前一个腔室比后一个腔室先 Ready，可以抓取前一个腔室里的晶圆作为 buffer 并且修改相应的 buffer 位

每一步有且仅有一个机械臂完成上表中某一行对应的动作。

装载 buffer 的思想是当第 i 道工序已经完成而第 i+1 道工序仍未完成，此时先把第 i 道工序的晶圆装载到机械臂上；如果第 i-1 道工序也已经完成，机械臂 2 另一个臂可以将其挪到第 i 道工序进行加工，而不会由于第 i+1 道工序没有完成，导致即使第 i-1 和第 i 道工序都已经完成，完成了第 i-1 道工序的晶圆也无法进行第 i 道工序的加工；而当第 i+1 道工序终于加工好时，机械臂 2 另一个没有装载着 buffer 的臂可以装载第 i+1 道工序的晶圆，装载着 buffer 的机械臂可以立即卸载 buffer。

生成子代的这一步被我们封装在 childnodesnew4 函数中，输入一个当前代的状态和其 Readytime，输出所有可能的子代。

3.1.4 Readytime

首先规定几个符号：

①PMTime：腔室的加工时间；②trans4=4：装载+旋转+卸载+旋转的时间；

③trans3=3：装载+旋转+卸载的时间；④trans2=2：装载+旋转的时间；

⑤transl=1：装载的时间

计算生成的子代的 Readytime 时，分以下几种情况：

- **机械臂 1**

- **在 LLA 装载晶圆并在 PM1l 里卸载晶圆**

机械臂 1 时间=原时间+trans4；

PM11 时间=max(机械臂 1 原时间，PM11 原时间)+trans3+PMTime(PM11)；

- **在 PM14 装载晶圆并在 LLB 里卸载晶圆**

机械臂 1 时间= max(机械臂 1 原时间， PM14 原时间)+trans4；

PM14 时间= max(机械臂 1 原时间， PM14 原时间)+trans3；

- **在工艺 P(i-1)腔室中装载晶圆并在工艺 P(i)腔室中卸载**

机械臂 1 时间= max(机械臂 1 原时间，装载处原时间，卸载处原时间)+trans4；

卸载处时间= max(机械臂 1 原时间，装载处原时间，卸载处原时间)+trans3+PMTime(卸载处)；

- **机械臂 2**
  - **作为单臂使用：在工艺 P(i-1)腔室中装载晶圆并在工艺 P(i)腔室中卸载**

机械臂 2 时间= max(机械臂 2 原时间，装载处原时间，卸载处原时间)+trans4；

卸载处时间= max(机械臂 2 原时间，装载处原时间，卸载处原时间)+trans3+PMTime(卸载处)；

  - **取 buffer**

机械臂 2 时间= max(机械臂 2 原时间， 装载处原时间)+trans2；

装载处时间= max(机械臂 2 原时间， 装载处原时间)+trans1；

  - **放 buffer**

机械臂 2 时间= max(机械臂 2 原时间， 卸载处原时间)+trans2；

卸载处时间= max(机械臂 2 原时间，卸载处原时间)+trans1+ PMTime(卸载处)；

根据上述原则计算处每一个子代的 Readytime 之后存入 Temptime，然后根据以下原则修改腔室时间：

- Readytime(PM)=max(Readytime(PM),Readytime(对应机械臂))；
- Readytime(BM)=max(Readytime(BM),min(Readytime(机械臂 1), Readytime(机械臂 2))

修改完腔室时间之后，我们把子代状态和对应的 Readytime 存入 TempReadytime 矩阵中。

3.1.5 Dominate

根据动态规划的原则，如果某一个子代的 Readytime 的各分量都对应大于另一子代的 Readytime 的各分量，则认为前者被后者支配(dominate)，可以证明前者一定不会出在最优路径中，于是可以删去前者。

我们的算法中，首先相同代的节点之间会相互比较，即找到状态相同的两个子代，若出现了支配的情况则删除被支配的子代，未被删除的子代会被存入 Allnodes 矩阵中；其次考虑到有双臂动作，可能出现不同代产生两个相同状态的节点（即不同的步数可能走到同一个位置），于是在 Allnodes 中我们也会寻找相同的节点并比较，从而删去被支配的节点。

3.1.6 Makespan

对于每一条完整的路径，由于在修改腔室时间中我们采取了如下规则：

Readytime(PM)=max(Readytime(PM),Readytime(对应机械臂))

于是 makespan 为两个机械臂时间的最大值。

在寻找最小的 makespan 的过程中，我们首先设 makespan=10000，然后比较所有的s_{end}的 makespan，如果小于当前的 makespan 就用该值代替当前的 makespan，最终找到最小的一个。最后，由于机械臂每一次动作之后都会旋转到下一次动作的位置，最后一次也默认旋转，但其实最后一次的旋转是不必要的，因此找到最小的 makespan 之后我们还要对其-1.

3.1.7 Route

在 3.1.5 小节中我们提到，TempReadytime 矩阵中存储了子代状态和对应的 Readytime，除此以外，我们还存储了子代的序号（即它是当代第几个元素）、父代序号（其父代是上一代第几个元素）和父代代数。根据这些信息，我们在如 3.1.6 中所述求得了最小的 makespan 之后，就可以逐步向上找到该路径经过的每一个节点，直到回到s_0。得到了这一条路径上的所有节点之后，根据状态 s 的定义，s 的最后两列就是我们需要的动作序列。

3.2 并行

对于并行和可重入工艺来说，在算法流程、判断支配关系、删除支配节点、寻找最小的 makespan 和搜索路径中与串行工艺并没有区别，我们在这一节中不在赘述，仅对并行的模型（3.2.1）、产生子代的可能方式（3.2.2）和 Readytime（3.2.3）进行说明。

3.2.1 并行模型

我们把状态s重新定义为如下形式：

$$s = (w, d_1, d_2, \dots, d_m, buffer1, buffer2, act1, act2)$$

其中，m 表示工艺数量（注意不是腔室的数量），w表示 LLA 中待加工的片的个数，d_i表示进行工艺P_i的片的个数；当 buffer1 或者 buffer2 不为 0 时，说明机械臂 2 抓取了 buffer，buffer1 和 buffer2 的数值即为该 buffer 应该放的地方；act1 和 act2 分别表示机械臂 1 和机械臂 2 的动作，其表示的含义与串行基本相同，仅在序号上有一些区别，此处不再赘述。

工艺和状态与腔室的对应关系如下：

工艺/s	P_1/d_1	P_2/d_2	P_3/d_3	P_4/d_4	P_5/d_5	P_6/d_6
腔室	PM11,PM12	BM11	PM21,PM22	PM23,PM24	BM22	PM13,PM14
max	2	1	2	2	1	2

3.2.2 Childnodes

并行工艺在生成子代时，需要遵循以下规则：

- 机械臂 1 和机械臂 2 作为单臂使用

  此时机械臂 2 需要满足的条件是至少有一个臂是空的。

  当前一道工艺已经完成，并且其后的工艺对应的腔室不满（<max）时，机械臂可以把晶圆往后转移以完成下一道工序，前一道工序对应的状态-1，后一道工序对应的状态+1。

- 机械臂 2 装载 buffer

  装载 buffer 的思想同串行工艺。当前一道工艺已经完成，而其后的工艺对应

的腔室已经满载（=max）时，机械臂 2 可以抓取前一道工艺的晶圆作为 buffer，并修改相应的 buffer 位，存储当前 buffer 将要卸载的腔室，前一道工序对应的状态-1。

- 机械臂 2 卸载 buffer

  当机械臂 2 装载的 buffer 将要卸载的腔室不满（<max）时，机械臂 2 可以在该腔室卸载 buffer，并把相应的 buffer 位置零，后一道工序对应的状态+1。

3.2.3 Readytime

并行工艺的腔室时间修改规则与串行相同。

在并行模型中，同一道工序的两个腔室被合并到了同一个状态位置，但是不同的腔室的 Readytime 是不一样的。分析之后发现，卸载处的腔室若已经有了一个晶圆，其所在腔室的 Readytime 大于同一道工序空着的那个腔室的 Readytime；卸载处的腔室若为空，也应该在先加工好即 Readytime 较小的腔室里卸载；同理，装载处的腔室若有且仅有一个晶圆，其所在腔室的 Readytime 大于同一道工序空着的那个腔室的 Readytime；装载处的腔室若有两个晶圆，则装载的应该是先加工好即 Readytime 比较小的那个腔室里的晶圆。

于是我们给出 minP 和 maxP 两个时间矩阵，分别存储同一道工序两个腔室较小和较大的 Readytime 及其对应的腔室序号。根据以上分析，卸载处的腔室时间应用 minP，装载处若不满，则应用 maxP，反之用 minP。

计算生成的子代的 Readytime 时，分以下几种情况：

- **机械臂 1**
  - **在 LLA 装载晶圆并在 P1 里卸载晶圆**

    机械臂 1 时间=原时间+trans4；

    minP1 时间=max(机械臂 1 原时间，minP1 原时间)+trans3+PMTime(P1)；
  - **在 P6 装载晶圆并在 LLB 里卸载晶圆**
    - **P6 里只有一个晶圆**

      机械臂 1 时间= max(机械臂 1 原时间，maxP6 原时间)+trans4；

      maxP6 时间= max(机械臂 1 原时间，maxP6 原时间)+trans3；
    - **P6 里有两个晶圆**

机械臂 1 时间= max(机械臂 1 原时间，minP6 原时间)+trans4；

minP6 时间= max(机械臂 1 原时间，minP6 原时间)+trans3；

- **在工艺 P(i-1)腔室中装载晶圆并在工艺 P(i)腔室中卸载**
  - **P(i-1)里只有一个晶圆**

机械臂 1 时间= max(机械臂 1 原时间，maxP(i-1)原时间，minP(i)原时间)+trans4；

minP(i)时间= max(机械臂 1 原时间，maxP(i-1)原时间，minP(i)原时间)+trans3+PMTime(卸载处)；

  - **P(i-1)里只有一个晶圆**

机械臂 1 时间= max(机械臂 1 原时间，minP(i-1)原时间，minP(i)原时间)+trans4；

minP(i)时间= max(机械臂 1 原时间，minP(i-1)原时间，minP(i)原时间)+trans3+PMTime(卸载处)；

- **机械臂 2**
  - **作为单臂使用：在工艺 P(i-1)腔室中装载晶圆并在工艺 P(i)腔室中卸载**
    - **P(i-1)里只有一个晶圆**

机械臂 1 时间= max(机械臂 1 原时间，maxP(i-1)原时间，minP(i)原时间)+trans4；

minP(i)时间= max(机械臂 1 原时间，maxP(i-1)原时间，minP(i)原时间)+trans3+PMTime(卸载处)；

    - **P(i-1)里只有一个晶圆**

机械臂 1 时间= max(机械臂 1 原时间，minP(i-1)原时间，minP(i)原时间)+trans4；

minP(i)时间= max(机械臂 1 原时间，minP(i-1)原时间，minP(i)原时间)+trans3+PMTime(卸载处)；

  - **取 buffer**
    - **装载处只有一个晶圆**

机械臂 2 时间= max(机械臂 2 原时间，maxP 装载处原时间)+trans2；

maxP 装载处时间= max(机械臂 2 原时间，maxP 装载处原时间)+trans1；

- **装载处有两个晶圆**

  机械臂 2 时间= max(机械臂 2 原时间，minP 装载处原时间)+trans2；

  minP 装载处时间= max(机械臂 2 原时间，minP 装载处原时间)+trans1；

- 放 buffer

机械臂 2 时间= max(机械臂 2 原时间，minP 卸载处原时间)+trans2；

minP 卸载处时间= max(机械臂 2 原时间，minP 卸载处原时间)+trans1+PMTime(卸载处)。

3.3 可重入

3.3.1 可重入模型

可重入工艺在并行工艺的基础上，增加了 PM22 的可重入属性，对此我们构建性的工艺和状态与腔室的对应关系如下：

工艺/状态	P_1/d_1	P_2/d_2	P_3/d_3	P_4/d_4
腔室	PM11,PM12,PM13	BM11	PM21	PM22
max	3	1	1	1
工艺/状态	P_5/d_5	P_6/d_6	P_7/d_7	P_8/d_8
腔室	PM23,PM24	PM22	BM22	PM14
max	2	1	1	2

3.3.2 Childnodes

可重入工艺在生成子代时，需要遵循以下规则：

- 机械臂 1 和机械臂 2 作为单臂使用

  此时机械臂 2 需要满足的条件是至少有一个臂是空的。

  当前一道工艺已经完成，并且其后的工艺对应的腔室不满（<max）时，机械臂可以把晶圆往后转移以完成下一道工序，前一道工序对应的状态-1，后一道工序对应的状态+1。

  对于P_4和P_6，判断 PM22 这个可重入腔室是否为满的条件是，如果$d_4 + d_6 = 1$，说明 PM 22 已满。

- 机械臂 2 装载 buffer

  装载 buffer 的思想同串行工艺。当前一道工艺已经完成，而其后的工艺对应的腔室已经满载（=max）时，机械臂 2 可以抓取前一道工艺的晶圆作为 buffer，并修改相应的 buffer 位，存储当前 buffer 将要卸载的腔室，前一道工序对应的状态-1。

  在可重入工艺中，我们需要考虑到以下情况：

  - 当有晶圆在进行P_4工艺时，我们认为 PM22 是满的，如果有晶圆已经完成P_6加工，而进行P_4加工的晶圆还没有完成，此时可以抓取完成P_6加工的晶圆作为 buffer；
  - 当有晶圆在进行P_6工艺时，我们认为 PM22 是满的，如果有晶圆已经完成P_4加工，而进行P_6加工的晶圆还没有完成，此时可以抓取完成P_4加工的晶圆作为 buffer；

  我们采取的解决办法是建立一个新的临时状态，在这个状态中若 PM22 腔室里有晶圆，则把d_4和d_6都置为 1.当考虑抓取P_3或者P_5作为 buffer 时通过这个临时状态就可以判断了。而当要抓取P_4或P_6作为 buffer 时则需要进一步判断原状态是否为 1，即是否真的有晶圆在进行加工。

- 机械臂 2 卸载 buffer

  当机械臂 2 装载的 buffer 将要卸载的腔室不满（<max）时，机械臂 2 可以在该腔室卸载 buffer，并把相应的 buffer 位置零，后一道工序对应的状态+1。判断 PM22 是否满的条件同上。

3.3.3 Readytime

可重入工艺的腔室时间修改规则与串行相同。

由于可重入工艺中也有并行，我们也给出 minP 和 maxP 两个时间矩阵，分别存储同一道工序几个腔室较小和较大的 Readytime 及其对应的腔室序号。卸载处的腔室时间应用 minP，装载处若不满，则应用 maxP，反之用 minP。

计算子代的 Readytime 的方法与并行基本相同，需要注意的是当操作涉及到可重入的腔室 PM22 时，P_3和P_6的时间都应该置为二者中较大的那个。

4 结果分析

4.1 串行

4.1.1 动作序列

我们输出动作序列如下所示：

1 0→2 0→1 0→3 0→0 4→0 5→2 0→1 0→3 0→0 4→0 6→0 5→0 7→2 0→1 0→3 0→0 4→0 6→0 5→0 8→0 7→2 0→1 0→9 0→3 0→0 4→0 6→0 5→0 8→0 7→2 0→10 0→9 0→3 0→0 4→0 6→0 5→0 8→0 7→0 6→11 0→10 0→9 0→0 8→0 7→11 0→10 0→9 0→0 8→11 0→10 0→9 0→11 0→10 0→11 0

其中如果每一个步骤的第一个数字为 0，代表只有机械手 2 在操作，如果第二个数字为 0，则代表只有机械手 1 在操作，由于机械手 2 实际上工作于单臂模式（即可以看作是一个臂的机械手，所以不需要对两臂进行任何的区分）。此外数字代表着将要搬运到的工作腔室，1 代表机械手将晶圆从 LLA 搬运到 PM11，2 代表从 PM11 搬运到 PM12，3 代表从 PM12 搬运到 BM11，4 代表从 BM11 到 PM21，5 代表从 PM21 到 PM22，6 代表从 PM22 到 PM23，7 代表从 PM23 到 PM24，8 代表从 PM24 到 BM12，9 代表从 BM22 到 PM13，10 代表从 PM13 到 PM14，11 代表从 PM14 到 LLB（也即该晶圆加工完成）。为了便于助教查验与理解，我们在附录 A 中附上了串行状态。

4.1.2 对应的动作时间

0→63→67→136→140→140→143→144→213→217→217→220→221→239→263→290→294→294→297→298→316→340→354→355→367→37→1371→374→393→407→417→431→432→444→448→451→458→470→484→494→508→509→523→535→547→561→585→586→600→612→638→662→689→715→766

4.1.3 工序时间

最终 5 个晶圆加工完成的工序时间为 769（程序显示的为 770），如前面的讨论所

述，最后一个加工时间实际上多加了 1s 的机械手旋转时间，这样的旋转是没有必要的，我们可以在最后的结果中将其剔除。

4.1.4 CPU 时间

```
>> serialnew9
时间已过 2.181157 秒。
```

如上图所示，串行工艺求解时长在 2.18s 左右。

4.2 并行

4.2.1 动作序列

1 2 0 0→1 3 0 0→2 4 0 0→0 0 4 5→1 2 0 0→3 4 0 0→0 0 4 6→1 3 0 0→0 0 5 7→0 0 6 8→2 4 0 0→0 0 4 5→1 2 0 0→3 4 0 0→0 0 4 6→0 0 7 9→9 10 0 0→0 0 8 9→9 11 0 0→0 0 5 7→2 4 0 0→0 0 6 8→0 0 4 5→10 12 0 0→11 12 0 0→0 0 7 9→9 10 0 0→0 0 8 9→9 11 0 0→0 0 5 7→0 0 7 9→10 12 0 0→11 12 0 0→9 10 0 0→10 12 0 0

由于是并行工艺，那么一个工艺对应着两个腔室，我们需要表示机械手从哪个腔室取件，到哪个腔室放件，所以我们每个动作用前两个数字表示单臂机械手的动作，用后两个数字表示双臂机械手的动作（这里双臂机械手还是处于单臂工作方式）。1 代表 LLA，2 代表 PM11，3 代表 PM12，4 代表 BM11，5 代表 PM21，6 代表 PM22，7 代表 PM23，8 代表 PM24，9 代表 BM12，10 代表 PM13，11 代表 PM14，12 代表 LLB。

4.2.2 对应的动作时间

0→4→142→145→146→150→153→154→279→283→288→291→292→296→299→412→415→416→423→425→427→429→437→538→542→558→561→562→569→570→575→684→688→707→711

4.2.3 工序时间

最终 5 个晶圆加工完成的工序时间为 833（程序显示的为 834），理由同上。

4.2.4 CPU 时间

```
>> parallel4
时间已过 0.540634 秒。
```

如上图所示，并行工艺求解时长在 0.54s 左右。

4.3 可重入

4.3.1 动作序列

1 2 0 0→1 3 0 0→1 4 0 0→2 5 0 0→0 0 5 6→3 5 0 0→0 0 5 100→1 2 0 0→1 3 0 0→4 5 0 0→0 0 6 7→0 0 100 6→0 0 5 100→0 0 7 8→0 0 6 7→0 0 100 6→0 0 7 9→0 0 6 7→2 5 0 0→0 0 5 6→3 5 0 0→0 0 7 100→0 0 8 7→0 0 100 8→0 0 6 100→0 0 5 6→0 0 7 10→10 11 0 0→0 0 100 7→0 0 9 100→0 0 7 9→0 0 100 7→0 0 7 10→0 0 6 7→11 12 0 0→0 0 8 100→10 11 0 0→0 0 7 8→0 0 100 7→0 0 7 10→0 0 9 7→11 12 0 0→10 11 0 0→0 0 7 10→0 0 8 7→11 12 0 0→0 0 7 10→10 11 0 0→11 12 0 0→10 11 0 0→11 12 0 0

由于存在并行工艺，那么一个工艺对应着两个或以上腔室，我们需要表示机械手从哪个腔室取件，到哪个腔室放件，所以我们每个动作用前两个数字表示单臂机械手的动作，用后两个数字表示双臂机械手的动作，其中如果双臂机械手的动作序列中 100 在后，即是从前面的数字代表的腔室取工件作为 buffer，如果 100 在前，则代表把机械臂上持有的 buffer 放在后面的数字代表的腔室。1 代表 LLA，2 代表 PM11，3 代表 PM12，4 代表 BM11，5 代表 PM21，6 代表 PM22，7 代表 PM23，8 代表 PM24，9 代表 BM12，10 代表 PM13，11 代表 PM14，12 代表 LLB。此外，可以发现 0 0 7 10→10 11 0 0 似乎有冲突，但是结合下面的动作时间发现 861 时刻双臂机械手取件到要在 10 卸载时已经为 863，但是 862 时间单臂机械手从 10 取件，所以并没有冲突。

4.3.2 对应的动作时间

0→4→8→203→206→207→210→211→215→219→269→273→275→322→334→338→387→399→414→417→418→452→455→459→480→482→508→511→512→520→563→567→618→622→624→626→628→675→679→730→734→741→745→787→791→858→861→862→975→979→1092

4.3.3 工序时间

最终 5 个晶圆加工完成的工序时间为 1095（程序显示的为 1096），理由同上。

4.3.4 CPU 时间

```
>> Reentrant2
时间已过 2.029840 秒。
```

如上图所示，并行工艺求解时长在 2.03s 左右。

5 作业总结

5.1 遇到问题和解决方案

5.1.1 模型建立

作业文档中一开始建议的状态建立为

$$z=(a_1,b_1,a_2,b_2,\cdots,a_m,b_m,a_{m+1},b_{m+1})$$

$$s=(w,c_1,d_1,c_2,d_2,\cdots,c_m,d_m,c_{m+1})$$

其中a_i表示机械手从P_{i-1}取片并旋转到下个动作位置，b_i表示机械手到P_i放片并旋转到下个动作位置；w表示 *LLA* 中待加工的片的个数，c_i表示机械手持有的从P_{i-1}取出的片的个数，d_i表示进行工艺P_i的片的个数。

但是在做作业过程中，我们发现对于机械臂的每个位置信息建立状态过于麻烦，而且比较累赘。因此我们简化了模型，把机械臂的旋转、装载、卸载组

合，并且保证卸载之后旋转到下一个抓取的位置。将状态与动作序列重新定义为

$$z = (z_1, z_2, \dots, z_m)$$

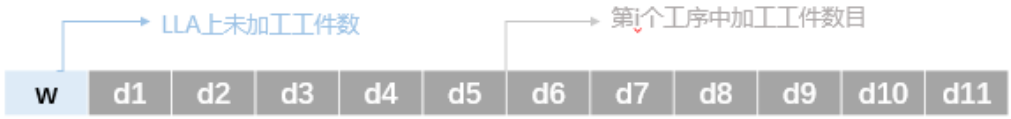
$$s = (w, d_1, d_2, \dots d_m)$$

$$s' = s + z \times A$$

其中 m 为工序数，在串行中即为 11；动作序列 z_i 即机械臂从第 i 个工作腔中拿起，并直接放入第 i+1 个工作腔。状态序列第一位为 LLA 上待加工的工件个数，d_i 表示第 i 道工序中加工工件数目。

动作序列：

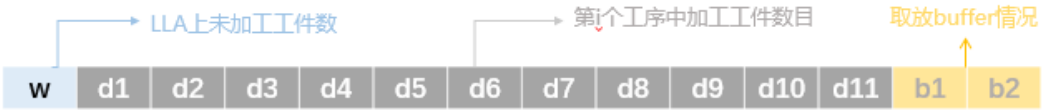
状态序列：



但在做题过程中以及和延渊渊助教的讨论过程中，我们发现对题意的理解出了些问题。一开始我们认为是双臂机械手同时起同时落，并且统一动作即同时抓同时放，结果讨论后发现，双臂机械手虽然同时转动起落，但是每一次只能控制其中一个爪子抓放。因此我们又增加了两个状态变量存储双臂机械手的动作情况：即有无抓取 buffer，从哪里抓取的 buffer，要放到哪个位置。（buffer 即目前从工作腔中抓取并且暂时不能直接放到下一个工作腔中的工件。）

动作序列：

状态序列：



其中 buffer1 、buffer2 位为 k+10 时，表示下一步要卸载到第 k 个工作腔（工序）内，为 k+20 时，表示双臂从第 k 个工作腔（工序）中抓取工件作为 buffer。虽然这样可以较好建立各个工序以及机械臂中工件数目的模型，但是也就不能简单地用 $s' = s + z * A$ 来产生下一代状态（子代状态），因此，我们想到将状态变量分为两个部分，机械臂 1 操作的工作腔部分，机械臂 2 操作的工作腔部分。

由于机械臂 1 为单臂，与 buffer 无关，因此依旧可以用 $s' = s + z * A$ 来生成子代状态中机械臂 1 操作的工作腔部分状态，而机械臂 2 操作的工作腔部分我们进行了分情况讨论：

- 当机械臂 2 作为单臂使用，即既不要抓取 buffer，也不用卸载 buffer 时，仍采用 $s' = s + z * A$ 来生成子代状态序列，并且状态序列的 buffer 位不作变动
- 当机械臂 2 抓取工件作为 buffer 时，buffer 位做相应改变，发生动作的工序的工件数也做相应改变（-1）
- 当机械臂 2 卸载 buffer 时，buffer 位做相应改变（变成 0），buffer 放下的位置的工序的工件数也做相应改变（+1）

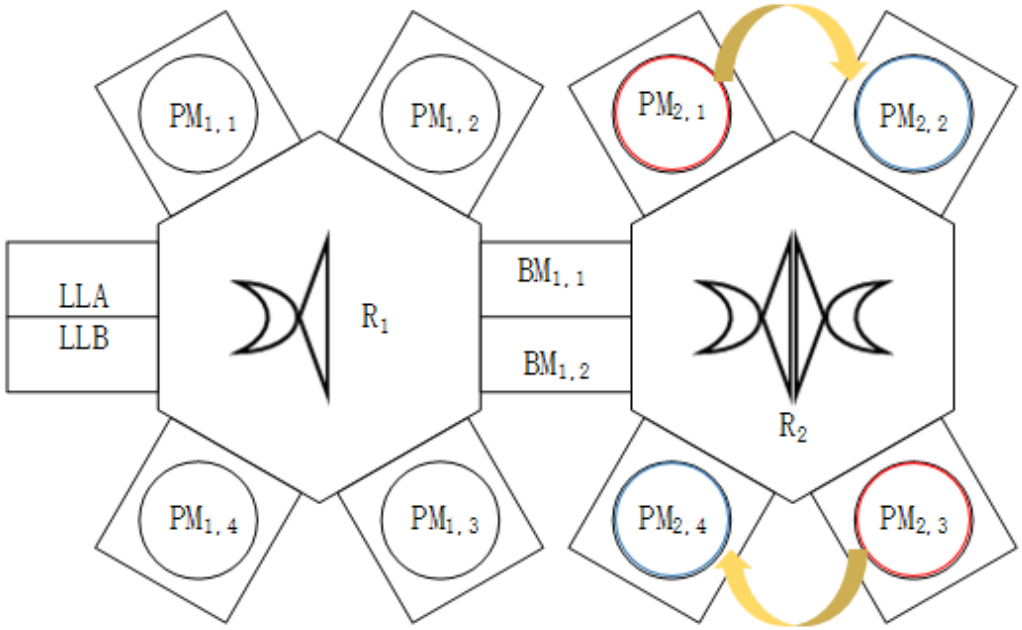
最后将两部分状态序列进行叠加。但这样以来，我们就需要增加两个变量来记录发生动作的位置，来计算 Ready time（即实现这个状态所花费的时间）。因此最终序列如下：

动作序列：

状态序列：

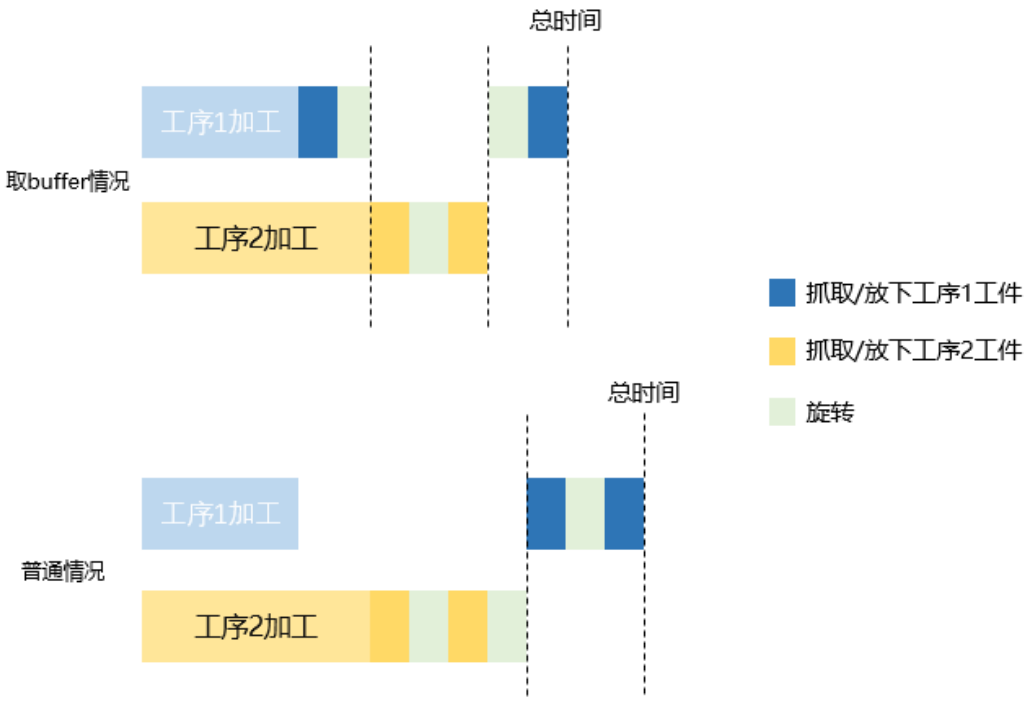
5.1.2 机械双臂的分析

之前说了，在我们一开始考虑时，认为是双臂机械手同时起同时落，并且统一动作即同时抓同时放，因此发现双臂只用在一种情况下会起到特殊作用，即$PM_{2,1}$，$PM_{2,3}$上都有工件，而$PM_{2,2}$，$PM_{2,4}$都为空，则双臂机械手可以转到相应位置，同时从$PM_{2,1}$，$PM_{2,3}$抓取工件，然后转 60°，放到$PM_{2,2}$，$PM_{2,4}$中。

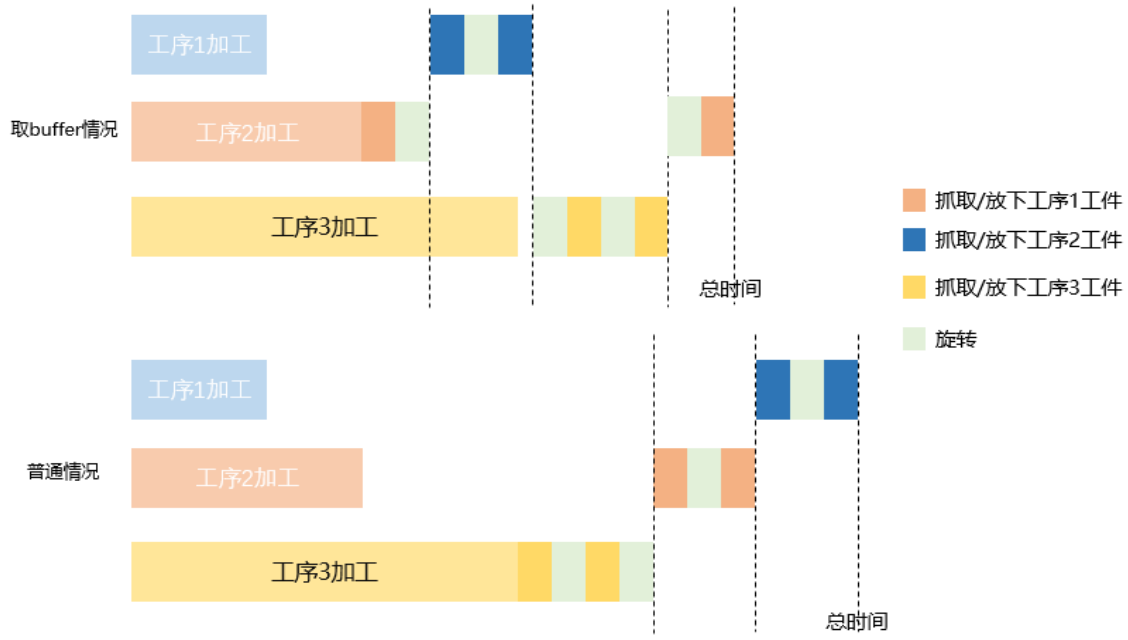


而理清双臂作用之后，我们发现双臂在以下几种情况会有优势。

- 当前后 2 个工作腔中都有工件，而较前工序的工件完成更早，可以先抓取较前工序的工件作为 buffer，等到较后工序的工件完成后将后者抓取，并将之前抓取的工件放到相应位置。这样一来，就可以节省机械臂 2 抓取使用的 1s 时间。如下图所示：



- 当前后 3 个工作腔中都有工件，并且完成时间按照工序依次增加，可以先抓取中间工序里的工件作为 buffer，再将较前工序的工件抓取并放置在中间工序，之后等到较后工序的工件完成后，进行移动工件放置 buffer 的动作。这样就可以提前加工较前工序的工件，并且节省机械臂 2 抓取时间。如下图所示：



依次类推，当前后 4 个、5 个工序都有工件也同理（虽然没有这么多工序）。同时我们发现，可以仅考虑前后两个工作腔中有工序的情况，而其他情况比如 3 个、

4 个工序都有工件的情况生产的子代状态序列都包含在其中。

5.1.3 重入的模型分析

$$P_0(LLA) \to P_1(PM_{1,1}, PM_{1,2}, PM_{1,3}, 200) \to P_2(BM_{1,1}, 0) \to P_3(PM_{2,1}, 60) \to$$
$$P_4(PM_{2,2}, 50) \to P_5(PM_{2,3}, PM_{2,4}, 130) \to P_6(PM_{2,2}, 50) \to$$
$$P_7(BM_{1,2}) \to P_8(PM_{1,4}, 110) \to P_9(LLB)$$

在一开始考虑可重入时，我们想到可以添加变量来记录工件是第几次进入这个重入腔室。如果是第一次进入重入工作腔$\mathrm{PM}_{2,2}$，则下一个要进入的工作腔为$\mathrm{PM}_{2,3}$或者 $\mathrm{PM}_{2,4}$；而如果是第二次进入，则下一个要进入的工作腔为 $\mathrm{BM}_{1,2}$.但是这样一来，即增加了变量，也增加了判断，会使得时间复杂度与空间复杂度都变大，而且在计算完成时间改动较大，比较麻烦。

因此我们想到，可以将 P_6 （即第二次进入重入工作腔）当作一个全新的工序，而 P_6 与 P_4 的工件数也互相独立，在此基础上，我们增加了两个约束条件：

- 当有工件要放到可重入腔（P_4 或 P_6）时，需要满足两个工序的总工件数目为 0；即

$$\mathrm{s}_5 + s_7 = 0$$

（s_i 表示第 $\mathrm{i}+1$ 道工序中加工工件的数目）

- 在计算 Readytime 时，考虑重入工作腔的两个工序的最大加工完成时间，并改变完成时间小的工序的时间。

这样一来，我们就可以沿袭之前的方法来计算可重入的完成时间了。

5.1.4 并行工艺 Readytime 的处理

在串行工艺中，我们计算 Readytime 的方法已经在上文有所叙述，大体思路即上一状态完成时间加上本状态的动作时间。但是有并行工作腔时，所要考虑的问题就多了一些。由于我们记录的状态序列只记录在某道工序上的加工工件数，与这道工序的哪个工作腔室内无关，因此在计算 Readytime 时，我们需要比较相同工序不同并行腔的完成时间（Readytime 记录的是每一个腔室的完成时间，不论是否为并行腔）。比如在并行工艺中， $\mathrm{PM}_{2,3}$ 与 $\mathrm{PM}_{2,4}$ 为两个并行工作腔，工序为

P_4 ，状态序列中的序号为 s_5 ，那么有工件要进入这道工序时有如下几种情况：

- 如果此时 $s_5 = 0$，即 $PM_{2,3}$ 与 $PM_{2,4}$ 中都没有工件，则将工件放入 Readytime 较小的腔室内；
- 如果此时 $s_5 = 1$，即 $PM_{2,3}$ 与 $PM_{2,4}$ 中有一个腔室有工件，那么显然可以知道，有工件的腔室一定是 Readytime 较大的腔室，因此同样将工件放入 Readytime 较小的腔室内

即放置工件时，都放入 Readytime 更小的工作腔内。但是从并行腔中取工件会稍微复杂一些。仍旧看刚刚的例子，若要从这道工序中取工件，则有如下几种情况：

- 如果此时 $s_5 = 1$，即 $PM_{2,3}$ 与 $PM_{2,4}$ 中有一个腔室有工件，那么显然可以知道，有工件的腔室一定是 Readytime 较大的腔室，因此我们取 Readytime 较大腔室的工件，（即子代 Readytime 在这个工作腔中按一定规则进行变化）；
- 如果此时 $s_5 = 2$，即 $PM_{2,3}$ 与 $PM_{2,4}$ 中都有工件，那么 Readytime 小的腔室中的工件更先完成，因此我们先取这个工件。

而当有 3 个并行腔时，情况会更加复杂一些，以可重入工艺的 $P_1(PM_{1,1}, PM_{1,2}, PM_{1,3})$ 为例，若从这个工序中取工件，会有如下 3 中情况：

- 如果此时 $s_2 = 1$，即 3 个并行腔中有一个腔室有工件，那么显然可以知道，这个腔室一定是 Readytime 最大的，因此就从最大的工作腔内取工件；
- 如果此时 $s_2 = 2$，即 3 个并行腔中有两个腔室有工件，同理，我们应当选取的是工件最先加工完成的腔室，而有工件的腔室 Readytime 一定大于没有工件的，因此我们从 Readytime 居中的工作腔内取件；
- 如果此时 $s_2 = 3$，即 3 个并行腔中都有工件，显然，我们应当从 Readytime 最小的工作腔内取件。

为了实现上述目的，我们设计了 minP，maxP，midP（仅在可重入工艺中）来分别记录各个工序中 Readytime 最小，最大，居中的工作腔序号以及他们对应的 Readytime，来更加简便地计算子代状态完成时间。

5.1.5 Pareto 最优和支配关系的确立

刚开始做这道题的时候，我们还没有正式开始学习动态规划相关内容，仅自学了教材与作业要求中给的文献。一开始看文献时，我们理解错了被支配路径的意义，认为是同一代节点中，Makespan 最小的节点会支配掉其他所有节点。但是在实现算法过程中，我们发觉这很可能会陷入到局部极小而不能达到实际的最优解。在仔细阅读文献之后，我们发现，Pareto 最优实际上是指相同状态节点的不同路径的完成时间最小的最小路径，而其他 Makespan 更大的路径就被支配了。

5.2 优化尝试

5.2.1 层间删除

在层间删除中，我们需要在 Allnodes 矩阵中找到相同的那些子代，然后判断是否存在被支配的情况。寻找相同的子代时我们用到了 unique 函数，根据返回的结果用了一个三重循环进行支配情况的判断。

如果不进行层间删除，算法也是可行的，因为最终我们都会比较 makespan 从而得到时间最短的路径；只是少了层间删除以后，我们可能会因为少删除了某些节点从而生成了更多的子代。

我们尝试删除层间比较的部分，发现 Allnodes 中的节点数量大大增加，但是总的运行时间变短了。说明进行层间删除所花费的时间代价比生成多余的子代的时间代价更大。

另外，本次作业只需要加工 5 片晶圆，不进行层间删除的时间更短；但是当待加工的晶圆数量增加时，如果不进行层间删除，Allnodes 的规模会更加大，那时候这种优化可能就非常有意义了。

5.2.2 判断旋转

机械臂在每一步动作之后都会旋转到下一步要进行动作的地方，但是刚开始我们想到，如果上一步完成动作的地方就是下一步要进行动作的地方，那么中间的旋转时间其实是可以省略的。一个简单的例子是第一步中，机械臂 1 把一片晶圆从

LLA 搬到 PM11，第二步把晶圆从 PM11 搬到 PM12，在这两步中，机械臂 1 只需要完成装载、旋转、卸载、（等待）、装载，但是我们一开始的程序却是装载、旋转、卸载、（旋转）、装载，多了中间旋转的 1s。

于是我们在计算 Readytime 时多加了一些判断：

- 机械臂 1：如果上一步完成动作的地方就是下一步要进行动作的地方；
- 机械臂 2：如果上一步完成动作后机械臂停留的地方就是下一步双臂之一要进行动作的地方；

那么机械臂就可以节省 1s 旋转的时间。

经过仔细的分析和计算，我们认为机械臂旋转的 1s 与加工时间相比太短了，在我们计算 Readytime 时这些旋转的时间的差异被更长的时间掩盖了，所以这些判断实际上是没有必要的。我们省略了这些判断之后，发现 makespan 并没有变大，这也验证了我们的分析。

最终，这一判断也被我们舍弃，CPU 时间变短了。

5.2.3 生成子代

当我们得到了第一个到达s_{end}的节点之后，便得到了一个 makespan。于是 Allnodes 中任何 max（机械臂 1 时间，机械臂 2 时间）比此 makespan 大的节点都没有必要再继续生成子代。

加入了这一判断之后，CPU 时间变长了，说明判断时间的代价比生成多余的子代的代价更大，于是这一方案被舍弃。

与 5.2.1 相似，本次作业只需要加工 5 片晶圆，不进行是否继续生成子代的判断的时间更短；但是当待加工的晶圆数量增加时，如果不提前删除一些节点，Allnodes 的规模会更加大，那时候这种优化可能就非常有意义了。

5.3 项目收获感想

5.3.1 运筹学习

通过本次大作业，我们对运筹课上所学的知识有了更加深入的了解，也有了实际的体会与应用，对动态规划的方法有了更好的掌握，同时也发现虽然课上听起来

简单容易的方法实际在程序中并不是那么容易实现，尤其是考虑被支配路径时，我们不仅要考虑同一层父代产生的相同子代，也要考虑不同层之间也可能存在相同的状态节点，而这样一来就会造成较大的算法复杂度，因此在实现上我们需要考虑合适合理的数据结构来减少计算量。

5.3.2 编程算法

由于作业要求中提示说“推荐 MATLAB，向量计算较为方便”，因此我们在一开始就使用了 MATLAB 来编写程序。虽然 MATLAB 确实在矩阵计算方面非常简便并且容易查找错误，但是同样它的缺点也是非常显而易见的。

首先，它会出现一些未知的不可控的数据处理过程，比如括号太多或者判断多次嵌套就偶尔会出现一些不知名的错误并且过一会儿就好了；其次，我们发现用动态规划来实现时，我们所产生的节点更加类似于一个树的类型，删除节点，增加节点等操作其实用链表实现会更加方便一些，而用 MATLAB 编程时，我们只能增加变量来存储层数，序号，以及子父代之间的关系，而在判断完 Pateto 最优解后需要“删除”节点，一些节点的序号与层数就会受到影响，必须再进行改变，因此我们只能选择退而求其次，不删除节点，保持序号不变，而不对这个节点生成子代。

5.2.3 文献报告

本次大作业是我们第一次利用已有的文献来复现算法，并且加以改进。在阅读英文文献的过程中，我们不仅了解与拓展了运筹学的相关知识，也学习了如何快速阅读与理解非母语文献，并且在整个算法实现过程中，我们的学术技能，思维能力与报告书写能力都得到了锻炼，感觉是一次非常宝贵的经历。

6 附录

6.1 成员分工

晏筱雯：动态规划思路分析，部分模型建立部分代码编写，部分报告编写；

白一晟：三篇文献分析与综述，部分模型建立，部分代码编写，部分报告编写；
郑　洁：分支定界法思路分析，部分模型建立，部分代码编写，部分报告编写。

6.2 参考文献

[1]Uno Wikborg and Tae-Eog Lee, “Noncyclic Scheduling for Timed DIscrete-Event Systems With Application to Single-Armed Cluster Tools Using Pareto-Optimal Optimization”, IEEE Transactions on Automation Science and Engineering, Vol.10, No.3, page.699-710, 2013

[2]Hyun-Jung Kim, Jun-Ho Lee and Tae-Eog Lee, “Noncyclic Scheduling of Cluster Tools With a Branch and Bound Algorithm”, IEEE Transactions on Automation Science and Engineering, Vol.12, No.2, page.690-700, 2015

[3]C. Jung and T.-E. Lee, “An efficient mixed integer programmingmodel based on timed Petri nets for diverse complex cluster tool scheduling problems”, IEEE Trans. Semicond. Manuf., Vol.25, No.2, page. 186 – 199, 2012