化学发光免疫分析仪运行的调度优化

摘要

化学发光免疫分析仪是一款多功能人体体液样本检测仪器，现已被广泛应用于临床检测，但是检测过程复杂，步骤繁多，且不同类型的生物芯片所需的检测时间也不同，这往往会导致化学发光免疫分析仪中各个检测环节难以连贯，同时得不到充分的利用。针对化学发光免疫分析仪运行调度优化问题的研究是一个具有实际意义的

针对同类芯片调度问题：通过对同类芯片调度问题进行分析，其本质是一个目标规划问题。在化学发光免疫分析仪运行的流程中，进行调度优化的目的是**最小化检测时长**，通过甘特图可以直观地看出，整个调度过程存在入仓顺序、温育盘卡槽限制、磁珠加工冲突、清洗盒与检查盒数目有限等约束条件。基于此，本文建立了**单目标同类芯片调度模型**，并通过 Excel 与 Matlab 进行计算，得到了 A 类芯片单位时间内最多可以检测 54 个芯片。由于 B 类芯片单位时间内无法完成一次检测，故取 2 个小时内的平均单位时间进行计算，得到最多可以检测 21 个芯片。

针对异类芯片调度问题：异类芯片调度优化是一个 NP-Hard 问题。不同于同类芯片的调度，异类芯片之间约束条件更多，建模更为困难。本文将检测流程简化打包成 6 个工序，从柔性作业车间调度问题中得到启发，结合磁珠加工环节带来的非线性过程，以递归思想，建立了**非线性混合流水线异类芯片调度模型。**为了解决模型中的非线性回溯引起的矛盾，提出了一种**冲突避免**机制，并对传统的遗传算法进行改进。

运用**基于冲突避免的改进遗传算法**对模型进行求解，计算得到问题 2 中调度两类芯片的最短检测时间为 8164 秒，问题 3 中调度三类芯片的最短检测时间为 7275 秒，该模型同样可以推广至 N 种芯片进行使用。

针对异类芯片差时调度问题：不同于异类芯片同时调度问题，差时更多的要考虑如何描述在非初始环境下，各芯片的调度优化方案。本文将差时调度问题划分为两个阶段，初阶段是初始环境下，两类芯片的调度问题，后阶段是非初始环境下，三类芯片的调度问题。基于上述建立的**非线性混合流水线异类芯片调度模型，**对其进行改进，加入一个**映射环节**，将时间成本映射于后阶段输入的调整，使非初始环境转变为初始环境，借助改进的遗传算法可以计算得到问题 5 所需要的最短检测时间为 7685 秒。

最后，我们对提出的模型进行了综合评价：本文的模型别出心裁，能够较好地解决提出的问题，具有较强的实用性。此外该模型在智能制造，资源整合方面也能进行推广。

关键词：单目标规划　非线性过程　柔性作业车间调度问题　冲突避免　改进遗传算法

目录

1 问题综述 1
1.1 问题背景 1
1.2 问题提出 1
2 问题分析 2
2.1 总体问题分析 2
2.2 同类芯片调度问题分析 2
2.3 异类芯片调度问题分析 3
2.4 异类芯片差时调度问题分析 3
3 模型假设 3
4 符号说明 4
5 模型建立与求解 5
5.1 同类芯片调度问题 5
5.1.1 数据处理与参数转化 5
5.1.2 同类芯片调度模型建立 6
5.1.3 同类芯片调度模型求解 8
5.2 异类芯片调度问题 9
5.2.1 非线性过程的冲突避免 10
5.2.2 异类芯片调度模型建立 11
5.2.3 异类芯片调度模型求解 12
5.3 异类芯片差时调度问题 14
6 模型的综合评价 15
6.1 模型的优点 15
6.2 模型的不足 15
参考文献 16
附 录 17
附录 1: 17
附录：关键代码 31

1 问题综述

1.1 问题背景

近年来，化学发光免疫测定（CLIA）因其灵敏度高、特异性好、应用范围广、设备简单、线性范围宽等特点，在生命科学、临床诊断、环境监测、食品安全和药物分析等不同领域得到越来越多的关注。进行一次化学发光免疫分析，首先要将人体的体液样本制作成“生物芯片”，而后芯片依次经过进仓读码、激光超声、过滤、定量、R1R2 加样、第一次温育、磁珠加样、第二次温育、清洗以及检测等步骤，而完成一次化学发光免疫测定往往需要花费大量时间与人力。随着社会工业化进程的加速，化学发光免疫测定全过程被整合至全自动化学发光免疫分析仪中，其是进行化学发光免疫测定的专业医学仪器，目前已被广泛应用于各大医院与医疗机构。

图1　全自动化学发光免疫分析仪

随着全自动化学发光免疫分析仪被广泛应用，如何合理运用仪器以提高检测效率成为困扰使用者的一大难题。化学发光免疫分析仪检测步骤繁杂、内含部件丰富，且不同类型的生物芯片所需的检测时间也不同，往往会导致化学发光免疫分析仪中各个检测环节得不到充分的利用。所以如何合理高效地调度化学发光免疫分析仪，安排不同类型生物芯片的检测顺序，是一个具有现实意义且值得深入研究思考的问题。

1.2 问题提出

化学发光免疫分析仪的调度涉及多方面的问题，需要完成规定的检测路径中的六道工序，在保证各检测工序不发生冲突的前提下，使得芯片的检测总的完成时间尽可能小。

需要从解决以下 5 个具体问题:

(1) 问题 1：现有一组 A 类型芯片，其第一次温育时间 10 分钟，第二次温育时间 5 分钟，则单位时间（1 小时）内最多可以完成多少芯片的检测？如果换成一组 B 类型芯片，其第一次温育时间 40 分钟，第二次温育时间 20 分钟的呢？

(2) 问题 2：若有 A，B 两种类型的芯片各 80 片，则它们全部检测完至少需要多少时间？

(3) 问题 3：若还有 C 型芯片，其第一次温育时间 25 分钟，第二次温育时间 10 分钟，而 A 型芯片 60 片、B 型芯片 50 片、C 型芯片 50 片，则全部检测完至少需要多少时间？

(4) 问题 4：针对问题 2 和问题 3，将你的模型和算法，分别推广到芯片数量为m_i，其第一次温育时间为a_i分钟、第二次温育时间为b_i分钟（i=1,2,3）的一般情形。进

一步，可以推广到 N 种类型的芯片吗？并自行举例验证。

(5) 问题 5：若开始有 A 型芯片 60 片、B 型芯片 70 片，运行 30 分钟后，新到 C 型芯片 20 片，问全部检测完至少需要多少时间？

2 问题分析

2.1 总体问题分析

题目以化学发光免疫分析仪的广泛应用为背景，介绍了化学发光免疫分析仪具体的运行过程（为了简化表达，增强阅读的连贯性，全文将“化学发光免疫分析仪”简称为“分析仪”）。通过分析问题，发现五个问题都是围绕如何合理调度生物芯片的检测顺序，使分析仪花费尽可能少的时间检测完所有芯片（或最短的时间检测尽可能多的芯片）。

分析仪运行的调度过程较为复杂，共有十个步骤共六道工序，且每道工序都有其固定的工位，分别为前处理（进仓读码、激光超声、过滤、定量、R1R2 加样）、第一次温育、磁珠加样、第二次温育、清洗和检测。其中前处理的过程是依次串行，温育的过程中温育盘仅有 40 个卡槽，清洗过程中清洗盒仅有 8 个，检测过程中检测盒仅有一个。芯片的检测过程需要按照给定的顺序进行，每一道工序都要完整的完成，且一个芯片一旦开始检测后，就不能停止，在上述约束的情况下，寻找分析仪检测所有芯片最短的调度时间，所以该问题是一个动态规划求最优解的问题。下图将分析仪的整个检测流程进行了更为直观的示意。

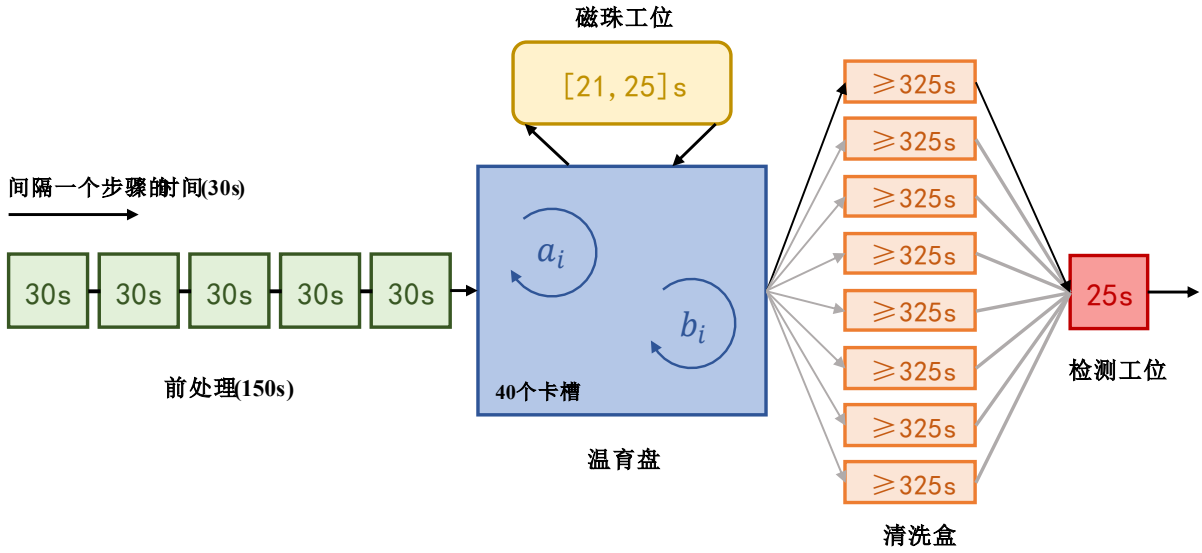

图2 分析仪检测示意图

通过分析提出的 5 个问题，可以将整个问题分成 3 个子问题：相同类型芯片调度、不同类型芯片调度、不同类型芯片间隔不同时间调度。

2.2 同类芯片调度问题分析

对于问题 1，该问分别给出了 A 类型芯片和 B 类型芯片的第一、二次温育的时间，需要求解仅同一类型芯片调度单位时间（1 小时）检测个数的最大值，并填写表格。

对于同一类型的芯片，首先，芯片与芯片之间温育时间不存在差别，即无需考虑芯片入仓的顺序问题，仅需要给出芯片入仓的时间。其次，由于温育盘卡槽、清洗盒、检测盒的数量有限，磁珠工位的加样时间不是一个定值，所以要充分考虑各工序之间的时间冲突，即需要完备用数学语言将限制条件进行转化。最后，需要在上述约束条件下，得到最短时间的调度方案并计算出芯片单位时间内的检测数量、入仓时间、第一次温育

起始时间、磁珠加样起始时间、第二次温育起始时间、清洗起始时间、检测起始时间和检测结束时间。

2.3 异类芯片调度问题分析

对于问题 2、问题 3 和问题 4，其本质都是一样的。通过给定不同芯片的个数，需要求解得到不同类型芯片的入仓顺序与停等时间，并计算全部检测完的最短时间。

对于不同类型的芯片，由于不同芯片的两次温育时间各不相同，相较于同类芯片的调度优化更为复杂。所以本文将整个异类芯片调度问题进行先拆解后组合的方式进行分析。在相同类型芯片的磁珠加样与返回第二次温育的工序中，可以通过简单的线性计算得到是否存在冲突，但是由于不同类型的芯片温育时间不同，所以该过程中夹杂着非线性的回溯。将整个检测过程中前几个工序提取出来，可以得到，如下非线性的检测流程部分。

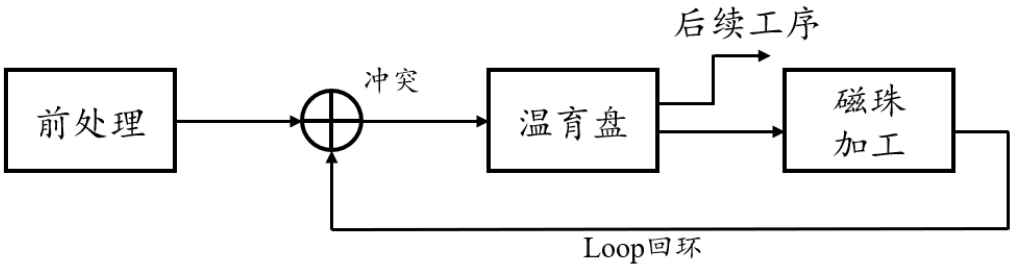

图3 检测过程中的非线性回溯

针对此，需要解决芯片在经过磁珠加工后第二次回到温育盘与第一次进入温育盘的芯片之间的冲突问题。

在不同类型芯片经过第二次温育后，接下来的工序中又存在多个清洗盒、一个检测盒的限制。通过查阅相关文献资料，发现解决后续工序问题，可以与柔性流水线调度的思想相结合，将分析仪中各工序中的卡槽、清洗盒、检测盒等视为加工机器，糅合非线性冲突避免的方案，建立非线性混合流水线异类芯片调度模型。通过基于冲突避免的改进遗传算法进行求解，得到最优的芯片检测顺序与最短的检测时间。

2.4 异类芯片差时调度问题分析

对于问题 5，该问将整个检测过程分为两部分，前 30 分钟仅检测 A、B 两类芯片，30 分钟后可以将 C 型芯片加入进行检测，求解全部检测完成的最短时间。

相较于上述的问题，问题 5 在 30 分钟后丰富了检测芯片种类，即芯片并不是都从初始状态加入，也就是在第 30 分钟时需要对当前进行的调度过程有一个判断，考虑到每一道工序都是连续不可中断的，势必在第 30 分钟时，仍会有芯片处于检测过程中的某一工序。考虑到设计算法的复杂性，为了简化计算，本问基于上述建立的非线性混合流水线异类芯片调度模型进行调整，设置一映射环节，将时间成本映射至输入变量调整，即先得到前 30 分钟 A、B 类型芯片的调度方案后，手动调整运行 30 分钟检测进行中的芯片的加工时长后按照原有优先级顺序加入，然后剩余未检测的 A、B 芯片再与 C 芯片一同按照无时差进行计算，得到最短检测时间。

3 模型假设

(1) 假设转盘或检测盘在每次的转动对位过程中，无论如何转动、移动多少距离，所需时长是一样的，且忽略同时转动对位过程中的时间冲突，即轮盘转动对位至磁珠工位的同时可以进行对位至清洗盒；

(2) 在整个检测过程中，各个检测环节不会积压芯片且不会出现故障；
(3) 忽略温育盘中卡槽的位置差异；
(4) 假设同一环节检测不同类型芯片不存在准备时间，即可以无缝衔接；
(5) 假设转盘对位的过程中无视物体碰撞冲突，例如某一芯片从温育盘进入磁珠工位的同时另一芯片可以从磁珠工位回到温育盘。

4 符号说明

本文为了便于使用数学语言解决化学发光免疫分析仪运行调度问题，故给出了部分使用次数较多的符号的含义，其余使用次数较少的符号在使用时进行注明。

表1 符号说明

符号	含义
$Tw_{1st}s_i$	第一次温育开始时间
$Tw_{1st}f_i$	第一次温育结束时间
Tcs_i	磁珠加工开始时间
Tcf_i	磁珠加工结束时间
$Tw_{2nd}s_i$	第二次温育开始时间
$Tw_{2nd}f_i$	第二次温育结束时间
Tqs_i	清洗工序开始时间
Tqf_i	清洗工序结束时间
Tjs_i	检测工序开始时间
Tjf_i	检测工序结束时间
T_z	检测花费的总时间
T_{jg}	相邻芯片入仓之间的间隔时间
T_{zl}	因冲突导致芯片 m 入仓的滞留时间
Tq_{zl}	芯片在清洗工序中滞留的时间
n	需要进行检测的芯片总数
c	每个芯片历经的工序数
m_k	第 k 道工序并行的机器数量
$p(k,j)\ (k\in c, j\in n)$	芯片 j 第 k 道工序有需要的检测时间
$i_k\ (i\in m)$	第 k 道工序的第 i 个机器
$T(i_k,j)$	芯片 j 第 k 道工序在机器 i 上完成检测的时间

5 模型建立与求解

5.1 同类芯片调度问题

同类芯片调度主要是解决问题 1，在该问题中不考虑芯片的种类，仅需计算不发生冲突的情况下，单位时间最多可以完成的芯片检测数目。

为了更为直观的感受，分析仪的工作过程中存在的冲突，以及需要进行限制的各项条件，利用 Matlab 绘制了 A 类芯片按最短时间间隔顺次进入仓内检测，且为了尽可能缩短时间，下图将磁珠加工时间设为 21s。

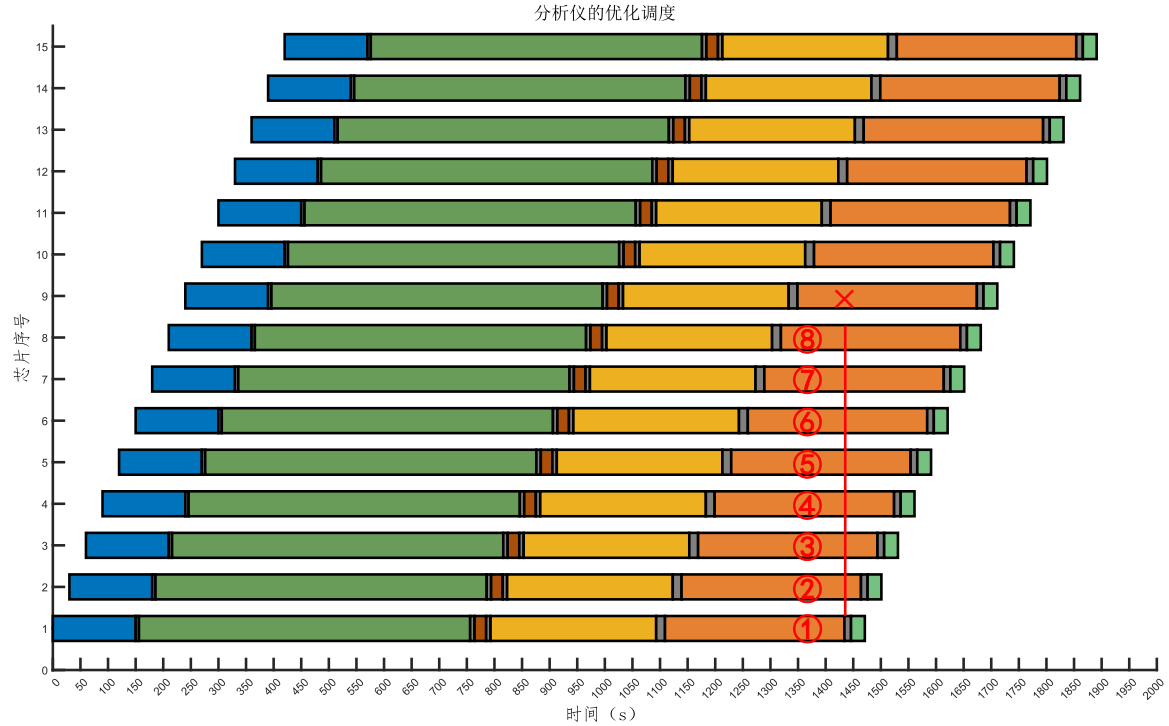

图4 清洗工序发生冲突情况甘特图

如图所示，可以看到如果不加以调度，仅按照间隔 30s 入仓检测的话，仅在第九个芯片进入后就会发生冲突。此外，除了清洗工序外，还需要考虑温育盘的卡槽数量，磁珠工位加样以及检测工序，由此可以分析得到约束得到最短检测时间的一些限制条件。

5.1.1 数据处理与参数转化

在具体进行建模之前，本文先将题目中的一些表达进行转化，并且对数据进行整合，这样更便于后续的问题解决与读者阅读。

表2 不同类型芯片各工序时刻表

工序	第一次温育开始时间	第一次温育结束时间	磁珠加工开始时间	磁珠加工结束时间	第二次温育开始时间
符号	$Tw_{1st}s_i$	$Tw_{1st}f_i$	Tcs_i	Tcf_i	$Tw_{2nd}s_i$
A	156	756	764	785	793
B		2556	2564	2572	2580
C		1656	1664	1685	1693

工序	第二次温育结束时间	清洗工序开始时间	清洗工序结束时间	检测工序开始时间	检测工序结束时间
符号	$Tw_{2nd}f_i$	Tqs_i	Tqf_i	Tjs_i	Tjf_i
A	1093	1109	≥1434	≥1446	≥1471
B	3780	3796	≥4121	≥4133	≥4158
C	2293	2309	≥2634	≥2646	≥2671

由于本问仅仅针对同类型的芯片，故本文将分析仪整个的检测过程中存在的各个工序所花费时间进行了整合，便于后续进行建模与计算。由于是同类型芯片，故不存在芯片顺序的差异。

值得注意的是，题目中磁珠加工时间并不是一个定值而是一段范围，但是由于本模型主要解决的是同类型芯片的调度问题，通过简单计算可以发现，由于顺序入仓的芯片之间已经存在了间隔时间T_{jg}，且间隔时间远大于磁珠加工时间的范围，所以不会存在因取不同的加工时间导致冲突的情况，所以为了缩短检测时间，在同类芯片调度模型中磁珠加工时间都取 21s。

5.1.2 同类芯片调度模型建立

通过上述的问题分析，本问需要建立一个满足诸多限制条件下，检测花费时间最短的同类型芯片的最优调度模型。由于优化目标仅为检测花费时间，所以可以将本问题建模为单目标规划，下面将给出目标函数以及各约束条件的表达式。

（1）目标函数的确定

为了寻找分析仪检测花费的最短时间，将给出检测花费时间的组成部分。由于本问仅考虑使用同一类型的芯片，且检测一旦开始后便不能停止，所以检测总花费的时间应由最后一个芯片花费的时间决定。

设共有同类型芯片 m 个，则检测花费的总时间T_z可以由下式表示

$$T_z=(m-1)T_{jg}+T_{zl}+Tjf_m+Tq_{zl}$$

式中T_{jg}表示相邻芯片入仓之间的间隔时间，T_{zl}表示因冲突导致芯片 m 入仓的滞留时间，Tjf_m表示第 m 个芯片在检测工序结束时的时间，Tq_{zl}表示芯片在清洗工序中滞留的时间。

T_{zl}与Tq_{zl}可有下式表示

$$T_{zl}=Tqs_{i+m}-Tqs_i-m\cdot T_{jg}$$

$$Tq_{zl}=Tqf_{i+m}-Tqs_{i+m}-325$$

（2）约束条件

由于同类型芯片顺序排列，其两次温育的时间是相同的，所以主要考虑的约束条件有进仓顺序、温育盘卡槽数目、磁珠加工时间冲突、清洗盒数目和检测盒数目。

1）进仓顺序约束

由于前处理的过程中仅有一个工作区，所以需要对进入前处理的芯片进行排布，令入仓的芯片满足题目要求

$$Tw_{1st}s_{i+1}\geq Tw_{1st}s_i+T_{jg}$$

即第$i+1$个芯片的入仓时间要比第 i 个芯片的入仓时间多至少一个间隔时间T_{jg}。

2）温育盘卡槽约束

温育盘的卡槽数目为 40 个，即同一时刻无论是第一次温育还是第二次温育最多有 40 个芯片可以同时留在卡槽中，所以要约束第$i+40$个芯片的第一次温育开始时间要比第 i 个芯片的第一次温育结束时间多至少 40 个间隔时间T_{jg}

$$Tw_{1st}s_{i+40} \geq Tw_{1st}f_i + 40 \cdot T_{jg}$$

此外为了保证从磁珠工位出来的芯片能够顺利返回温育盘，所以需要满足以下两个条件中的任意一个：一是第$i+1$个芯片的第一次温育结束时间小于等于第i个芯片的第二次温育开始时间，即

$$Tw_{1st}f_{i+1} \leq Tw_{2nd}s_i$$

二是第$i+40$个芯片第一次温育开始时间要大于第i个芯片的第二次温育开始时间，即

$$Tw_{1st}s_{i+40} \geq Tw_{2nd}s_i$$

若以上两个条件如果同时满足，第一个条件的优先级高于第二个条件，即可以对第二个约束条件进行优化。如果满足第一个条件，则意味着前一个芯片从磁珠工位出来的同时，后一个芯片会随即进入磁珠工位，所以第$i+40$个芯片可以进入温育盘开始第一次温育，即温育盘与磁珠工位之间最多共有 41 个芯片且保持着动态平衡。但是又会有新的约束，即，第$i+41$个芯片第一次温育开始时间要大于等于第i个芯片的第二次温育结束时间，即

$$Tw_{1st}s_{i+41} \geq Tw_{2nd}f_i$$

3）磁珠加工空间约束

磁珠加工是分析仪检测过程中一个回溯过程，即芯片从温育盘进入磁珠工位后还需要返回至温育盘，又因为磁珠工位仅一个，所以第$i+1$个芯片的磁珠加工开始时间要大于第 i 个芯片的磁珠加工结束时间。

$$Tcs_{i+1} \geq Tcf_i$$

4）清洗盒数目约束

清洗盒的数量为 8 个，所以清洗盒数目的约束条件为第$i+8$个芯片的清洗工序开始时间要大于第 i 个芯片的清洗工序结束时间

$$Tqs_{i+8} \geq Tqf_i$$

5）检测盒数目约束

分析仪检测过程中最后一道工序为检测工序，检测盒仅有一个却无法同时进行检测，又由于同类型芯片都是顺序入仓，所以类比于上述约束条件，检测盒数目的约束条件为第$i+1$个芯片的检测工序开始时间要大于第 i 个芯片检测工序结束时间。

$$Tjs_{i+1} \geq Tjf_i$$

（3）同类芯片调度模型

综上所述，本文围绕同类芯片调度建立了单目标规划模型，完整优化模型可有下式表示

$$\min \quad T_z$$

$$s.t.\begin{cases} Tw_{1st}s_{i+1} \geq Tw_{1st}s_i + T_{jg} \\ Tw_{1st}s_{i+40} \geq Tw_{1st}f_i + 40 \cdot T_{jg} \\ Tw_{1st}s_{i+40} \geq Tw_{2nd}s_i \\ Tcs_{i+1} \geq Tcf_i \\ Tqs_{i+8} \geq Tqf_i \\ Tjs_{i+1} \geq Tjf_i \end{cases}$$

或者

$$\min \quad T_z$$

$$s.t.\begin{cases} Tw_{1st}s_{i+1} \geq Tw_{1st}s_i + T_{jg} \\ Tw_{1st}s_{i+40} \geq Tw_{1st}f_i + 40 \cdot T_{jg} \\ Tw_{1st}f_{i+1} \leq Tw_{2nd}s_i \\ Tw_{1st}s_{i+41} \geq Tw_{2nd}f_i \\ Tcs_{i+1} \geq Tcf_i \\ Tqs_{i+8} \geq Tqf_i \\ Tjs_{i+1} \geq Tjf_i \end{cases}$$

5.1.3 同类芯片调度模型求解

该模型由于仅为同类芯片的调度，故计算量不大，较为简单，关于具体的参数数据，已经在先前给出说明。

（1）A 类型芯片的调度

利用 Excel 与 Matlab 对上述模型进行求解，并画出部分甘特图，具体的计算结果详见附录。单位时间内 A 芯片最多可以完成 54 个芯片的检测。

表3 A 类型芯片调度结果（部分）

芯片序号	进仓时间	第一次温育起始时间	磁珠加样起始时间	第二次温育起始时间	清洁起始时间	检测起始时间	检测结束时间
1	0	156	764	793	1109	1446	1471
2	30	186	794	823	1139	1476	1501
3	60	216	824	853	1169	1506	1531
4	90	246	854	883	1199	1536	1561
				⋮			
53	2070	2226	2834	2863	3179	3516	3541
54	2100	2256	2864	2893	3209	3546	3571
55	2130	2286	2894	2923	3239	3576	3601

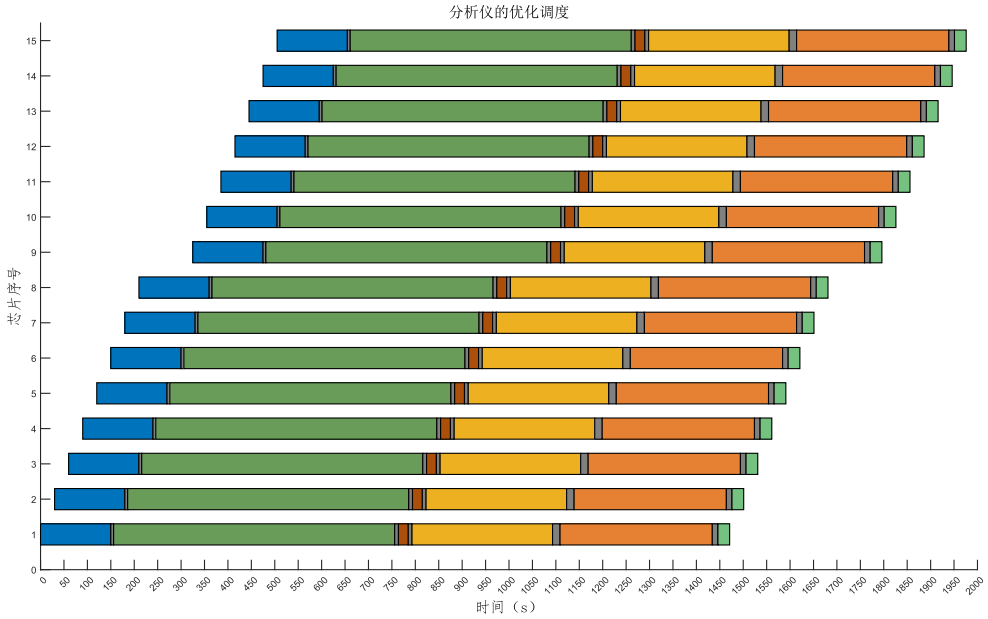

图5 A 类芯片调度甘特图（部分）

（2）B 类型芯片的调度

由于 B 类芯片的第一次温育时间为 40 分钟，第二次温育时间为 20 分钟，仅两次温育的时间就已经超过题目给出的单位时间（一个小时），所以本文将单位时间理解为平均单位时间，共计算了 2 个小时内 B 类芯片最多可以完成的检测数量，平均后得到了单位时间内 B 芯片最多可以完成 21 个芯片的检测。

芯片序号	进仓时间	第一次温育起始时间	磁珠加样起始时间	第二次温育起始时间	清洁起始时间	检测起始时间	检测结束时间
1	0	156	2564	2593	3809	4146	4171
2	30	186	2594	2623	3839	4176	4201
3	60	216	2624	2653	3869	4206	4231
4	90	246	2654	2683	3899	4236	4261
				•			
				•			
				•			
40	1510	1666	4074	4103	5319	5656	5681
41	2400	2556	4964	4993	6209	6546	6571
42	3637	3793	6201	6230	7446	7783	7808

5.2 异类芯片调度问题

异类芯片调度是为了解决问题 2、问题 3 和问题 4，整体来看这三个问题的本质是相同的，都是通过优化不同类型芯片的检测顺序排列，从而实现整体检测的时间最短。如果继续沿用解决问题 1 的思想，采用单目标规划的思想去建立模型，有以下几点较为棘手的问题不好解决：

1.芯片数量多，种类丰富，且不同芯片的两次温育时间不同，建模难度大；

2.分析仪检测步骤复杂，包含诸如出入温育盘、磁珠加工时间不固定等，约束条件描述不清；

3.求解过程中各参数相互制约，对于代码编程要求高。

故此，我们经过广泛的调研与阅读，充分发散思维，进行头脑风暴，将问题简化整合。从另一个角度看，如果将整个分析仪的检测过程进行划分，到第二次温育开始之前，可以视为单流水线至并机调度问题，在参考了相关文献后，我们决定可以采取柔性作业车间调度的思想作为模型的基本框架[1]，综合考虑磁珠加工过程返回时的冲突，建立非线性混合流水线异类芯片调度模型。

5.2.1 非线性过程的冲突避免

为了顺利建立异类芯片调度模型，首先要解决的是因为磁珠加工而导致的芯片反复进出温育盘导致的非线性的问题。对于非线性流水线，在并行计算机体系结构中比较常见，为了提高处理机的运算速度，常用非线性的前馈与反馈连接，以增加少量的硬件就能够把计算机的处理速度提升几倍[2]。在计算机中一条非线性流水线往往采用流水线的连接图与预约表一同表示，文献[3]利用禁止等待时间来限制冲突的发生，并找到最小的平均等待时间的启动循环，来保证流水线在所有功能段任何周期都不会发生冲突。

受到计算机体系结构中解决非线性问题的启发，我们给出了在非线性过程中解决冲突的方案。在本问题中，我们可以通过设计判断过程来规避芯片返回温育盘时引起的冲突。在第 i 个芯片进入磁珠加工的瞬间，将当前温育盘各卡槽芯片的状态记录下来，我们将磁珠加工中的芯片的优先级置为最高，设置了几个判断问题，如果有肯定的回答，即不会发生冲突，具体判断语句如下所示：

1.当前可用卡槽数目是否大于 2；

2.当前温育盘中芯片是否有 21-25s 后进入磁珠加工或清洗工序；

3.21-25s 后是否会有新的芯片进入温育盒。

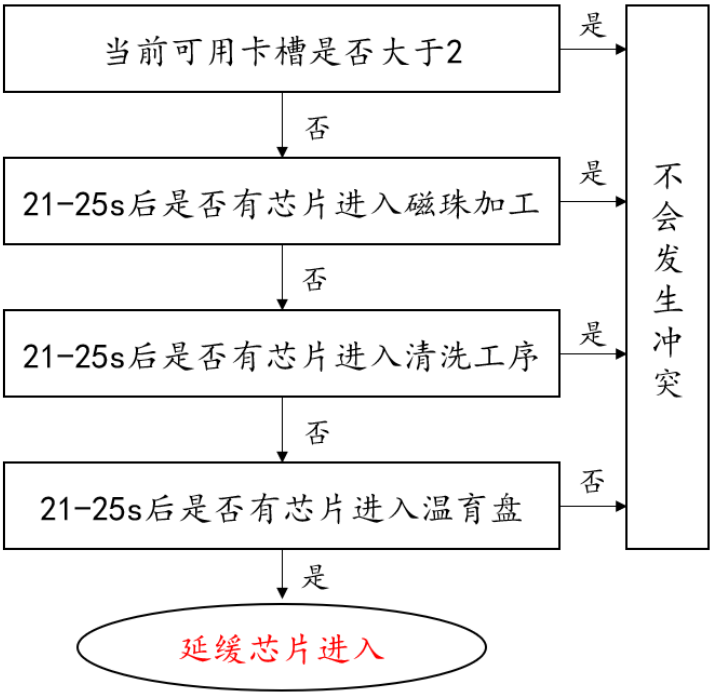

图6 判断是否发生冲突过程图

如果上述判断语句条件都不满足，可以采取延缓芯片进入温育盘的方式，留出空拍。在具体实现的过程中可以使用表格储存当前温育盘中的信息，并进行计算，将会在后续模型求解的过程中重点讲述。

5.2.2 异类芯片调度模型建立

对于异类芯片调度，芯片经过单行流水线（前处理）与并行机（温育盘卡槽、清洗盒）的处理，又涵盖着非线性反馈回溯（磁珠加工），对异类芯片的调度运行是一个复杂的优化模型。不过根据前期的调研与文献阅读，我们认为可以将柔性作业车间调度问题的思想作为模型建立的框架。

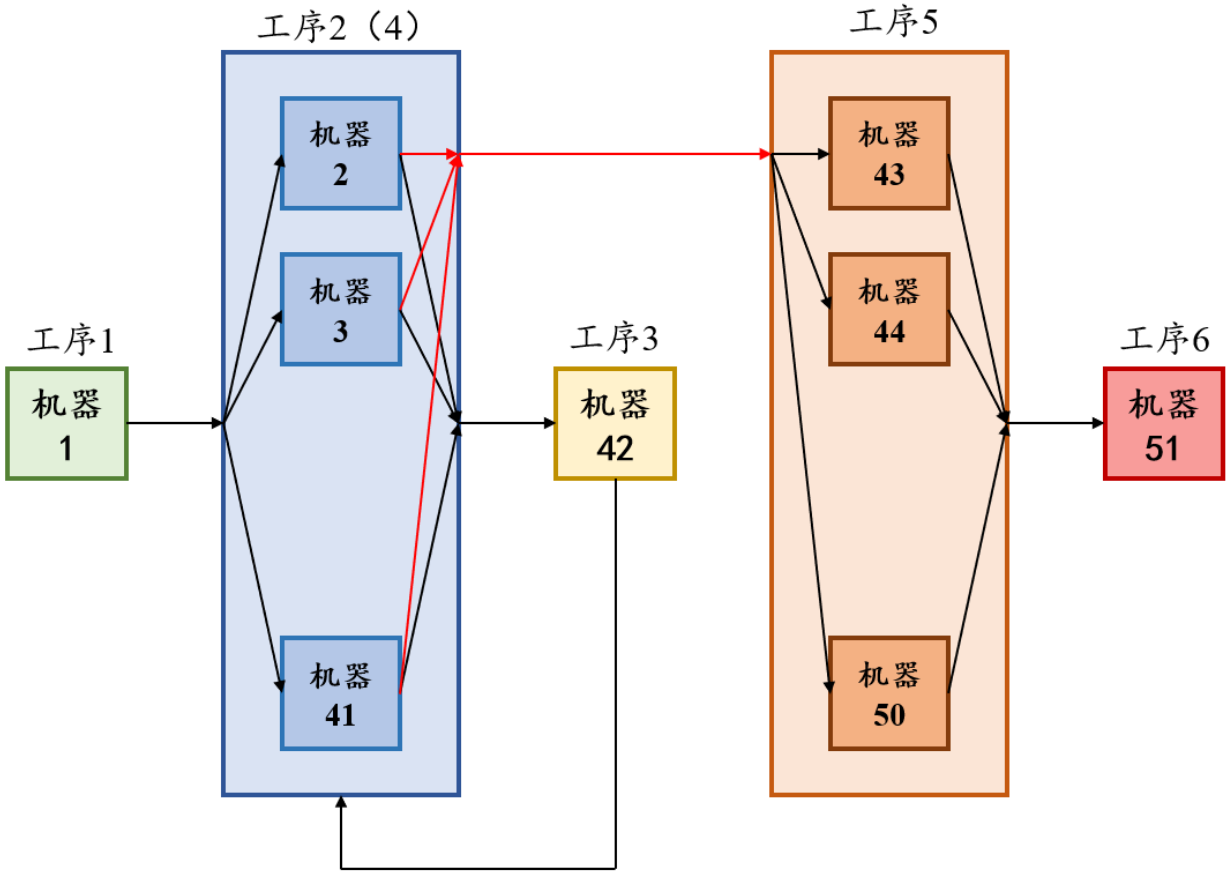

图7 重新命名的流程示意图

首先，将整个检测流程进行分割，把轮盘转动的时间与工序时间相整合，组成新的工序，并且将原工序中的卡槽、清洗盒等视为新工序中的机器，则本文研究的分析仪芯片检测过程有以下特点：

（1）同一工序中的所有机器都相同；

（2）每个芯片都可以在某一工序的任何机器上进行加工；

（3）任意时刻每个芯片至多在一个机器上进行检测；

（4）每个机器在某一时刻只能检测一个芯片；

（5）芯片的检测过程不允许中断。

根据以上特点，我们发现异类芯片的最优调度问题高度契合柔性作业车间调度，此外，分析仪的检测流程中存在一个非线性的过程，也就是因为磁珠加工的原因，致使工序 2 与工序 4 之间共用相同的机器，不过上文已经得到了非线性过程冲突避免的方法。

基于此，我们建立了非线性混合流水线异类芯片调度模型，来寻找时间花费最少的异类芯片检测调度方案。

对于工序 1，可以随机选择芯片在空闲的机器上依次进行检测，进行工序 2 时，按照先完成的先检测的原则，在空闲机器上优先检测当前可用芯片（该芯片在上一工序已经完成检测），以此类推，直到最后一个芯片在最后一道工序完成检测。

表4 补充符号解释

符号	解释
n	需要进行检测的芯片总数
c	每个芯片历经的工序数
m_k	第 k 道工序并行的机器数量
$p(k,j)\ (k \in c, j \in n)$	芯片 j 第 k 道工序有需要的检测时间
$i_k\ (i \in m)$	第 k 道工序的第 i 个机器
$T(i_k, j)$	芯片 j 第 k 道工序在机器 i 上完成检测的时间

用一组递推公式可以表示为：

$$T(i_1,1) = p(1,1)$$

表示第 1 个芯片在第一道工序的时间等于该芯片在第一阶段完成检测的时间；

$$T(i_k,1) = T(i_{k-1},1) + p(k,1)$$

表示第 1 个芯片第 k 道工序完成检测的时间等于该芯片紧前工序的完成检测时间加上当前工序的检测时间；

$$T(i_1, j) = T(i_1, j-1) + p(1, j)$$

表示芯片 j 第 1 道工序完成检测的时间等于同一机器上紧前芯片第 1 道工序的完成检测时间加上芯片 j 第 1 道工序的检测时间；

$$T(i_k, j) = \max\{T(i_{k-1}, j), T(i_k, j-1)\} + p(k, j)$$

表示芯片 j 在工序 k 的完工时间，等于芯片 j 紧前工序的完成检测时间或同一机器紧前工件 j-1 的完成检测时间中的最大值加上工件 j 在阶段 k 的检测时间。

因此，进而可以得到该模型的目标函数：

$$\min\{T(m_c, n)\}$$

5.2.3 异类芯片调度模型求解

根据上述所建立的非线性混合流水线异类芯片调度模型，其复杂程度较高，利用传统算法求解，编程难度较大。针对类似 NP-hard 问题，一类基于生物进化或自然现象的启发式算法得到了极大的发展，这类算法求解效率高、对目标函数和约束条件适应性强因此广受青睐。遗传算法作为最早用于求解优化调度的算法，其具有较强的全局搜索能力、通用性强且具有隐含并行性等优点，本文将结合具体问题与建立的非线性判断准则对传统的遗传算法进行改进，使其能够用于求解上述建立的模型。

5.2.3.1 编码

对于遗传算法，其核心是交叉和变异过程，但其实编码也同等重要。针对本问题，先将全部芯片按照 1~n 的顺序编号，我们采取的编码方式是基于芯片类型的随机全排列方法。编码代表了芯片被处理的优先级，因为该问题仅有 A、B、C 三种类型的芯片，为了减少计算量，便于优化，我们首先将 A、B、C 分区，在区内随机排列。

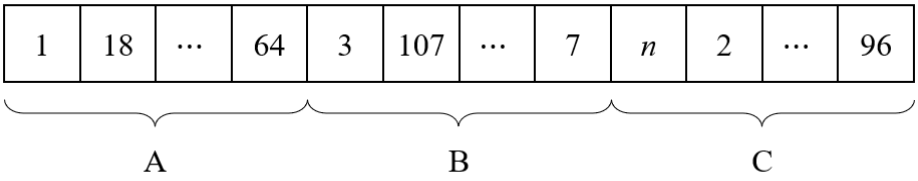

图8 基于芯片类型的随机初始编码示意图

5.2.3.2 适应度函数

适应度函数作为评价个体优劣的标准，适应度越高，则个体越接近最优解，适应度越低，则个体解越差。为了简化计算，我们就采用目标函数的倒数作为适应度函数来进行计算。

$$F_i = \frac{1}{\min\{T(m_c, n)\}},\ i = 1, 2, \cdots, M$$

其中 i 代表个体，M 为种群的总数。

5.2.3.3 选择操作

选择操作是第一轮初筛，遵循优胜劣汰原则，将劣质个体淘汰，替换成优秀的个体，初步优化种群。本文在该步骤采用轮盘赌的方式重新选择个体，遗传概率可有下式表示

$$P_i = \frac{F_i}{\sum_{i=1}^{M} F_i},\ i = 1, 2, \cdots, M$$

累计遗传概率为

$$q_i = \sum_{j=1}^{i} P_j$$

5.2.3.4 交叉操作

种群中的个体随机进行两两配对，配对成功的两个个体作为父代 1 和父代 2 进行交叉操作。随机生成两个不同的基因点位，子代 1 继承父代 2 交叉点位之间的基因片段，其余基因按顺序继承父代 1 中未重复的基因；子代 2 继承父代 1 交叉点位之间的基因片段，其余基因按顺序继承父代 2 中未重复的基因。

5.2.3.5 变异操作

采用两点变异的方式，随机生成两个基因位，并交换两个基因位上的基因。

5.2.3.6 冲突避免判断

本文在设计遗传算法的过程中在个体进入判断终止条件之前，先进该调度方案是否会引发冲突的判断。在设计算法时，为了便于编写程序，我们将非线性过程进行拉平，即工序 2 与工序 4 共用 40 个机器。

基于上述的冲突避免判断机制，在个体经过变异后对个体的顺序对进行冲突避免判断，如果不满足则返回改进适应度重新进行计算，如果满足则可以进行后续的条件判断。

整个算法的具体流程如下图所示。

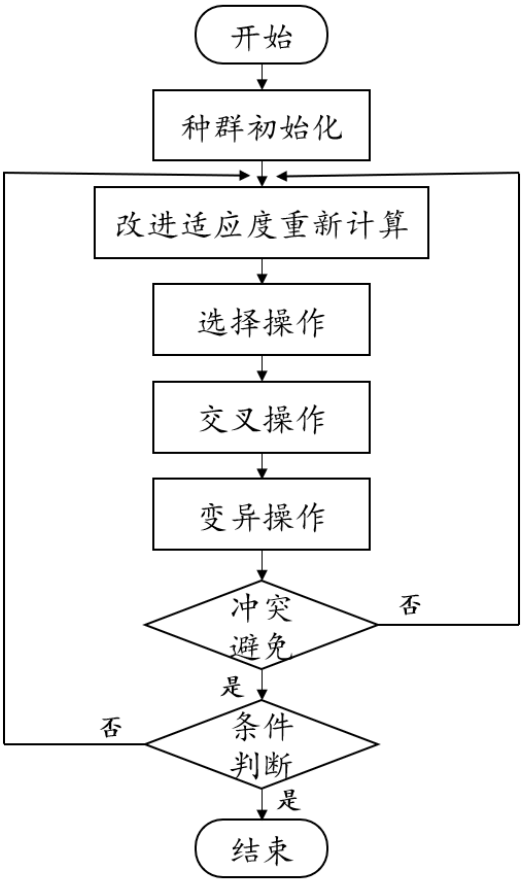

图9 改进遗传算法的流程图

（1）问题 2 的求解结果

A，B 类芯片各 80 片，全部检测完所需要的最短时间为 8164s。具体的排列顺序以及各时间点的结果详见附录。

（2）问题 3 的求解结果

A 类芯片 60 片、B 类芯片 50 片、C 类芯片 50 片全部检测完所需要的最短时间为 7275s。具体的排列顺序以及时间点的结果详见附录。

（3）问题 4 的求解结果

当芯片数量为 m_i，即模型中芯片数量可以表示为

$$n' = m_1 + m_2 + m_3$$

然后将上述模型中的 $p(k, j)$ 根据给出的 a_i 与 b_i 的具体值进行调整即可。

如果推广为 N 种类型的芯片的话，需要对 n 进行更新

$$n'' = m_1 + m_2 + m_3 + \ldots + m_N$$

5.3 异类芯片差时调度问题

异类芯片差时调度是为了解决问题 5，不同于其他问题，其他问题都是给出检测芯片的个数，求解检测过程中用得最短时间，而问题 5 是在前 30 分钟仅允许 A 与 B 类型的芯片，30 分钟之后再加入 C 类型芯片。

经过分析，我们认为这个问题应该分成两个阶段进行求解，分别为 30 分钟前阶段与 30 分钟后阶段。

（1）30 分钟前阶段

前 30 分钟的异类芯片调度策略应与上述建立的解决问题 2 的模型与求解相一致，所以可以直接由问题 2 得到芯片调度方案。

（2）30 分钟后阶段

后 30 分钟的异类芯片调度与问题 3 的不同之处在于，问题 3 是同时进行，初始环境都是空白的，而问题 5 的后阶段虽然也是 3 类芯片进行调度，但是前一阶段仍有开始检测但未完成检测的芯片在分析仪上面，所以要解决的是如何刻画 30 分钟时分析仪的运行情况。

综上，本文基于上述建立的非线性混合流水线异类芯片调度模型上进行改动，增加了一个映射环节，将 30 分钟时的运行环节进行描述。由于前 30 分钟仅有 A 与 B 的调度方案是已知的，所以还是遵循两阶段求解，第一步先得到 30 分钟时运行情况，然后将当前的情况记录下来；第二步将记录下来的运行情况进行复刻，即可以手动调整 30 分钟时的输入，给予当时还在进行检测的芯片更高的优先级，更改输入来复现 30 分钟时的运行情况，例如某芯片已经在工序 5 进行了 5s，可以将调整该芯片的前四道工序置为 0，再将工序 5 所需时间减少 5s。

经过计算得到问题 5 的结果为 7685 秒。

6 模型的综合评价

6.1 模型的优点

(1) 模型将柔性作业车间的思想与非线性流水线相结合，模型想法独特，创新性强，通过运算也具有较强的实用性。

(2) 本文使用的遗传算法是一种启发式算法，对于求解类似优化问题比较适用，此外本文在算法原有的基础之上，又进行了改进，增加了冲突避免机制，更具有应用的广泛性。

6.2 模型的不足

(1) 模型在对于轮盘转动的分析解决上过于粗暴，对于一些细节处理没有很细腻，求解的遗传算法容易陷入局部最优解。

(2) 本文提出的模型仅针对当前场景，由于算法主要参考的是柔性作业车间和递归的思想，其更适用于小数据集。

参考文献

[1] 周鹏鹏，翟志波，戴玉森.基于改进遗传算法的柔性作业车间调度问题研究[J].组合机床与自动化加工技术,2023,(3): 183-186，192

[2] 郑芹.非线性流水线的无冲突调度的分析[J].福建电脑,2005,第 21 卷(3): 43-44

[3] 师军.非线性流水线计算机调度问题的研究[J].电脑开发与应用,1995,第 8 卷(4): 39-43

附　录

附录 1:

A 型芯片的调度结果

芯片序号	进仓时间	第一次温育起始时间	磁珠加样起始时间	第二次温育起始时间	清洗起始时间	检测起始时间	检测结束时间
1	0	156	764	793	1109	1446	1471
2	30	186	794	823	1139	1476	1501
3	60	216	824	853	1169	1506	1531
4	90	246	854	883	1199	1536	1561
5	120	276	884	913	1229	1566	1591
6	150	306	914	943	1259	1596	1621
7	180	336	944	973	1289	1626	1651
8	210	366	974	1003	1319	1656	1681
9	325	481	1089	1118	1434	1771	1796
10	355	511	1119	1148	1464	1801	1826
11	385	541	1149	1178	1494	1831	1856
12	415	571	1179	1208	1524	1861	1886
13	445	601	1209	1238	1554	1891	1916
14	475	631	1239	1268	1584	1921	1946
15	505	661	1269	1298	1614	1951	1976
16	535	691	1299	1328	1644	1981	2006
17	650	806	1414	1443	1759	2096	2121
18	680	836	1444	1473	1789	2126	2151
19	710	866	1474	1503	1819	2156	2181
20	740	896	1504	1533	1849	2186	2211
21	770	926	1534	1563	1879	2216	2241
22	800	956	1564	1593	1909	2246	2271
23	830	986	1594	1623	1939	2276	2301
24	860	1016	1624	1653	1969	2306	2331
25	975	1131	1739	1768	2084	2421	2446
26	1005	1161	1769	1798	2114	2451	2476
27	1035	1191	1799	1828	2144	2481	2506

28	1065	1221	1829	1858	2174	2511	2536
29	1095	1251	1859	1888	2204	2541	2566
30	1125	1281	1889	1918	2234	2571	2596
31	1155	1311	1919	1948	2264	2601	2626
32	1185	1341	1949	1978	2294	2631	2656
33	1300	1456	2064	2093	2409	2746	2771
34	1330	1486	2094	2123	2439	2776	2801
35	1360	1516	2124	2153	2469	2806	2831
36	1390	1546	2154	2183	2499	2836	2861
37	1420	1576	2184	2213	2529	2866	2891
38	1450	1606	2214	2243	2559	2896	2921
39	1480	1636	2244	2273	2589	2926	2951
40	1510	1666	2274	2303	2619	2956	2981
41	1625	1781	2389	2418	2734	3071	3096
42	1655	1811	2419	2448	2764	3101	3126
43	1685	1841	2449	2478	2794	3131	3156
44	1715	1871	2479	2508	2824	3161	3186
45	1745	1901	2509	2538	2854	3191	3216
46	1775	1931	2539	2568	2884	3221	3246
47	1805	1961	2569	2598	2914	3251	3276
48	1835	1991	2599	2628	2944	3281	3306
49	1950	2106	2714	2743	3059	3396	3421
50	1980	2136	2744	2773	3089	3426	3451
51	2010	2166	2774	2803	3119	3456	3481
52	2040	2196	2804	2833	3149	3486	3511
53	2070	2226	2834	2863	3179	3516	3541
54	2100	2256	2864	2893	3209	3546	3571
55	2130	2286	2894	2923	3239	3576	3601

附录 2:

B 型芯片的调度结果

芯片序号	进仓时间	第一次温育起始时间	磁珠加样起始时间	第二次温育起始时间	清洗起始时间	检测起始时间	检测结束时间
1	0	156	2564	2593	3809	4146	4171
2	30	186	2594	2623	3839	4176	4201
3	60	216	2624	2653	3869	4206	4231
4	90	246	2654	2683	3899	4236	4261
5	120	276	2684	2713	3929	4266	4291
6	150	306	2714	2743	3959	4296	4321
7	180	336	2744	2773	3989	4326	4351
8	210	366	2774	2803	4019	4356	4381
9	325	481	2889	2918	4134	4471	4496
10	355	511	2919	2948	4164	4501	4526
11	385	541	2949	2978	4194	4531	4556
12	415	571	2979	3008	4224	4561	4586
13	445	601	3009	3038	4254	4591	4616
14	475	631	3039	3068	4284	4621	4646
15	505	661	3069	3098	4314	4651	4676
16	535	691	3099	3128	4344	4681	4706
17	650	806	3214	3243	4459	4796	4821
18	680	836	3244	3273	4489	4826	4851
19	710	866	3274	3303	4519	4856	4881
20	740	896	3304	3333	4549	4886	4911
21	770	926	3334	3363	4579	4916	4941
22	800	956	3364	3393	4609	4946	4971
23	830	986	3394	3423	4639	4976	5001
24	860	1016	3424	3453	4669	5006	5031
25	975	1131	3539	3568	4784	5121	5146
26	1005	1161	3569	3598	4814	5151	5176
27	1035	1191	3599	3628	4844	5181	5206
28	1065	1221	3629	3658	4874	5211	5236
29	1095	1251	3659	3688	4904	5241	5266

30	1125	1281	3689	3718	4934	5271	5296
31	1155	1311	3719	3748	4964	5301	5326
32	1185	1341	3749	3778	4994	5331	5356
33	1300	1456	3864	3893	5109	5446	5471
34	1330	1486	3894	3923	5139	5476	5501
35	1360	1516	3924	3953	5169	5506	5531
36	1390	1546	3954	3983	5199	5536	5561
37	1420	1576	3984	4013	5229	5566	5591
38	1450	1606	4014	4043	5259	5596	5621
39	1480	1636	4044	4073	5289	5626	5651
40	1510	1666	4074	4103	5319	5656	5681
41	2400	2556	4964	4993	6209	6546	6571
42	3637	3793	6201	6230	7446	7783	7808

附录 3:

A、B 型芯片的调度结果

芯片序号	芯片类型	进仓时间	第一次温育起始时间	磁珠加样起始时间	第二次温育起始时间	清洗起始时间	检测起始时间	检测结束时间
1	B	0	156	2564	2593	3809	4146	4171
2	B	30	186	2594	2623	3839	4176	4201
3	B	60	216	2624	2653	3869	4206	4231
4	B	90	246	2654	2683	3899	4236	4261
5	B	120	276	2684	2713	3929	4266	4291
6	B	150	306	2714	2743	3959	4296	4321
7	B	180	336	2744	2773	3989	4326	4351
8	B	210	366	2774	2803	4019	4356	4381
9	A	240	396	1004	1033	1349	1686	1711
10	A	270	426	1034	1063	1379	1716	1741
11	A	300	456	1064	1093	1409	1746	1771
12	B	330	486	2894	2923	4139	4476	4501
13	B	360	516	2924	2953	4169	4506	4531
14	B	390	546	2954	2983	4199	4536	4561
15	B	420	576	2984	3013	4229	4566	4591
16	B	450	606	3014	3043	4259	4596	4621
17	B	480	636	3044	3073	4289	4626	4651
18	B	510	666	3074	3103	4319	4656	4681
19	B	540	696	3104	3133	4349	4686	4711
20	A	570	726	1334	1363	1679	2016	2041
21	A	600	756	1364	1393	1709	2046	2071
22	A	630	786	1394	1423	1739	2076	2101
23	B	660	816	3224	3253	4469	4806	4831
24	B	690	846	3254	3283	4499	4836	4861
25	B	720	876	3284	3313	4529	4866	4891
26	B	750	906	3314	3343	4559	4896	4921
27	B	780	936	3344	3373	4589	4926	4951
28	B	810	966	3374	3403	4619	4956	4981
29	B	840	996	3404	3433	4649	4986	5011

30	B	870	1026	3434	3463	4679	5016	5041
31	A	900	1056	1664	1693	2009	2346	2371
32	A	930	1086	1694	1723	2039	2376	2401
33	A	960	1116	1724	1753	2069	2406	2431
34	B	990	1146	3554	3583	4799	5136	5161
35	B	1020	1176	3584	3613	4829	5166	5191
36	B	1050	1206	3614	3643	4859	5196	5221
37	B	1080	1236	3644	3673	4889	5226	5251
38	B	1110	1266	3674	3703	4919	5256	5281
39	B	1140	1296	3704	3733	4949	5286	5311
40	B	1170	1326	3734	3763	4979	5316	5341
41	B	1200	1356	3764	3793	5009	5346	5371
42	A	1230	1386	1994	2023	2339	2676	2701
43	A	1260	1416	2024	2053	2369	2706	2731
44	A	1290	1446	2054	2083	2399	2736	2761
45	B	1320	1476	3884	3913	5129	5466	5491
46	B	1350	1506	3914	3943	5159	5496	5521
47	B	1380	1536	3944	3973	5189	5526	5551
48	B	1410	1566	3974	4003	5219	5556	5581
49	B	1440	1596	4004	4033	5249	5586	5611
50	B	1470	1626	4034	4063	5279	5616	5641
51	B	1500	1656	4064	4093	5309	5646	5671
52	B	1530	1686	4094	4123	5339	5676	5701
53	A	1560	1716	2324	2353	2669	3006	3031
54	A	1590	1746	2354	2383	2699	3036	3061
55	A	1620	1776	2384	2413	2729	3066	3091
56	B	1650	1806	4214	4243	5459	5796	5821
57	B	1680	1836	4244	4273	5489	5826	5851
58	B	1710	1866	4274	4303	5519	5856	5881
59	B	1740	1896	4304	4333	5549	5886	5911
60	B	1770	1926	4334	4363	5579	5916	5941
61	B	1800	1956	4364	4393	5609	5946	5971
62	B	1830	1986	4394	4423	5639	5976	6001
63	B	1860	2016	4424	4453	5669	6006	6031

64	B	1975	2131	4539	4568	5784	6121	6146
65	A	2039	2195	2803	2832	3148	3485	3510
66	A	2069	2225	2833	2862	3178	3515	3540
67	A	2099	2255	2863	2892	3208	3545	3570
68	A	2129	2285	2893	2922	3238	3575	3600
69	B	2159	2315	4723	4752	5968	6305	6330
70	B	2189	2345	4753	4782	5998	6335	6360
71	B	2219	2375	4783	4812	6028	6365	6390
72	B	2249	2405	4813	4842	6058	6395	6420
73	B	2279	2435	4843	4872	6088	6425	6450
74	B	2490	2646	5054	5083	6299	6636	6661
75	B	2520	2676	5084	5113	6329	6666	6691
76	A	2550	2706	3314	3343	3659	3996	4021
77	A	2580	2736	3344	3373	1109	4026	4051
78	A	2610	2766	3374	3403	1109	4056	4081
79	B	2640	2796	5204	5233	6449	6786	6811
80	B	2670	2826	5234	5263	6479	6816	6841
81	B	2700	2856	5264	5293	6509	6846	6871
82	B	2730	2886	5294	5323	6539	6876	6901
83	B	2760	2916	5324	5353	6569	6906	6931
84	B	2790	2946	5354	5383	6599	6936	6961
85	B	2820	2976	5384	5413	6629	6966	6991
86	B	2850	3006	5414	5443	6659	6996	7021
87	A	2880	3036	3644	3673	1109	4326	4351
88	A	2910	3066	3674	3703	1109	4356	4381
89	A	2940	3096	3704	3733	1109	4386	4411
90	B	2970	3126	5534	5563	6779	7116	7141
91	B	3000	3156	5564	5593	6809	7146	7171
92	B	3030	3186	5594	5623	6839	7176	7201
93	B	3060	3216	5624	5653	6869	7206	7231
94	B	3090	3246	5654	5683	6899	7236	7261
95	B	3120	3276	5684	5713	6929	7266	7291
96	B	3150	3306	5714	5743	6959	7296	7321
97	B	3180	3336	5744	5773	6989	7326	7351

98	A	3210	3366	3974	4003	4319	4656	4681
99	A	3240	3396	4004	4033	1109	4686	4711
100	A	3270	3426	4034	4063	1109	4716	4741
101	B	3300	3456	4064	4093	1109	4746	4771
102	B	3330	3486	4094	4123	1109	4776	4801
103	B	3360	3516	4124	4153	1109	4806	4831
104	B	3390	3546	4154	4183	1109	4836	4861
105	B	3420	3576	4184	4213	1109	4866	4891
106	B	3450	3606	4214	4243	1109	4896	4921
107	B	3565	3721	4329	4358	1109	5011	5036
108	B	3595	3751	4359	4388	1109	5041	5066
109	B	3625	3781	4389	4418	1109	5071	5096
110	A	3655	3811	4419	4448	1109	5101	5126
111	A	3685	3841	4449	4478	1109	5131	5156
112	A	3715	3871	4479	4508	1109	5161	5186
113	A	3745	3901	4509	4538	1109	5191	5216
114	A	3775	3931	4539	4568	1109	5221	5246
115	A	3890	4046	4654	4683	1109	5336	5361
116	A	3920	4076	4684	4713	1109	5366	5391
117	A	3950	4106	4714	4743	1109	5396	5421
118	A	3980	4136	4744	4773	1109	5426	5451
119	A	4010	4166	4774	4803	1109	5456	5481
120	A	4040	4196	4804	4833	1109	5486	5511
121	A	4070	4226	4834	4863	1109	5516	5541
122	A	4100	4256	4864	4893	1109	5546	5571
123	A	4215	4371	4979	5008	1109	5661	5686
124	A	4245	4401	5009	5038	1109	5691	5716
125	A	4275	4431	5039	5068	1109	5721	5746
126	A	4305	4461	5069	5098	1109	5751	5776
127	A	4335	4491	5099	5128	1109	5781	5806
128	A	4365	4521	5129	5158	1109	5811	5836
129	A	4395	4551	5159	5188	1109	5841	5866
130	A	4425	4581	5189	5218	1109	5871	5896
131	A	4540	4696	5304	5333	1109	5986	6011

132	A	3685	3841	4449	4478	1109	5131	5156
133	A	3715	3871	4479	4508	1109	5161	5186
134	A	3745	3901	4509	4538	1109	5191	5216
135	A	3775	3931	4539	4568	1109	5221	5246
136	A	3890	4046	4654	4683	1109	5336	5361
137	A	3920	4076	4684	4713	1109	5366	5391
138	A	3950	4106	4714	4743	1109	5396	5421
139	A	3980	4136	4744	4773	1109	5426	5451
140	A	4010	4166	4774	4803	1109	5456	5481
141	A	4040	4196	4804	4833	1109	5486	5511
142	A	4070	4226	4834	4863	1109	5516	5541
143	A	4100	4256	4864	4893	1109	5546	5571
144	A	4215	4371	4979	5008	1109	5661	5686
145	A	4245	4401	5009	5038	1109	5691	5716
146	A	4275	4431	5039	5068	1109	5721	5746
147	A	4305	4461	5069	5098	1109	5751	5776
148	A	4335	4491	5099	5128	1109	5781	5806
149	A	4365	4521	5129	5158	1109	5811	5836
150	A	4570	4726	5334	5363	1109	6016	6041
151	A	4600	4756	5364	5393	1109	6046	6071
152	A	4630	4786	5394	5423	1109	6076	6101
153	A	4660	4816	5424	5453	1109	6106	6131
154	A	4690	4846	5454	5483	1109	6136	6161
155	A	4720	4876	5484	5513	1109	6166	6191
156	A	4750	4906	5514	5543	1109	6196	6221
157	A	4865	5021	5629	5658	1109	6311	6336
158	A	4895	5051	5659	5688	1109	6341	6366
159	A	4925	5081	5689	5718	1109	6371	6396
160	A	4955	5111	5719	5748	1109	6401	6426

附录 4:

A、B、C 型芯片的调度结果

芯片序号	芯片类型	进仓时间	第一次温育起始时间	磁珠加样起始时间	第二次温育起始时间	清洗起始时间	检测起始时间	检测结束时间
1	B	0	150	156	1664	1693	2293	2309
2	B	30	180	186	1694	1723	2323	2339
3	B	60	210	216	1724	1753	2353	2369
4	B	90	240	246	1754	1783	2383	2399
5	B	120	270	276	1784	1813	2413	2429
6	B	150	300	306	1814	1843	2443	2459
7	B	180	330	336	1844	1873	2473	2489
8	B	210	360	366	1874	1903	2503	2519
9	C	240	390	396	1904	1933	2533	2549
10	C	270	420	426	1934	1963	2563	2579
11	A	300	450	456	1964	1993	2593	2609
12	B	330	480	486	1994	2023	2623	2639
13	B	360	510	516	2024	2053	2653	2669
14	B	390	540	546	2054	2083	2683	2699
15	B	420	570	576	2084	2113	2713	2729
16	B	450	600	606	2114	2143	2743	2759
17	B	480	630	636	2144	2173	2773	2789
18	B	510	660	666	2174	2203	2803	2819
19	B	540	690	696	2204	2233	2833	2849
20	C	570	720	726	2234	2263	2863	2879
21	C	600	750	756	2264	2293	2893	2909
22	C	630	780	786	2294	2323	2923	2939
23	B	660	810	816	2324	2353	2953	2969
24	B	690	840	846	2354	2383	2983	2999
25	B	720	870	876	2384	2413	3013	3029
26	B	750	900	906	2414	2443	3043	3059
27	B	780	930	936	2444	2473	3073	3089

28	B	810	960	966	2474	2503	3103	3119
29	B	840	990	996	2504	2533	3133	3149
30	B	870	1020	1026	2534	2563	3163	3179
31	A	900	1050	1056	2564	2593	3193	3209
32	A	930	1080	1086	2594	2623	3223	3239
33	A	960	1110	1116	2624	2653	3253	3269
34	B	990	1140	1146	2654	2683	3283	3299
35	B	1020	1170	1176	2684	2713	3313	3329
36	B	1050	1200	1206	2714	2743	3343	3359
37	B	1080	1230	1236	2744	2773	3373	3389
38	B	1110	1260	1266	2774	2803	3403	3419
39	B	1140	1290	1296	2804	2833	3433	3449
40	B	1170	1320	1326	2834	2863	3463	3479
41	B	1200	1350	1356	2864	2893	3493	3509
42	A	1230	1380	1386	2894	2923	3523	3539
43	A	1260	1410	1416	2924	2953	3553	3569
44	A	1290	1440	1446	2954	2983	3583	3599
45	B	1320	1470	1476	2984	3013	3613	3629
46	B	1350	1500	1506	3914	3943	5143	5159
47	B	1380	1530	1536	3944	3973	5173	5189
48	B	1410	1560	1566	3974	4003	5203	5219
49	B	1440	1590	1596	4004	4033	5233	5249
50	B	1470	1620	1626	4034	4063	5263	5279
51	B	1500	1650	1656	4064	4093	5293	5309
52	B	1530	1680	1686	4094	4123	5323	5339
53	A	1560	1710	1716	2324	2353	2653	1109
54	A	1590	1740	1746	2354	2383	2683	1109
55	A	1620	1770	1776	2384	2413	2713	1109
56	B	1650	1800	1806	4214	4243	5443	5459
57	B	1680	1830	1836	4244	4273	5473	5489
58	B	1710	1860	1866	4274	4303	5503	5519
59	B	1740	1890	1896	4304	4333	5533	5549

60	B	1770	1920	1926	4334	4363	5563	5579
61	B	1800	1950	1956	4364	4393	5593	5609
62	B	0	150	156	1664	1693	2293	2309
63	B	30	180	186	1694	1723	2323	2339
64	B	60	210	216	1724	1753	2353	2369
65	A	90	240	246	1754	1783	2383	2399
66	A	120	270	276	1784	1813	2413	2429
67	A	150	300	306	1814	1843	2443	2459
68	A	180	330	336	1844	1873	2473	2489
69	B	210	360	366	1874	1903	2503	2519
70	B	240	390	396	1904	1933	2533	2549
71	B	270	420	426	1934	1963	2563	2579
72	B	300	450	456	1964	1993	2593	2609
73	B	330	480	486	1994	2023	2623	2639
74	B	360	510	516	2024	2053	2653	2669
75	B	390	540	546	2054	2083	2683	2699
76	A	420	570	576	2084	2113	2713	2729
77	A	450	600	606	2114	2143	2743	2759
78	A	480	630	636	2144	2173	2773	2789
79	B	510	660	666	2174	2203	2803	2819
80	B	540	690	696	2204	2233	2833	2849
81	B	570	720	726	2234	2263	2863	2879
82	B	600	750	756	2264	2293	2893	2909
83	B	630	780	786	2294	2323	2923	2939
84	B	660	810	816	2324	2353	2953	2969
85	B	690	840	846	2354	2383	2983	2999
86	B	720	870	876	2384	2413	3013	3029
87	A	750	900	906	2414	2443	3043	3059
88	A	780	930	936	2444	2473	3073	3089
89	A	810	960	966	2474	2503	3103	3119
90	B	840	990	996	2504	2533	3133	3149
91	B	870	1020	1026	2534	2563	3163	3179

92	B	900	1050	1056	2564	2593	3193	3209
93	B	930	1080	1086	2594	2623	3223	3239
94	B	960	1110	1116	2624	2653	3253	3269
95	B	990	1140	1146	2654	2683	3283	3299
96	B	1020	1170	1176	2684	2713	3313	3329
97	B	3180	3336	5744	5773	6989	7326	7351
98	A	3210	3366	3974	4003	4319	4656	4681
99	A	3240	3396	4004	4033	1109	4686	4711
100	A	3270	3426	4034	4063	1109	4716	4741
101	B	3300	3456	4064	4093	1109	4746	4771
102	B	3330	3486	4094	4123	1109	4776	4801
103	B	3360	3516	4124	4153	1109	4806	4831
104	B	3390	3546	4154	4183	1109	4836	4861
105	B	3420	3576	4184	4213	1109	4866	4891
106	B	3450	3606	4214	4243	1109	4896	4921
107	B	3565	3721	4329	4358	1109	5011	5036
108	B	3595	3751	4359	4388	1109	5041	5066
109	B	3625	3781	4389	4418	1109	5071	5096
110	A	3655	3811	4419	4448	1109	5101	5126
111	A	3685	3841	4449	4478	1109	5131	5156
112	A	3715	3871	4479	4508	1109	5161	5186
113	A	3745	3901	4509	4538	1109	5191	5216
114	A	3775	3931	4539	4568	1109	5221	5246
115	A	3890	4046	4654	4683	1109	5336	5361
116	A	3920	4076	4684	4713	1109	5366	5391
117	A	3950	4106	4714	4743	1109	5396	5421
118	A	3980	4136	4744	4773	1109	5426	5451
119	A	4010	4166	4774	4803	1109	5456	5481
120	A	4040	4196	4804	4833	1109	5486	5511
121	A	4070	4226	4834	4863	1109	5516	5541
122	A	4100	4256	4864	4893	1109	5546	5571
123	A	4215	4371	4979	5008	1109	5661	5686
124	A	4245	4401	5009	5038	1109	5691	5716
125	A	4275	4431	5039	5068	1109	5721	5746

126	A	4305	4461	5069	5098	1109	5751	5776
127	A	4335	4491	5099	5128	1109	5781	5806
128	A	4365	4521	5129	5158	1109	5811	5836
129	A	4395	4551	5159	5188	1109	5841	5866
130	A	4425	4581	5189	5218	1109	5871	5896
131	A	4540	4696	5304	5333	1109	5986	6011
132	A	4600	4410	4416	6824	6853	8053	8069
133	A	4630	4440	4446	6854	6883	8083	8099
134	A	4660	4470	4476	6884	6913	8113	8129
135	A	4690	4500	4506	6914	6943	8143	8159
136	A	4720	4530	4536	6944	6973	8173	8189
137	A	4750	4560	4566	6974	7003	8203	8219
138	A	4865	4590	4596	7004	7033	8233	8249
139	A	4895	4620	4626	7034	7063	8263	8279
140	A	4925	4650	4656	7064	7093	8293	8309
141	A	4955	4680	4686	7094	7123	8323	8339
142	A	4985	4710	4716	7124	7153	8353	8369
143	A	5015	4740	4746	7154	7183	8383	8399
144	A	5045	4770	4776	7184	7213	8413	8429
145	A	5075	4800	4806	7214	7243	8443	8459
146	A	5190	4830	4836	7244	7273	8473	8489
147	A	5220	4860	4866	7274	7303	8503	8519
148	A	5250	4890	4896	7304	7333	8533	8549
149	A	5280	4920	4926	7334	7363	8563	8579
150	A	4570	4726	5334	5363	1109	6016	6041
151	A	4600	4756	5364	5393	1109	6046	6071
152	A	4630	4786	5394	5423	1109	6076	6101
153	A	4660	4816	5424	5453	1109	6106	6131
154	A	4690	4846	5454	5483	1109	6136	6161
155	A	4720	4876	5484	5513	1109	6166	6191
156	A	4750	4906	5514	5543	1109	6196	6221
157	A	4865	5021	5629	5658	1109	6311	6336
158	A	4895	5051	5659	5688	1109	6341	6366
159	A	4925	5081	5689	5718	1109	6371	6396

160	A	4955	5111	5719	5748	1109	6401	6426

附录 5：

```
clc;
clear;
axis([0,2000,0,15.5]);
set(gca,'xtick',0:50:2000) ;
set(gca,'ytick',0:1:15.5) ;
xlabel('Ê±¼ä£¨s£©','FontName','·ÂËÎ','Color','k','FontSize',16)
ylabel('Ð¾Æ¬Ðòº Å','FontName','·ÂËÎ','Color','k','FontSize',16,'Rotation',90)
title('·ÖÎöÒÇµÄÓÅ»¯µ÷¶È','fontname','ËÎÌå','Color','k','FontSize',16);
n_bay_nb = 15;
n_task_nb = 165;
n_start_time = xlsread('result_1_3.xlsx');
n_duration_time =xlsread('result_2.xlsx');
n_bay_start = xlsread('result_3.xlsx');
n_job_id = xlsread('color.xlsx');
rec=[0,0,0,0];
color=[0.00,0.45,0.74;
            0.41,0.61,0.35;
            0.67,0.31,0.04;
            0.93,0.69,0.13;
            0.90,0.50,0.20;
            0.46,0.76,0.50;
            0.50,0.50,0.50];
for i = 1:n_task_nb
   rec(1) = n_start_time(i);
   rec(2) = n_bay_start(i)+0.7;
   rec(3) = n_duration_time(i);
   rec(4) = 0.6;

      U = rectangle('Position',rec,'LineWidth',0.5,'LineStyle','-','FaceCol-
or',[color(n_job_id(i)+1,1),color(n_job_id(i)+1,2),color(n_job_id(i)+1,3)]);
end
```

```
    from time import time
    from pyeasyga import pyeasyga
    import random
    import codecs

    data=[]

    data.append(T)
    data.append(ni)
    data.append(ma)
```

```
    data.append(Mij)
    data.append(pjk)

    def max_processing_time(data):
        pjk=data[4]
        max_time=0
        for i in range(0,len(pjk)):
            for j in range(0,len(pjk[i])):
                if pjk[i][j]>max_time and pjk[i][j]!=1000:
                    max_time=pjk[i][j]
        return max_time


    def create_individual(data):
        individual=[]
        start_times=[0]*data[2]
        jobs=data[0]
        list_to=[2,1,2,0,1,2,0,1,1,0]
        random_number=random.randint(0,len(list_to)-1)
        reference=list_to[random_number]
        if reference == 1:
          a=0
          for i in range(0,jobs):
              for j in range(0,data[1][i]):
                  position_X=random.randint(0,len(data[3][a])-1)
                  X=data[3][a][position_X]
                  S=start_times[X-1]
                  individual.append(S)
                  individual.append(X)
                  start_times[X-1]=start_times[X-1]+data[4][a][X-1]
                  a+=1
        elif reference == 2:
          a=len(data[3])-1
          for i in range(0,jobs):
              for j in range(0,data[1][i]):
                  position_X=random.randint(0,len(data[3][a])-1)
                  X=data[3][a][position_X]
                  S=start_times[X-1]
                  individual.append(S)
                  individual.append(X)
                  start_times[X-1]=start_times[X-1]+data[4][a][X-1]
                  a-=1
        else:
          for i in range(0,jobs):
              for j in range(0,data[1][i]):
                  X=random.randint(1,data[2])
                  max_time=max_processing_time(data)
                  S=random.randint(0,max_time)
                  individual.append(S)
                  individual.append(X)
```

```
            return individual

        def mutate(individual):
            mutate_index1=random.randrange(len(individual))
            mutate_index2=random.randrange(len(individual))
            #max_time=max_processing_time(data)
            if ((mutate_index1%2)==0 and (mutate_index2%2)==0) or ((mutate_index1%2)!=0 and \
                (mutate_index2%2!=0)):
                individual[mutate index1], individual[mutate index2] = individual[mutate index2], individua
l[mutate_index1]
            elif (mutate_index1%2)==0 and (mutate_index2%2)!=0:
                #if individual[mutate_index1]>(max_time/2):
                 # individual[mutate index1]=individual[mutate index1]+random.randint(-(max time/2),(max
_time/2))
                new_index=random.randrange(0,len(individual),2)
                individual[mutate index1], individual[new index] = individual[new index], individual[mutat
e_index1]
                individual[mutate_index2]=random.randint(1,data[2])
            else:
                #if individual[mutate_index2]>(max_time/2):
                 # individual[mutate index2]=individual[mutate index2]+random.randint(-(max time/2),(max
_time/2))
                new_index=random.randrange(0,len(individual),2)
                individual[mutate index2], individual[new index] = individual[new index], individual[mutat
e_index2]
                individual[mutate_index1]=random.randint(1,data[2])

        def is_feasible_machine(operation,machine,data):
            Mij=data[3]
            count=0
            for i in range(0,len(Mij[operation])):
                if machine==Mij[operation][i]:
                    count+=1
            if count == 0:
                return False
            else:
                return True

        def operations_in_machine(machine,individual):
            result=[]
            i=0
            while i<len(individual):
                if individual[i+1]==machine:
                    result.append(int(i/2))
                i+=2
            return result

        def fitness(individual,data):
            fitness=0
            pjk=data[4]
            i=0
            for op in range(0,len(pjk)):
```

```
            if (individual[i]+pjk[op][individual[i+1]-1])>fitness:
                fitness=individual[i]+pjk[op][individual[i+1]-1]
            i+=2
        # ------restrictions----------
        fouls=0
        j=0
        k=0
        # for each job, C of current operation must be less than the next
        for job in range(0,len(ni)):
            for op2 in range(0,ni[job]-1):
                if (individual[j]+pjk[k][individual[j+1]-1])>individual[j+2] or\
                    individual[j]>=individual[j+2]:
                        fouls+=4
                j+=2
                k+=1
            j+=2
            k+=1
        l=0
        while l<len(individual):
            if not is_feasible_machine(int(l/2),individual[l+1],data):
                fouls+=2
            l+=2

        count_zeros=0
        for machine2 in range(1,data[2]+1):
            operations2=operations_in_machine(machine2,individual)
            for op4 in range(0,len(operations2)):
                if individual[operations2[op4]*2]==0:
                    count_zeros+=1
                start_reference=individual[operations2[op4]*2]
                end_reference=individual[operations2[op4]*2]+pjk[operations2[op4]][machine2-1]
                for op5 in range(0,len(operations2)):
                    if op5 != op4:
                        s=individual[operations2[op5]*2]
                        c=individual[operations2[op5]*2]+pjk[operations2[op5]][machine2-1]
                        if s<=start_reference and c>=end_reference:
                            fouls+=2
                        elif s>=start_reference and s<=end_reference and c<=end_reference:
                            fouls+=2
                        elif s<=start_reference and c>start_reference and c<=end_reference:
                            fouls+=2
                        elif s>=start_reference and s<end_reference and c>=end_reference:
                            fouls+=2
        if count_zeros == 0:
            fouls+=1
        fitness=fitness+(fouls*1000)
        return fitness


steps=[]
```

```
    count_increment=0

    def genetic_algorithm_scheduling(data,counter,pop_size=100,num_generations=500):
        start_time = time()
        ga = pyeasyga.GeneticAlgorithm(data,maximise fitness=False,population size=pop size,generati
ons=num_generations,mutation_probability=0.3) # initialization of the algorithm
        ga.create_individual=create_individual
        ga.mutate_function=mutate
        ga.fitness_function=fitness
        ga.run()
        best_individual=ga.best_individual()
        steps.append(best_individual)
        best_fitness=best_individual[0]
        if best_fitness>1000 and counter<10:
            counter+=1
            new_generations=num_generations+100
            genetic_algorithm_scheduling(data,counter,pop_size,new_generations)
        elif best_fitness>1000 and counter==10:
            print(ga.best_individual())
            end_time=time()
            print("检验总时长",(end_time-start_time))



    genetic_algorithm_scheduling(data,count_increment,pop_size=200)
```