第八届湖南省研究生数学建模竞赛

基于改进遗传算法的化学发光免疫分析仪运行调度优化

摘要

本文针对全自动化学发光免疫分析仪中五种情形下芯片输入的调度问题，基于决策论与动态规划的思想，将其抽象为以最大化芯片数量以及最小化完工时间为目标的可重入混合流水线调度(RHFSP)数学模型, 并使用在精英选择策略、重组及变异操作上改进后的自适应 CHC 遗传算法(ACHC-GA)对模型进行求解。

针对问题一：对于化学发光免疫分析仪的运行流程进行分析，要求在单位时间内尽可能多地处理的A类或B类芯片数量，建立了RHFSP单类型芯片检测最优化模型MOD1，该模型充分考虑了芯片进仓时间、芯片的每道工序开始时间对总共处理芯片数量的影响，通过基于跨代精英选择、异物种重组、大变异策略的**遗传算法**进行求解，求解得到的 A 类芯片单位时间内的最大完工数量为 **14** 片；B 类芯片最大完工数量为 **0** 片。

针对问题二：要求计算混合处理给定数量的 A 类及 B 类芯片的最短总完工时间。在问题 1 的基础上去除加工时间约束，修改目标函数为总最短完工时长，建立了 RHFSP 双类芯片处理时间最小化调度模型 MOD2。设计了**资源优化温育共享解码**方案，充分利用有限资源，减少等待时间，极大提高了求解质量，求解得到的 A 类、B 类芯片混合处理的最短总完工时间为 27543s。

针对问题三：在问题二的基础上，将芯片的类型增加到 3 类：A、B、C 类，建立了 RHFSP 三类芯片处理时间最小化调度模型 MOD3, 求解得到的 A 类、B 类、C 类芯片混合处理的最短总完工时间为 27230s。

针对问题四：从**理论推导**证明了将模型与算法推广到具有 N 种类型芯片，数量为m_i片一般情形的可行性，并建立了RHFSP混合芯片处理时间最小化调度通用模型MOD4。**通过实验**进一步验证了所提模型与算法的有效性及普适性。

针对问题五：在问题三的基础上，引入 C 类芯片进仓时间约束，基于此建立了 RHFSP 芯片异步处理时间最小化调度模型 MOD5，并设计了新的解码方案，求解得到的 A 类、B 类、C 类芯片混合处理的最短总完工时间为 26760s。

最后, 我们对提出的模型进行全面的评价: 本文的模型能合理解决提出的化学发光免疫分析仪运行调度优化问题, 具有实用性强, 算法效率高等特点, 该模型在可重入混合流水线调度问题的处理上均能适用。

关键词：化学发光免疫分析仪　可重入混合流水线优化　调度优化　遗传算法

目录

1 问题综述

1.1 问题背景

化学发光免疫分析(CLIA)技术不同于于荧光免疫、放射免疫、酶联免疫等分析技术，它将化学发光技术与免疫反应结合，用于测定各种抗原、抗体、脂肪酸、酶、激素等标志物含量，是新兴的应用于临床疾病诊断的免疫检测技术。在化学发光原材料方面，国产化学发光厂家正在逐渐摆脱酶、抗原抗体等核心原料严重依赖外国进口的现状，化学发光免疫分析技术在临床诊断方面发展势头较猛[1] 。全自动化学发光免疫分析仪是基于该技术特制的仪器，它通过人体体液样本，例如血清、唾液等，实现对诸如蛋白、核酸等指标的精准检测，从而根据检测指标判断被检者是否患有某种疾病。其广泛应用于生物化学、医学及环境检验等方面，是一种高端自动化的分析仪器，常用于检测人体内的肿瘤标志物、感染性疾病、药物浓度、特定功能的蛋白、体液免疫相关物质、内分泌激素等指标，逐渐成为免疫

检验主流使用的仪器。全自动化学发光免疫分析仪采用全自动操作自主完成样本前处理、温育、清洗及检测等实验内容，化学反应简单，从而克服传统人工检测存在的效率低、结果精确度差、耗时费力等缺点，极大程度上方便了分析人体体液的需求，具备快检测、高灵敏、特异性好等优势，最重要的是具有较好的检测结果。该仪器的应用是将带有人体体液的生物芯片作为载体送入分析仪内，通过前期处理、第一次温育、磁珠加样、第二次温育、清洗及检测 6 个核心步骤完成芯片的检测过程，如图 1 所示。

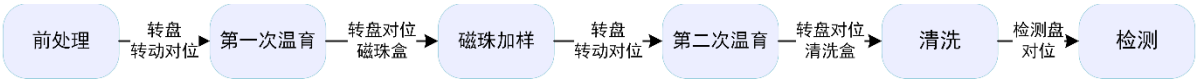

图 1 全自动化学发光免疫分析仪检测流程

随着医院人流量的增大，对于免疫检测分析的需求增多，等待所有工序完成检验显然比较耗时，如何充分利用每台全自动化学发光免疫分析仪，在短时间内做尽可能多的芯片分析逐渐成为目前值得深究的问题。该问题可抽象为可重入混合流水线优化(RHFSP)问题，许多学者用遗传算法、粒子群算法、极大学习机等传统算法解决了该类问题，较新的算法有灰狼算法和布谷鸟算法、离散花朵授粉算法，但目前几乎没有学者对化学发光免疫分析仪中芯片输入的调度优化进行探究，本文尝试使用遗传算法结合化学发光免疫分析仪调度进行优化，根据实际问题建立起合适的数学模型进行求解。

1.2 问题提出

全自动化学发光免疫分析仪内含 6 道处理工序，且每道工序所含处理工位的数量不同，是一种可重入混合流水线优化问题。可重入混合流水线优化涉及多方面的考虑, 在传统流水线调度问题的基础上发掘了新的内容。相较于传统流水线调度并结合本次对化学发光免疫分析仪的研究内容，其特征表现为加工的芯片存在多个加工工序，且 6 道工序中有两道工序存在两个以上的加工工位，它属于组合优化问题。流水线优化问题在目标方面往往由完工时间、处理数量、完工效率等内容组成，在该研究中，聚焦于完工数量与完工时间。题目以全自动化学发光免疫分析仪的内部操作为背景,介绍了化学发光免疫分析仪的外形及结构，并详细描述了从芯片输入到检测完成的一系列工序，并给出每道工序对应的时间及顺序, 问题一要求我们计算单位时间内最多能够处理的 A 类及 B 类芯片数量；问题二要求我们在给定芯片数量的情况下最少耗费的完工时间，在问题一基础上混合了两类芯片；问题三在问题二的基础上进一步将 C 类芯片放入检测；问题四进

一步考虑芯片数量为未知的情况，试验模型能否推广到 N 类芯片；问题五多考虑了中途放入芯片的情况。

基于题目文件给出的全自动化学发光免疫分析仪工作流程，我们首先考虑仅有一类芯片的情况下处理一组芯片所需耗费的最短时间，随后循序渐进来处理多组芯片混合所需耗费的最短时间. 对于可重入混合流水线优化问题，每个工位一个时间只能处理一个芯片，且每个芯片也至多在一个工位上加工，全自动化学发光免疫分析仪拥有处理芯片的 6 个核心步骤，也不例外是任意时刻每个芯片至多在一个工位上加工。化学发光免疫分析仪的结构包括前处理工作台、大转盘(转盘本身具有温育功能，亦称温育盘)和转盘上方固定的 8 个清洗盒、1 个磁珠加样盒，以及 1 个可以转动的检测盘等。转盘上均匀分布着用来置放芯片的 40 个卡槽。即某时刻除温育转盘可以加工 40 个芯片，清洗盒可以清洗 8 个芯片外，其他每道工序只包含一个工位，一个工位只能加工一个芯片；在此问题中，每个工位有固定的芯片输入顺序，不可打乱。芯片处理时，整个分析仪的执行时间包括处理时间和各个工位部件的对位时间。芯片数量及其在每个工位上的执行时间是已知的、确定的并且是独立的。并且由题目文件给出的说明可知，一旦开始在工位上处理芯片，此过程在完成之前不能中断。图 2 给出了该问题的示意图。

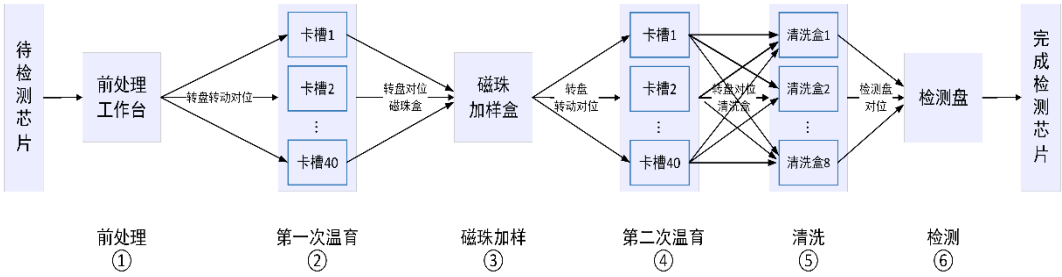

图 2 化学发光免疫分析仪运行的调度优化问题示意图

进一步分析发现，全自动化学发光免疫分析仪处理芯片的时间会受芯片输入时间，输入类型，输入顺序的影响。全自动化学发光免疫分析仪处理一组芯片所需耗费的时间与一组内每个芯片输入的时间联系紧密。温育工位和清洗工位拥有多个处理卡槽，且是耗费时间较长的工序，那么合理安排芯片输入的时间，可以充分地利用温育及清洗的工位，使得整组芯片的处理时间缩短。在芯片有多种类型时，不同类型芯片的输入顺序也会完工时间造成影响，不同类型芯片的温育时间有差异，利用温育时间的不同可以更充分地利用温育系统。

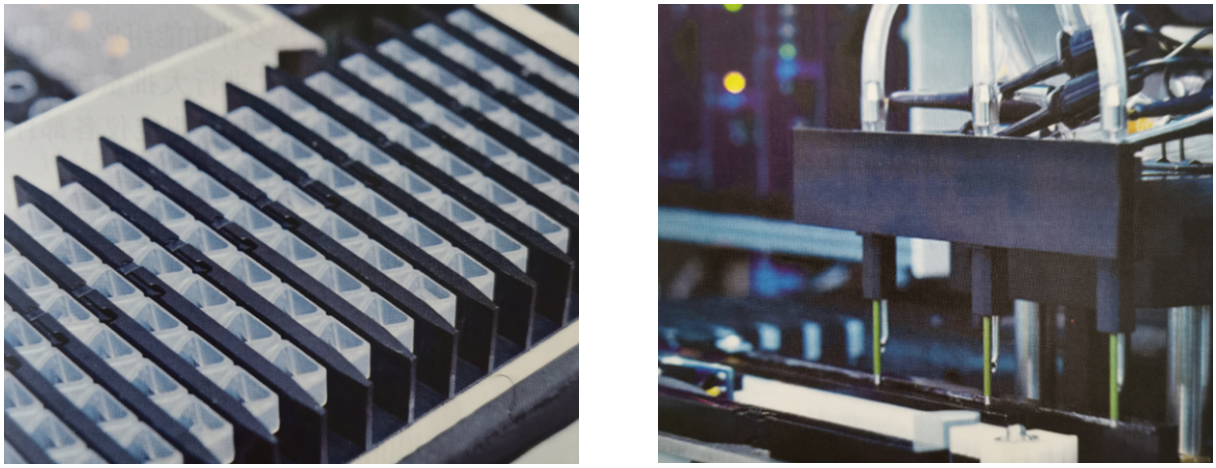

图 3 温育系统(左)清洗系统(右)

本文需要从给定时间内芯片处理数量最多, 给定芯片数量芯片处理时间最短以及多类型芯片同时处理时间最短角度考虑, 解决以下 5 个问题:

(1) 问题一：要求根据给定的 A、B 两组芯片，获取单位时间内能够单独检测各组芯片的最大数量。其中两组芯片的两次温育时间各不相同，A 类型芯片的两次温育时间分别为 10min、5min；B 类型芯片的两次温育时间分别为 40min、20min。
(2) 问题二：要求根据给定数量的 A、B 两组芯片，获取全部混合检测完芯片至少耗费的时间。其中两种类型芯片数量各为 80 片。
(3) 问题三：要求在问题二的基础上新增 C 型芯片，获取全部检测完芯片至少耗费的时间。其中三种类型芯片数量分别为 60 片、50 片、50 片，C 类型芯片的两次温育时间分别为 25min、10min。
(4) 问题四：要求在问题二和问题三的基础上，将模型和算法分别推广到芯片数量为m_i片，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。进一步举例验证推广到 N 种类型的芯片是否可行。
(5) 问题五：要求在 A、B 两组芯片已经开始检验 30 分钟后新增检验 C 型芯片 20 片，获取全部混合检测完芯片至少耗费的时间。其中 A 型芯片 60 片、B 型芯片 70 片。

1.3 资料条件

【2023B】题目文件提供了全自动化学发光免疫分析仪的工作原理和全自动化学发光免疫分析仪处理芯片的方案要求。在该文件中介绍了全自动化学发光免疫分析仪，人体的体液样本事先被放到一个称为生物芯片的载体中，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。文件把芯片的读码、激光超声、过滤、定量和 R1R2 加样归为前处理阶段，芯片检测的核心流程主要包括前处理、第一次温育、磁珠加样、第二次温育、清洗和检测等六道工序。每道工序都有其固定的处理工位，也需要一定的时间，并且由于检测需要，每片芯片的检测流程一旦开始便不能停止，直到检测完成。

2 模型假设与符号说明

2.1 模型基本假设

(1)工序的顺序是确定的；
(2)每个工位只能加工一个芯片；
(3)任意时刻每个芯片至多在一个工位上加工；
(4)每个芯片可以在某工序的任意一个工位上加工；
(5)执行时间包括处理时间和对位时间；
(6)芯片数量及其在每个工位上的执行时间是已知的、确定的并且是独立的；
(7)一旦开始在工位上处理芯片，此过程在完成之前不能中断。

2.2 符号说明

假设需处理n个芯片，每个芯片要经历c道工序，共有加工工位m个。最后一个芯片(第n个)完成处理的时刻即整个处理阶段的完工时间，用$C(m_c,n)$表示。m个工位中，第 1 个为前处理工位，2～41 为温育工位，42 为磁珠工位，43～50 为清洗工位，51 为检测工位。本文定义了如下 11 个使用次数较多的符号, 其余符号在使用时注明。

表1 符号说明

符号	含义	单位
n	芯片数量	个
c	工序数量	道
m	工位数量	个
q	芯片类型	类
M_k	第k道工序的可用工位集合	无
$p(k,j)$	芯片j在工序k上的执行时间	秒
$S(i_k,j)$	芯片j在工序k的工位i上的开始加工时刻	秒
$C(i_k,j)$	芯片j在工序k的工位i上的结束加工时刻	秒
N_{ki}	工序k的工位i上加工的芯片数量	个
X_{ijk}	当芯片j在工序k的工位i上进行加工时$X_{ijk}=1$,否则$X_{ijk}=0$	无
T	总的最大加工时间限制	秒
Δ_k	工序k结束后转盘转动对位所需时间	秒

3 问题 1 模型建立与求解

3.1 问题 1 分析

题目文件中表 1 给出了各个工位处理芯片所需的时间，问题一要求我们计算单位时间内最多能够单独处理的 A 类及 B 类芯片数量，即 1 小时内通过 6 道工序的最多芯片数量。除开各个工序所需花费的时间，还有转动对位的时间。芯片前期处理的 5 个步骤在同一个工作台上顺序进行，前期处理总共花费 150s。前期处理后转盘转动对位花费 6s；第一次温育时长为 0～60min，A、B、C 三类芯片各不相同；温育后对位磁珠盒花费 8s，磁珠加样花费在 21～25s 的区间内；磁珠对位转盘花费 8s；对位后开始第二次温育，花费在 0～30min；后转盘对位清洗盘花费 16s，清洗花费大于等于 325s；清洗过后对位检测盘花费 12s；检测开始至结束花费 25s。整体运行流程如图 4 所示。

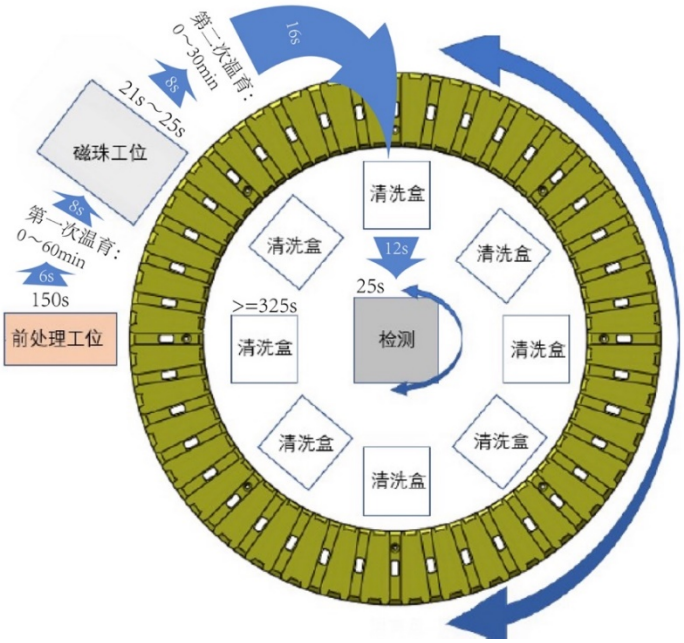

图 4　化学发光免疫分析仪运行结构展示

问题一需计算 A 类与 B 类芯片通过前处理、第一次温育、磁珠加样、第二次温育、清洗、检测工序，记录每道工序开始的时间以及总共花费的时间，并获取单位时间内处理的最大芯片数量。以分析 A 类为例，第一个芯片进仓从第 0s 开始，第一次温育开始的时间为前期处理的时间与转盘对位的时间之和 156s；磁珠加样开始的时间为上道工序开始时间、第一次温育与对位磁珠盒的时间之和 764s；第二次温育开始的时间为上道工序开始时间、磁珠加样的时间与对位转盘的时间之和 793s；清洗开始的时间为上道工序开始时间、第二次温育的时间与对位清洗盒的时间之和 1109s；检测开始的时间为上道工序开始时间、清洗的时间与对位检测盒的时间之和 1446s；最终检测结束的时间为检测开始的时间与检测工序的时间之和 1627s。A 类芯片的第一个芯片甘特图流程表示如图 5。

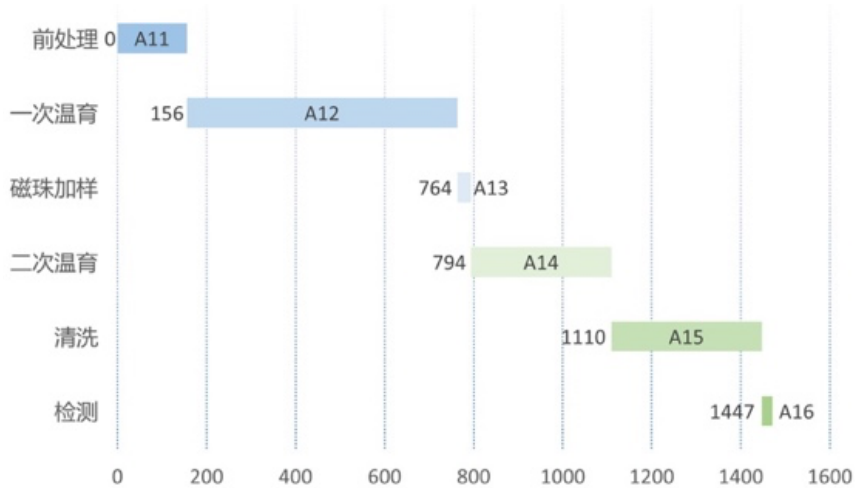

图 5　A 类第一个芯片 A1 处理流程甘特图

同理，后续芯片可以使用该流程计算出工序的结束以及开始时间，在前一芯片完成工序产生空位后加入处理。假设$\mathrm{C}(i_k,j)$是第j个芯片经过第i个工位所需的时间，此时是处理的第k道工序。同理$\mathrm{C}(i_1,j)$是第j个芯片经过第i个工位加工所需的时间；$\mathrm{C}(i_k,1)$是第1个芯片经过第i个工位所需的时间；$\mathrm{C}(i_1,1)$是第1个芯片经过第1个工位所需的时间。$\mathrm{p}(k,j)$为第j个芯片经过第k道工序的用时，各完工时间关系如下所示。

$$C(i_1,1)=p(1,1) \tag{1}$$

$$\mathrm{C}(i_k,1)=\mathrm{C}(i_{k-1},1)+\mathrm{p}(k,1) \tag{2}$$

$$\mathrm{C}(i_1,j)=\mathrm{C}(i_1,j-1)+\mathrm{p}(1,j) \tag{3}$$

$$\mathrm{C}(i_k,j)=\max\{C(i_{k-1},j),C(i_k,j-1)\}+\mathrm{p}(k,j) \tag{4}$$

经过分析，本研究可以建立目标函数为最大化芯片数量的模型，通过前述假设条件建立约束，满足单位时间的约束，使用遗传算法对其进行求解。需要满足的硬约束如下：

(1)唯一性约束：每个芯片在每道工序上的加工只能被分配给一个工位。

(2)分配约束：分配给每道工序内所有工位的芯片数之和为总的芯片数。

(3)工序处理顺序约束：当前工序未加工完成的芯片，下一工序不能开始处理。

(4)芯片处理时间约束：同一工序上，芯片完成时间取决于其在工位上的加工时间和开始时间。

(5)独占性约束：每个工位在同一时间最多仅能处理一个芯片。

(6)共享性约束：第一次温育与第二次温育共用同一大转盘上分布的卡槽。

(7)总加工时间约束：总加工时间满足在单位时间的范围内。

3.2 模型的准备与建立

3.2.1 数据定义

(1)常量

n——*芯片数量*

c——*工序数量*

m——*工位数量*

M_k——*第 k 道工序的可用工位集合*

$p(k,j)$——*芯片j在工序k上的执行时间*

T——*总的最大加工时间限制*

Δ_k——*工序 k 结束后转盘转动对位所需时间*

(2)变量

$S(i_k,j)$——*芯片j在工序k的工位i上的开始加工时刻*

$C(i_k,j)$——*芯片j在工序k的工位i上的结束加工时刻*

(3)中间变量

N_{ki}——*芯片数序 k 的工位 i 上加工的芯片数量*

X_{ijk}——*当芯片 j 在工序 k 的工位 i 上进行加工时*$X_{ijk}=1$,*否则*$X_{ijk}=0$

3.2.2 问题 1 模型建立

目标函数：由问题 1 可知，需要计算单位时间内能够处理的最多芯片处理数量。故目标函数设为最大化芯片数量。

$$\text{Max } n \tag{5}$$

约束条件：根据假设条件设置约束。

(1)唯一性约束：确保每个芯片在每道工序上的加工只能被分配给一个工位.

$$\sum_{i=1}^{M_k} X_{ijk}=1, i\in M_k, j\in[1,n], k\in[1,c] \tag{6}$$

(2)分配约束：分配给每道工序内所有工位的芯片数之和为总的芯片数.

$$\sum_{i=1}^{M_k} N_{ki}=n, k\in[1,c] \tag{7}$$

(3)工序处理顺序约束：即当前工序未加工完成的芯片，下一工序不能开始处理.

$$C(i_k,j)+\Delta_k \leq S(i_{k+1},j), i\in M_k, j\in[1,n], k\in[1,c] \tag{8}$$

(4)芯片处理时间约束：同一工序上，其完成时间取决于其在工位上的加工时间和开始时间.

$$C(i_k,j)=S(i_k,j)+p(k,j), i\in M_k, j\in[1,n], k\in[1,c] \tag{9}$$

其中，$S(i_k,j)$满足取值如下.

$$S(i_k,j)=max\{C(i_k,j-1), C((i-1)_k,j)\} \tag{10}$$

即，芯片i在阶段j的开始时间，等于芯片i紧前工序的完工时间或同一工位紧前芯片$i-1$的完工时间的最大值。

(5)独占性约束：每个工位在同一时间最多仅能处理一个芯片.

$$\sum_{j=1}^{n} X_{ijk} \le 1, i \in M_k, k \in [1,c] \tag{11}$$

(6)共享性约束：第一次温育与第二次温育共用同一大转盘上分布的卡槽.

$$M_2 = M_4 \tag{12}$$

(7)总加工时间约束.

$$C(m_c, n) \le T \tag{13}$$

(8)变量取值范围.

$$x_{ijk} = \begin{cases} 1, \text{若芯片 i 被安排在工序 j 的工位 k 上加工} \\ 0, \text{否则} \end{cases}, \forall i, j, k; \tag{14}$$

综上所述，可以建立问题 1 的数学模型。RHFSP 优化模型一的数学描述如下：

MOD1

$$\text{Max} \quad n$$

$$s.t. \begin{cases} \sum_{i=1}^{M_k} X_{ijk} = 1, i \in M_k, j \in [1,n], k \in [1,c] \\ \sum_{i=1}^{M_k} N_{ki} = n, k \in [1,c] \\ C(i_k, j) + \Delta_k \le S(i_{k+1}, j), \quad i \in M_k, j \in [1,n], k \in [1,c] \\ C(i_k, j) = S(i_k, j) + p(k, j), i \in M_k, j \in [1,n], k \in [1,c], k \in [1,c] \\ S(i_k, j) = \max\{C(i_k, j-1), C((i-1)_k, j)\} \\ \sum_{j=1}^{n} X_{ijk} \le 1, i \in M_k, k \in [1,c] \\ M_2 = M_4 \\ C(m_c, n) \le T \\ x_{ijk} = \begin{cases} 1, \text{若芯片 i 被安排在工序 j 的工位 k 上加工} \\ 0, \text{否则} \end{cases}, \forall i, j, k; \end{cases} \tag{15}$$

3.3 模型的求解与分析

问题 1 模型是一个可重入混合流水线优化问题，有多个作业多道工序，且每道工序拥有不同的工位数量，是 NP-hard 的组合优化问题。遗传算法作为解决复杂组合优化问题的有效方法，可以在本研究中应用。遗传算法主要思想是“优胜劣汰”、“适者生存”，是一种元启发式算法，本质上是一种随机搜索的过程。遗传算法的承载对象是“染色体”，每个染色体通过交叉变异操作产生新的后代。其中作为评判个体优劣的适应度值是整个遗传算法的核心，对应到研究内容即模型中的目标函数值，通过适应度值则可以判断后代与父代的优劣，从而保留较好的个体，摒弃较差的个体,通过这样的不断繁衍进化，最后收敛到一群最适应环境的个体，从而求得问题的优质解。

3.3.1 算法设计

算法的整体流程图如图所示。

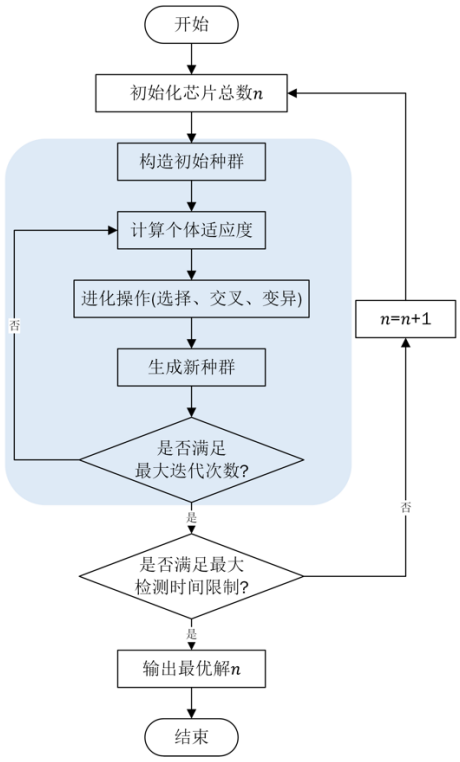

图 6　遗传算法解决问题 1 的流程图

STEP1 染色体编码

本文采用芯片排列的编码方法，即 n 个芯片以一定的随机顺序排成队列。用 $U_i=[u_1,u_2,...,u_n]$ 表示芯片集在初始工序的加工顺序，其中 u_i 表示第 i 个加工的芯片。例如 $U_i=[3,2,4,1,5]$ 表示第 i 个染色体中五个芯片的加工顺序为 $3\rightarrow2\rightarrow4\rightarrow1\rightarrow5$。

由于芯片的加工受上一工序影响较大，则在每个工序结束后对所有芯片根据其当前的结束时间进行非降序列重新排列编码，以此可以实现更快地收敛到问题的相对最优解。

STEP2 染色体解码

由于本问题带有约束条件，编码后的个体不一定是问题的可行解，因此采用解码算法通过约束控制对个体构造一个可行的调度方案。由于工序 2(第一次温育)和工序 4(第二次温育)属于可重入工序，共享并行卡槽，其解码方式有其特殊性，总体解码的具体步骤如下：

(1)当工序$k=1,5,6$时，即前处理、清洗和检测三个工序，因不具有可重入工序特征，采用一般的方法进行解码，即确定在工序k中各个芯片的工位选择和各个芯片的加工顺序，确定方法遵循步骤①-④；

①记芯片$j=1$；

②选择工位：优先选择芯片j在各个工位中计算出的完成时间最小的，若完成时间相同，则选取当前工位负载最小的工位，工位负载即累计运行时长，若工位负载也相同，则随机选取工位。芯片j在各个工位的完成时间计算如下：记该工位在当前加工完最后一个芯片的完成时间为$last_M$，记该芯片在该工序该工位上的运行时间为run_P，则芯片 j 在该工位上的完成时间为$last_M+run_P$。

③计算芯片 j 在上一步选择的工位m上的开始运行时间、运行时间和结束时间：开始运行时间 start 为芯片 j 在上一工序的完成时间和工位m在加工完上

一个芯片的完成时间的最大值，即$start = max(last_P, last_M)$；运行时间$run$为芯片$j$在该工序$k$该工位$m$上的运行时间$run_P$，即$run = run_P$；结束时间$end$为开始时间加上运行时间，即$end = start + run$；

④记录芯片j的开始时间、结束时间、运行时间和工位选择，以及工位m的开始时间、结束时间、运行时间和加工芯片；

⑤令$j = j + 1$，循环执行步骤②-④直到$j = n$；

(2)当$k = 2$时，为第一次温育工序，由于第一次温育和第二次温育共享 40 个卡槽，故在设计这三个工序的调度方案时需考虑可重入约束，具体遵循如下步骤：

①选择上一工序即前处理阶段完成时间最早的芯片即$j = 1$，进入第一次温育工序，得到第一个芯片在第一次温育的结束时间$e1$；

②判断在$e1$之前还有哪些芯片可以进入第一次温育工序，即当芯片若进入第一次温育，通过(1)中步骤②-④可以得到第一次温育的开始时间，若开始时间小于$e1$，则可以进入，由此充分利用工位资源；

③记完成第二次温育工序的芯片数量为 0;

④选择最早结束第一次温育的芯片进入磁珠加样工序

⑤判断该芯片磁珠加样结束前是否还可以有其他工件进入温育 1，由于磁珠工位已被该芯片占用，故不必判断是否有其他工件能进入磁珠工位；

⑥磁珠加样结束的芯片进入第二次温育工序

⑦判断该芯片第二次温育结束前是否还可以有其他工件进入第一次温育工序和磁珠加样工序；

⑧循环执行④-⑦直到全部工件均完成第二次温育工序。

STEP3 适应度函数

将染色体解码过程中得到的各芯片在各工序工位上的结束时间进行比较，用该调度方案的最晚结束时间的倒数作为适应度值，即最晚结束时间越早，适应度值越大，该染色体代表的调度方案越优。

STEP4 选择算子

遗传算法的选择算子通过筛选种群进化过程中的个体来影响种群的进化结果。本文使用轮盘赌选择算子，其基本思想是生存能力越好的个体被选择的概率越大，是一种有放回的随机采样方法。个体U_i(其中$i = 1,2,...,N$， N为种群大小)被选中的概率如式(16)所示，与其适应度函数的值成正比。然后将种群中所有个体的适应度值累计再归一化，如式(17)所示。最终产生一个随机数，选择随机数落在区域的对应个体作为选择的个体。

$$P_{select}(U_i) = \frac{fitness(U_i)}{\sum_{i=1}^{N} fitness(U_i)} \tag{16}$$

$$P_{select}(U_i) = \frac{\sum_{k=1}^{i} fitness(U_i)}{\sum_{j=1}^{N} fitness(U_j)} \tag{17}$$

STEP6 自适应交叉和变异算子

交叉是遗传算法全局优化的主要手段，通过对两个父代染色体基因的重新组合从而实现对解空间的高效搜索，而变异通过改变某个或多个基因座上的基因值为新个体的产生提供了机会。并改变交叉和变异算子的概率计算公式，使其跟随算法的迭代过程进行自适应更改。交叉概率 Pc 越大，新个体产生的速度越快。但过大会使优秀个体的结构

很快被破坏；Pc 过小，搜索过程缓慢，以至于停滞不前。变异概率 Pm 过小，不易产生新个体结构；Pm 过大，变成纯粹的随机搜索。未达到最优解，陷入局部最优时，适当对其进行调整。

首先需设置一定的交叉和变异概率，通过随机数选择是否进行交叉和变异。对于交叉，本文选择多点交叉算子，即在两个父代染色体上随机设置多个交叉点，然后进行基因互换，由此产生两个新的个体。对于变异，本文选择交换突变算子，在父代染色体上随机选择多个突变点，然后对突变点随机打乱，最后将打乱的突变基因赋予给原来的染色体，以此形成了一条新的变异个体。

STEP7 跨代精英保留

通过交叉和变异产生的子代种群中，父代和子代的精英基因即最优基因合并，然后去选择适应度最高的个体，将其与当前最优个体(精英)进行比较，适应度更高的个体成为新的最优个体(精英)。

STEP8 终止条件

若满足种群进化最大代数(即最大迭代次数)，则算法停止，否则返回 STEP3。

对于问题 1，按照 3.3.1 所述的算法流程进行求解，我们将遗传算法与该问题相结合，进行编程操作。问题 1 是已知最晚结束时间，求最大的单类加工芯片数量，在上述算法的基础上添加一个最外层循环，即设置一个较小的初始芯片数量，然后进行遗传算法求解，当最晚结束时间大于题目中给定时间时，则退出循环，取上一个求解结果作为本题的相对最优调度方案，并得到了最大的加工芯片数量，算法流程如图 6 所示。

3.3.2 问题 1 求解结果及分析

最终求出单位时间内处理 A 类芯片的最大数量 **14**，B 类芯片的最大数量 0。得到单位时间内 A 类芯片的调度结果如附录 A 所示，芯片进入时间调度的甘特图如图 7 所示，由于工序工位(工位)数较多，仅展示前 10 个芯片的调度结果。图中不同序号的芯片用不同颜色表示，并记录了每个芯片对应的工序工位位置。

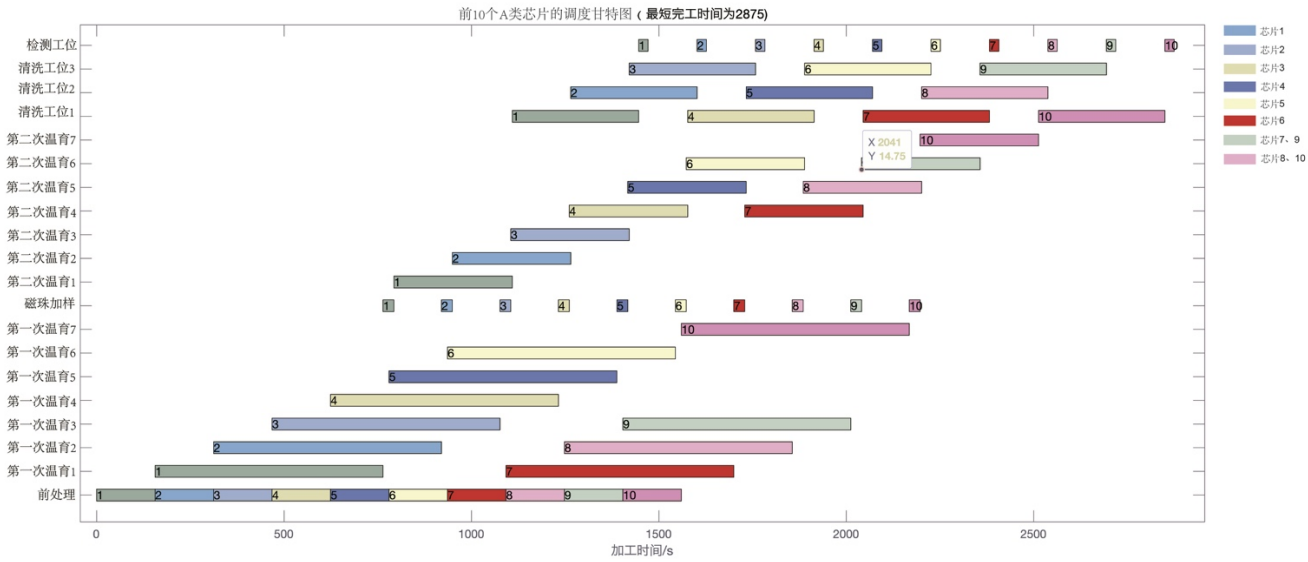

图 7　A 类前 10 个芯片处理流程甘特图

B 类芯片前 2 个芯片处理流程甘特图如下图 7 所示，第一个芯片处理完成的时间已经超过了 4000s，即单位时间内处理的 B 类芯片最大数量为 **0**。

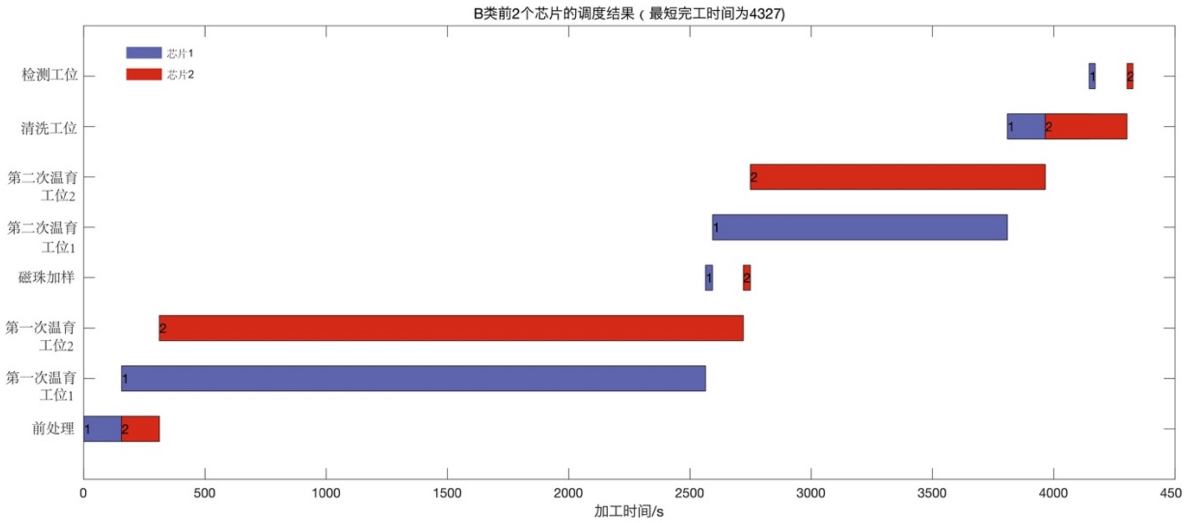

图 8　B 类前 2 个芯片处理流程甘特图

4 问题 2 模型建立与求解

4.1 问题 2 分析

问题二有两组不同数量的芯片，需计算混合处理两组芯片的最短完工时间，记录每道工序开始的时间以及总共花费的时间。问题二在问题一的基础上多处理了一类芯片，需要判断两类芯片投放顺序的先后。可根据上述分析建立数学模型，相较于问题一，模型的目标函数更改为最小化完工时间，模型的总完工时间约束不再需要考虑。芯片类型的投放顺序是影响完工时间的主要因素。

经过分析，问题二可以建立目标函数为最小化处理 A、B 类芯片完工时间的模型，通过前述假设条件建立约束，使用遗传算法对其进行求解。需要满足的硬约束同问题一，时间约束不再需要考虑，同时芯片数量不再是一个变化的值而是一个确定的常量：

(1)唯一性约束：每个芯片在每道工序上的加工只能被分配给一个工位。

(2)分配约束：分配给每道工序内各工位的各类型芯片数之和为需检测芯片数。

(3)工序处理顺序约束：当前工序未加工完成的芯片，下一工序不能开始处理。

(4)芯片处理时间约束：同一工序上，芯片完成时间取决于其在工位上的加工时间和开始时间。

(5)独占性约束：每个工位在同一时间最多仅能处理一个芯片。

(6)共享性约束：第一次温育与第二次温育共用同一大转盘上分布的卡槽。

4.2 模型准备与建立

4.2.1 数据定义

(1)常量

n——*总芯片数量*

q——*芯片类型，* $q \in \{1,2\}$*，分别对应 A、B 类*

n_q——*对应 A、B 类芯片数量*

(2)变量

$p(k, j_q)$——*q 类型芯片 j 在工序 k 上的加工时间*

$S(i_k, j_q)$——*q 类型芯片 j 在工序 k 的工位 i 上的开始加工时刻*

$C(i_k, j_q)$——*q类型芯片j在工序k的工位i上的结束加工时刻*

N_{ki}^{q}——工序k的工位i上加工的q*类型*芯片数量

(3)中间变量

X_{ijk}^{q}——*当q类型的芯片j在工序 k 的工位 i 上进行加工时*$X_{ijk}^{q}=1$*,否则*$X_{ijk}^{q}=0$

4.2.2 问题 2 模型建立

问题 2 需满足的约束同问题 1，相较于模型一，模型二的目标函数变为最小完工时间，即最小化$\mathrm{C}(m_c, n)$，并去除了对总加工时间的时间约束，同时增加了芯片数量约束，即芯片数量不再是一个变化的值而是一个确定的常量。

目标函数：最小化完工时间

$$\mathrm{MinC}(m_c, n)=\max\{C(m_{c-1}, n), C(m_c, n-1)\}+p(c, n) \tag{18}$$

约束条件：

(1)唯一性约束：确保每个芯片在每道工序上的加工只能被分配给一个工位.

$$\sum_{i=1}^{M_k} X_{ijk}^{q}=1, i\in M_k, j\in[1, n], k\in[1, c], q\in 1,2 \tag{19}$$

(2)分配约束：分配给每道工序内各工位的不同类型芯片数之和为需检测芯片数.

$$\sum_{i=1}^{M_k} N_{ki}^{q}=n_q, k\in[1, c], q\in 1,2 \tag{20}$$

(3)工序处理顺序约束，即当前工序未加工完成的芯片，下一工序不能开始处理.

$$C(i_k, j)+\Delta_k \leq S(i_{k+1}, j_q), i\in M_k, j\in[1, n], k\in[1, c], q\in 1,2 \tag{21}$$

(4)芯片处理时间约束：同一工序上，芯片完成时间取决于其在工位上的加工时间和开始时间.

$$C(i_k, j_q)=S(i_k, j_q)+p(k, j_q), i\in M_k, j\in[1, n], k\in[1, c], q\in 1,2 \tag{22}$$

其中，$S(i_k, j_q)$满足取值如下.

$$S(i_k, j_q)=\max C(i_k, j_q-1), C((i-1)_k, j_q) \tag{23}$$

即，芯片j_q在阶段k的开始时间，等于芯片j_q紧前工序的完工时间或同一工位紧前芯片j_q-1的完工时间的最大值。

(5)独占性约束：每个工位在同一时间最多仅能处理一个芯片.

$$\sum_{q=1}^{2}\sum_{j=1}^{n} X_{ijk}^{q}\leq 1, i\in M_k, k\in[1, c] \tag{24}$$

(6)共享性约束：第一次温育与第二次温育共用同一大转盘上分布的卡槽.

$$M_2=M_4 \tag{25}$$

(7)芯片数量关系：芯片总数与各类型芯片数之间的关系.

$$n=\sum_{q=1}^{2} n_q \tag{26}$$

(8)变量取值范围约束.

$$X_{ijk}^{q}=\begin{cases}1, \text{若芯片} i \text{被安排在工序} j \text{的工位 } k \text{上加工}\\ 0, \text{否则}\end{cases}, \forall i, j, k; \tag{27}$$

综上所述，可以建立问题 2 的数学模型。RHFSP 优化模型二的数学描述如下：

MOD2

$$\text{Min}\quad \mathrm{C}(m_c,n)=\max\{C(m_{c-1},n),C(m_c,n-1)\}+p(c,n) \tag{12}$$

$$s.t.\begin{cases}\sum_{i=1}^{M_k}X_{ijk}^{q}=1,i\in M_k,j\in[1,n],k\in[1,c],q\in 1,2\\ \sum_{i=1}^{M_k}N_{ki}^{q}=n_q,k\in[1,c],q\in 1,2\\ C(i_k,j_q)+\Delta_k\leq S(i_{k+1},j_q),i\in M_k,j\in[1,n],k\in[1,c],q\in 1,2\\ C(i_k,j_q)=S(i_k,j_q)+p(k,j_q),i\in M_k,j\in[1,n],k\in[1,c],q\in 1,2\\ S(i_k,j)=\max\{C(i_k,j_q-1),C((i-1)_k,j_q)\},q\in\{1,2\}\\ \sum_{q=1}^{2}\sum_{j=1}^{n}X_{ijk}^{q}\leq 1,i\in M_k,k\in[1,c]\\ M_2=M_4\\ n=\sum_{q=1}^{2}n_q\\ X_{ijk}^{q}=\begin{cases}1,\text{若芯片 i 被安排在工序 j 的工位 k 上加工}\\ 0,\text{否则}\end{cases},\forall i,j,k;q\in 1,2\end{cases} \tag{28}$$

4.3 模型求解与分析

4.3.1 算法设计

问题 2 同样采用遗传算法求解，算法的流程与问题 1 相似，分为个体编码设计、解码设计、初始种群生成、适应度值函数计算、选择算子、交叉算子、变异算子以及精英保留策略。问题 2 是已知加工芯片数量，求最早的多类芯片加工最晚结束时间，在上述算法的基础上，仅需在初始编码时设置某类芯片的序号范围，以及设置不同类别芯片对应不同工序的不同加工时间，再进行求解，最后在最优解中由预定义的序号范围知所属的芯片类别。其中$\mathrm{C}(m_c,n)$为问题 2 的目标函数值。

4.3.2 问题 2 求解结果与分析

问题 2 是已知加工芯片数量，求最早的多类芯片加工最晚结束时间，在上述算法的基础上，仅需在初始编码时设置某类芯片的序号范围，以及设置不同类别芯片对应不同工序的不同加工时间，再进行求解，最后在最优解中由预定义的序号范围知所属的芯片类别。最终求出单位时间内处理 A 类，B 类芯片的最短完工时间为 **27543s**。

对于问题 2 的求解，按照 4.3.1 所述的算法流程进行，我们将遗传算法与该问题相结合，进行编程操作。得到 A 类、B 类芯片各工序开始时间的调度结果如附录 A 所示，芯片进入时间调度的甘特图如图 9 所示，由于工序工位(工位)数较多，仅展示前 10 个芯片的调度结果。图中不同序号的芯片用不同颜色表示，并记录了每个芯片对应的工序工位位置。

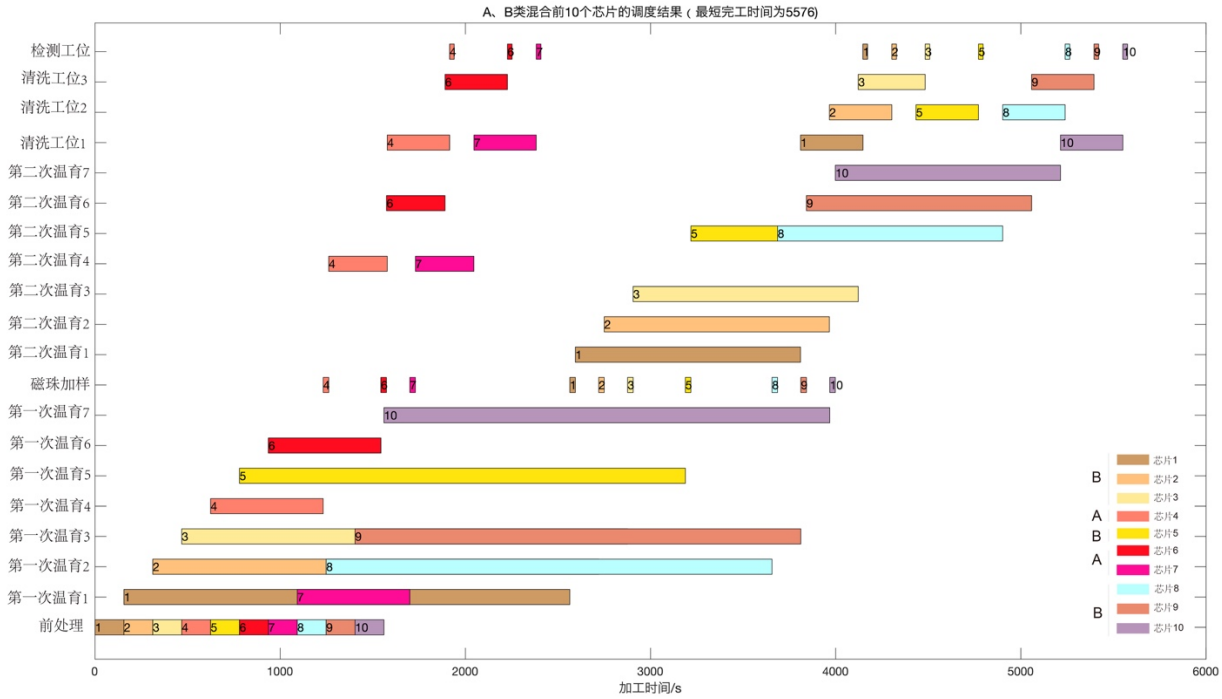

图 9　A、B 类前 10 个芯片处理流程甘特图

遗传算法迭代次数分别设置为 50、100、200，最终 200 次迭代效果最优，其最短完工时间的 100、200 次迭代的收敛曲线如下图所示。

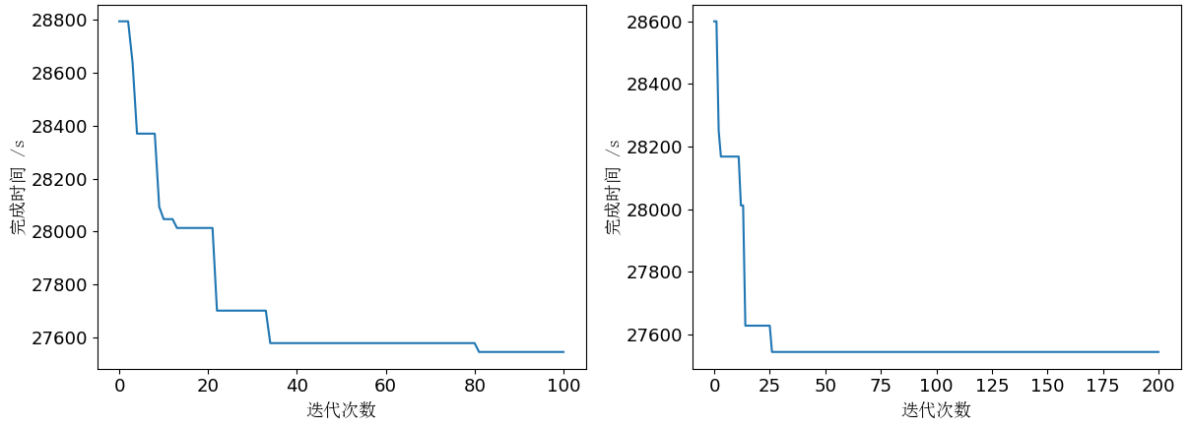

图 10　A、B 类前 10 个芯片最短完工时间的遗传算法收敛曲线

5 问题 3 模型建立与求解

5.1 问题 3 分析

问题三的处理同问题二。问题三需计算 A 类、B 类芯片与 C 类芯片共同通过前处理、第一次温育、磁珠加样、第二次温育、清洗、检测工序，记录每道工序开始的时间以及总共花费的时间，并获取给定芯片数量的最短总完工时间。问题三在问题二的基础上多处理了一类芯片，需要判断三类芯片投放顺序的先后。可根据上述分析建立数学模型。芯片类型的投放顺序仍然是影响完工时间的主要因素。需要满足的硬约束同问题二：

(1)唯一性约束：每个芯片在每道工序上的加工只能被分配给一个工位。

(2)分配约束：分配给每道工序内各工位的各类型芯片数之和为需检测芯片数。

(3)工序处理顺序约束：当前工序未加工完成的芯片，下一工序不能开始处理。

(4)芯片处理时间约束：同一工序上，芯片完成时间取决于其在工位上的加工时间和开始时间。

(5)独占性约束：每个工位在同一时间最多仅能处理一个芯片。

(6)共享性约束：第一次温育与第二次温育共用同一大转盘上分布的卡槽。

5.2 模型准备与建立

5.2.1 数据定义

(1)常量

n——*总芯片数量*

q——*芯片类型，* $q \in \{1,2,3\}$，*分别对应* A、B、C *类*

n_q——*对应* A、B、C *类芯片数量*

5.2.2 问题 3 模型建立

问题 3 需满足的约束同问题 2，相较于模型二，模型三的目标函数中n为三个类型芯片总共的数量，目标函数为最小化$\mathrm{C}(m_c, n)$。

目标函数：最小化完工时间

$$\mathrm{MinC}(m_c, n) = \max\{C(m_{c-1}, n), C(m_c, n-1)\} + p(c, n) \tag{29}$$

约束条件：约束条件同问题二仅改变$q \in \{1,2,3\}$取值范围，其他相同。

(1)唯一性约束：确保每个芯片在每道工序上的加工只能被分配给一个工位.

$$\sum_{i=1}^{M_k} X_{ijk}^{q} = 1, i \in M_k, j \in [1, n], k \in [1, c], q \in 1,2,3 \tag{30}$$

(2)分配约束：分配给每道工序内各工位的不同类型芯片数之和为需检测芯片数.

$$\sum_{i=1}^{M_k} N_{ki}^{q} = n_q, k \in [1, c], q \in 1,2,3 \tag{31}$$

(3)工序处理顺序约束，即当前工序未加工完成的芯片，下一工序不能开始处理.

$$C(i_k, j) + \Delta_k \leq S(i_{k+1}, j_q), i \in M_k, j \in [1, n], k \in [1, c], q \in 1,2,3 \tag{32}$$

(4)芯片处理时间约束：同一工序上，芯片完成时间取决于其在工位上的加工时间和开始时间.

$$C(i_k, j_q) = S(i_k, j_q) + p(k, j_q), i \in M_k, j \in [1, n], k \in [1, c], q \in 1,2,3 \tag{33}$$

其中，$S(i_k, j_q)$满足取值如下.

$$S(i_k, j_q) = \max C(i_k, j_q - 1), C((i-1)_k, j_q) \tag{34}$$

即，芯片j_q在阶段k的开始时间，等于芯片j_q紧前工序的完工时间或同一工位紧前芯片$j_q - 1$的完工时间的最大值。

(5)独占性约束：每个工位在同一时间最多仅能处理一个芯片.

$$\sum_{q=1}^{3} \sum_{j=1}^{n} X_{ijk}^{q} \leq 1, i \in M_k, k \in [1, c] \tag{35}$$

(6)共享性约束：第一次温育与第二次温育共用同一大转盘上分布的卡槽.

$$M_2 = M_4 \tag{36}$$

(7)芯片数量关系：芯片总数与各类型芯片数之间的关系.

$$n=\sum_{q=1}^{3}n_q \tag{37}$$

(8)变量取值范围约束.

$$X_{ijk}^{q}=\begin{cases}1,\text{若芯片}i\text{被安排在工序}j\text{的工位 }k\text{上加工}\\0,\text{否则}\end{cases},\forall i,j,k; \tag{38}$$

综上所述，可以建立问题 3 的数学模型。RHFSP 优化模型三的数学描述如下：

MOD3

$$\mathrm{MinC}(m_c,n)=\max\{C(m_{c-1},n),C(m_c,n-1)\}+p(c,n) \tag{39}$$

$$s.t.\begin{cases}\sum_{i=1}^{M_k}X_{ijk}^{q}=1,i\in M_k,j\in[1,n],k\in[1,c],q\in 1,2,3\\ \sum_{i=1}^{M_k}N_{ki}^{q}=n_q,k\in[1,c],q\in 1,2,3\\ C(i_k,j_q)+\Delta_k\le S(i_{k+1},j_q),i\in M_k,j\in[1,n],k\in[1,c],q\in 1,2,3\\ C(i_k,j_q)=S(i_k,j_q)+p(k,j_q),i\in M_k,j\in[1,n],k\in[1,c],q\in 1,2,3\\ S(i_k,j)=\max\{C(i_k,j_q-1),C((i-1)_k,j_q)\},q\in\{1,2,3\}\\ \sum_{q=1}^{3}\sum_{j=1}^{n}X_{ijk}^{q}\le 1,i\in M_k,k\in[1,c]\\ M_2=M_4\\ n=\sum_{q=1}^{3}n_q\\ X_{ijk}^{q}=\begin{cases}1,\text{若芯片}i\text{被安排在工序}j\text{的工位 }k\text{上加工}\\0,\text{否则}\end{cases},\forall i,j,k;q\in 1,2,3\end{cases} \tag{40}$$

5.3 模型求解与分析

5.3.1 算法设计

问题 3 同样采用遗传算法求解，算法的流程与问题 2 相似，分为个体编码设计、解码设计、初始种群生成、适应度值函数计算、选择算子、交叉算子、变异算子以及精英保留策略。问题 3 同样是已知加工芯片数量，求最早的多类芯片加工最晚结束时间，在上述算法的基础上，仅需在初始编码时设置某类芯片的序号范围，以及设置不同类别芯片对应不同工序的不同加工时间，再进行求解，最后在最优解中由预定义的序号范围知所属的芯片类别。其中$\mathrm{C}(m_c,n)$为问题 3 的目标函数值，其中n为三种类型芯片总共处理的数量。

5.3.2 问题 3 求解结果与分析

问题 3 是已知加工芯片数量，求最早的三类芯片加工最晚结束时间，在问题二算法的基础上，仅需在初始编码时多设置 C 类芯片的序号范围，以及设置不同类别芯片对应不同工序的不同加工时间，再进行求解，最后在最优解中由预定义的序号范围知所属的芯片类别。最终求出单位时间内处理 A 类、B 类、C 类芯片的最短完工时间为 **27230s**。

对于问题 3 的求解，按照 5.3.1 所述的算法流程进行，将遗传算法与该问题相结合，进行编程操作。得到 A 类、B 类、C 类芯片各工序开始时间的调度结果如附录 A 所示，芯片进入时间调度的甘特图如图 11 所示，由于工序工位(工位)数较多，仅展示前 10 个芯片的调度结果。图中不同序号的芯片用不同颜色表示，并记录了每个芯片对应的工序工位位置，右下角为每个芯片对应的颜色及编号、类型。

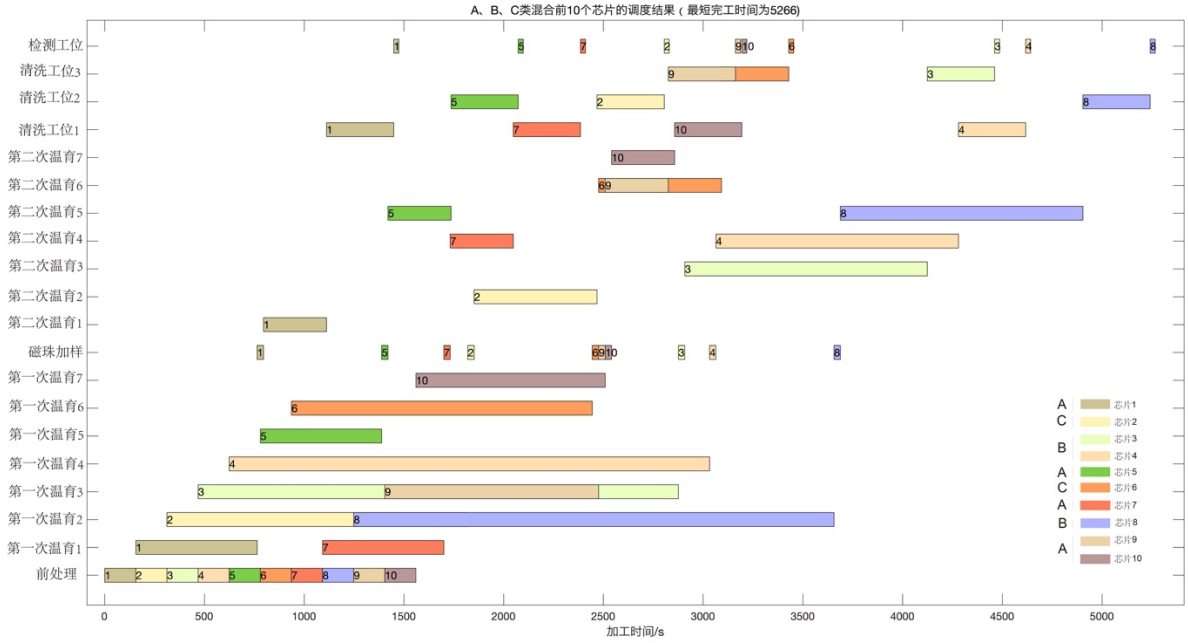

图 11　A、B、C 类前 10 个芯片处理流程甘特图

遗传算法迭代次数分别设置为 100、200，最终 200 次迭代效果最优，其最短完工时间的 100、200 次迭代的收敛曲线如下图所示。

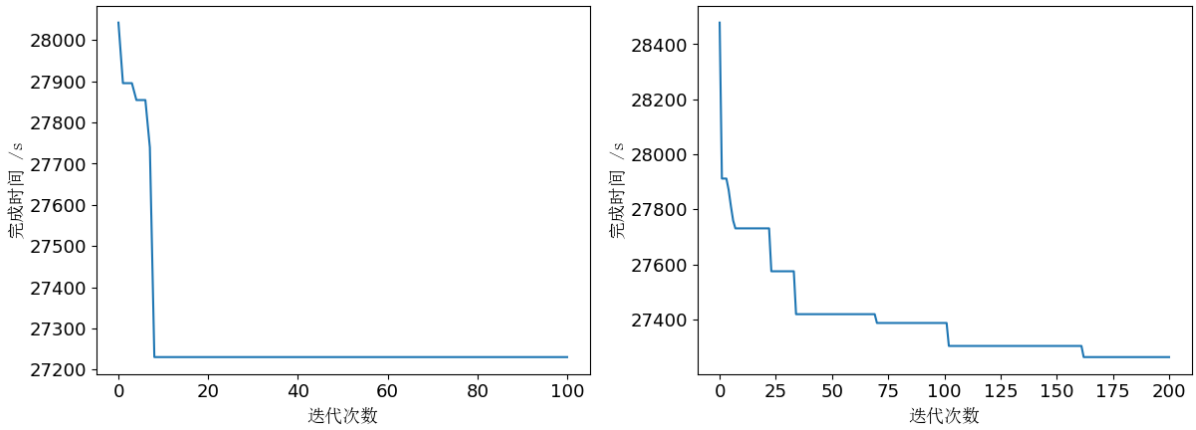

图 12　A、B、C 类前 10 个芯片处理最短完工时间的遗传算法收敛曲线

6 问题 4 分析及模型推广

6.1 问题 4 分析

不失一般性，问题 4 分别使用a_i、b_i代表第一次温育时间(第 2 道工序处理时间)和第二次温育时间(第 4 道工序处理时间)，使用m_i代表各类型芯片数量。针对问题 2 和问题 3 建立的模型 MOD2 和 MOD3，差别仅为q的取值范围不同，即芯片类型种类数的不同。显然，问题 4 是问题 2 与问题 3 的一般情形。

易知，两次温育时间的变化仅会改变$p(k, j_q)$的取值，具体的，若$i = 1,2,3$，则

$$\begin{cases} p(2,j_1)=a_1 \\ p(4,j_1)=b_1 \\ p(2,j_2)=a_2 \\ p(4,j_2)=b_2 \\ p(2,j_3)=a_3 \\ p(4,j_3)=b_3 \end{cases}, \forall j \tag{41}$$

同样满足如下工序处理时间约束及$C(i_k,j_q)$递推关系：

$$C(i_k,j_q)=S(i_k,j_q)+p(k,j_q) \tag{42}$$

$$S(i_k,j_q)=\max C(i_k,j_q-1),C((i-1)_k,j_q) \tag{43}$$

芯片数量m_i的不同仅会影响芯片总数n。经过上述分析，MOD2 和 MOD3 可推广到具有 N 种类型的芯片，芯片数量分别为m_1，$m_2 \ldots m_N$，第一次温育时间分别为a_1,$a_2 \ldots a_N$，第二次温育时间分别为b_1,$b_2 \ldots b_N$倾向下的调度优化问题，同样满足如下硬约束：

(1)唯一性约束

(2)分配约束

(3)工序处理顺序约束

(4)芯片处理时间约束

(5)独占性约束

(6)共享性约束

6.2 模型准备与推广

6.2.1 数据定义

(1)常量

n——*总芯片数量*

q——*芯片类型，*$q \in \{1,2,\ldots,N\}$，*分别对应 A、B 等到 N 类*

题目中所给i与模型设置类型q相同，故$a_i=a_q$为第一次温育时间，b_q为第二次

a_q——第一次温育时间

b_q——第二次温育时间

m_q——各类型芯片数量

6.2.2 问题 4 模型推广

问题 4 需满足的约束同问题 2 和问题 3，仅需改变q的取值范围($q \in [1,N]$)及考虑芯片数量变化对需加工芯片总数n的影响。

目标函数：最小化完工时间

$$\mathrm{MinC}(m_c,n)=\max\{C(m_{c-1},n),C(m_c,n-1)\}+p(c,n) \tag{44}$$

约束条件：

(1)唯一性约束：与问题 2 与问题 3 相比，仅需改变q的取值范围.

$$\sum_{i=1}^{M_k} X_{ijk}^q=1, i \in M_k, j \in [1,n], k \in [1,c], q \in [1,N] \tag{45}$$

(2)分配约束：与问题 2 与问题 3 相比，仅需改变q的取值范围.

$$\sum_{i=1}^{M_k} N_{ki}^{q} = m_q, k \in [1,c], q \in [1,N] \tag{46}$$

(3)工序处理顺序约束：与问题 2 与问题 3 相比，仅需改变q的取值范围.

$$C(i_k,j) + \Delta_k \leq S(i_{k+1},j_q), i \in M_k, j \in [1,n], k \in [1,c], q \in [1,N] \tag{47}$$

(4)芯片处理时间约束：与问题 2 与问题 3 相比，仅需改变q的取值范围.

$$C(i_k,j_q) = S(i_k,j_q) + p(k,j_q), i \in M_k, j \in [1,n], k \in [1,c], q \in [1,N] \tag{48}$$

其中，$S(i_k,j_q)$满足取值如下.

$$S(i_k,j_q) = \max C(i_k,j_q - 1), C((i-1)_k,j_q) \tag{49}$$

(5)独占性约束：与问题 2 与问题 3 相比，仅需改变q的取值范围.

$$\sum_{q=1}^{N}\sum_{j=1}^{n} X_{ijk}^{q} \leq 1, i \in M_k, k \in [1,c] \tag{50}$$

(6)共享性约束：

$$M_2 = M_4 \tag{51}$$

(7)芯片数量关系：仅需改变q的取值范围及各类型芯片数量(常数).

$$n = \sum_{q=1}^{N} m_q \tag{52}$$

(8)变量取值范围约束

$$X_{ijk}^{q} = \begin{cases} 1, \text{若芯片} i \text{被安排在工序} j \text{的工位 } k \text{上加工} \\ 0, \text{否则} \end{cases}, \forall i,j,k; \tag{53}$$

综上，MOD2 和 MOD3 易进行推广，建立问题 4 的普适性模型，见 MOD4。

MOD4

$$\text{Min} \quad C(m_c,n) = \max\{C(m_{c-1},n), C(m_c,n-1)\} + p(c,n)$$

$$s.t. \begin{cases} \sum_{i=1}^{M_k} X_{ijk}^{q} = 1, i \in M_k, j \in [1,n], k \in [1,c], q \in [1,N] \\ \sum_{i=1}^{M_k} N_{ki}^{q} = m_q, k \in [1,c], q \in 1,2 \\ C(i_k,j_q) + \Delta_k \leq S(i_{k+1},j_q), i \in M_k, j \in [1,n], k \in [1,c], q \in [1,N] \\ C(i_k,j_q) = S(i_k,j_q) + p(k,j_q), i \in M_k, j \in [1,n], k \in [1,c], q \in [1,N] \\ S(i_k,j) = \max\{C(i_k,j_q - 1), C((i-1)_k,j_q)\}, q \in [1,N] \\ \sum_{q=1}^{N}\sum_{j=1}^{n} X_{ijk}^{q} \leq 1, i \in M_k, k \in [1,c] \\ M_2 = M_4 \\ n = \sum_{q=1}^{N} m_q \\ X_{ijk}^{q} = \begin{cases} 1, \text{若芯片 i 被安排在工序 j 的工位 k 上加工} \\ 0, \text{否则} \end{cases}, \forall i,j,k; q \in [1,N] \end{cases} \tag{54}$$

6.2.3 算法推广

问题4相较于问题3只将芯片的类型、芯片对应数量进行推广到N类与$\sum_{q=1}^{N} m_q$个。芯片类型不同，仅在初次温育及二次温育工序上时间有所不同，其他基本相同。相较于问题3，仅需要考虑多种类型的芯片能否采用该算法进行求解。该问题中同样采用遗传算法求解，算法的流程与问题 3 相似，分为个体编码设计、解码设计、初始种群生成、适应度值函数计算、选择算子、交叉算子、变异算子以及精英保留策略。问题3同样是已知加工芯片数量，求最早的多类芯片加工最晚结束时间，在上述算法的基础上，仅需在初始阶段使用随机数设置N类芯片的数量、第一次温育时长以及第二次温育时长，然后进行求解，最后在最优解中由预定义的序号范围知所属的芯片类别。其中$\mathrm{C}(m_c, n)$为问题4 的目标函数值，其中n仍为N种类型芯片总共处理的数量。

查阅资料可知，常见的检测芯片类型如表2所示，共有23种类型。

表2 常见抗体芯片类型

细胞因子(Cytokine)	基质金属蛋白酶(Matrix Metalloproteinase)
凋亡因子(Apoptosis)	白介素(Interleukin)
肥胖因子(Adipokine)	磷酸化(Phosphorylation)
血管生成因子(Angiogenesis)	粘附因子(Adhesion Molecule)
炎症因子(Inflammation)	凝集素(Lectin)
凝集素(Lectin)	快速 Ig 分型(Rapid Isotyping)
趋化因子(Chemokine)	胰岛素样生长因子(IGF-1 Receptor)
生长因子(Growth Factor)	糖基化(Glycosylation)
干细胞(Stem Cell)	T 细胞免疫应答(T Cell Response)
可溶性受体(Soluble Receptor)	白介素 1 家族(IL-1 Family)
胃癌标记物(Gastric Cancer Bi-omarker)	骨代谢(Bone Metabolism)
神经因子(Neuro Discovery)	动脉粥样硬化(Atherosclerosis)

在算法推广过程中，我们尝试将芯片类型分别推广至q=5、q=20、q=50种，每种类型的芯片第一次与第二次温育时间的取值范围从题目文件的表1中可获知，分别从0～60min、0～30min 内随机生成，由于时间限制先设置算法处理每种芯片的数量为5，结果显示算法能够照常运行，且得出最短完工结果，说明算法能够较好满足推广要求。

q=5时，芯片进入时间调度的甘特图如图13所示，由于工序工位(工位)数较多，仅展示前 10 个芯片的调度结果。图中不同序号的芯片用不同颜色表示，并记录了每个芯片对应的工序工位位置，右下角为每个芯片对应的颜色及编号、类型。遗传算法 50 次迭代的收敛曲线如图14所示。五个类型的芯片温育时间如表3所示。

表3 芯片类型对应的温育时长

芯片类型及对应数量	**对应的温育时长**
[A, B, C, D, E]	[3048, 606, 2766, 2974, 1099]#第一次温育时长(s)
[41, 41, 48, 35, 40]	[1029, 1304, 1167, 639, 1175]#第二次温育时长(s)

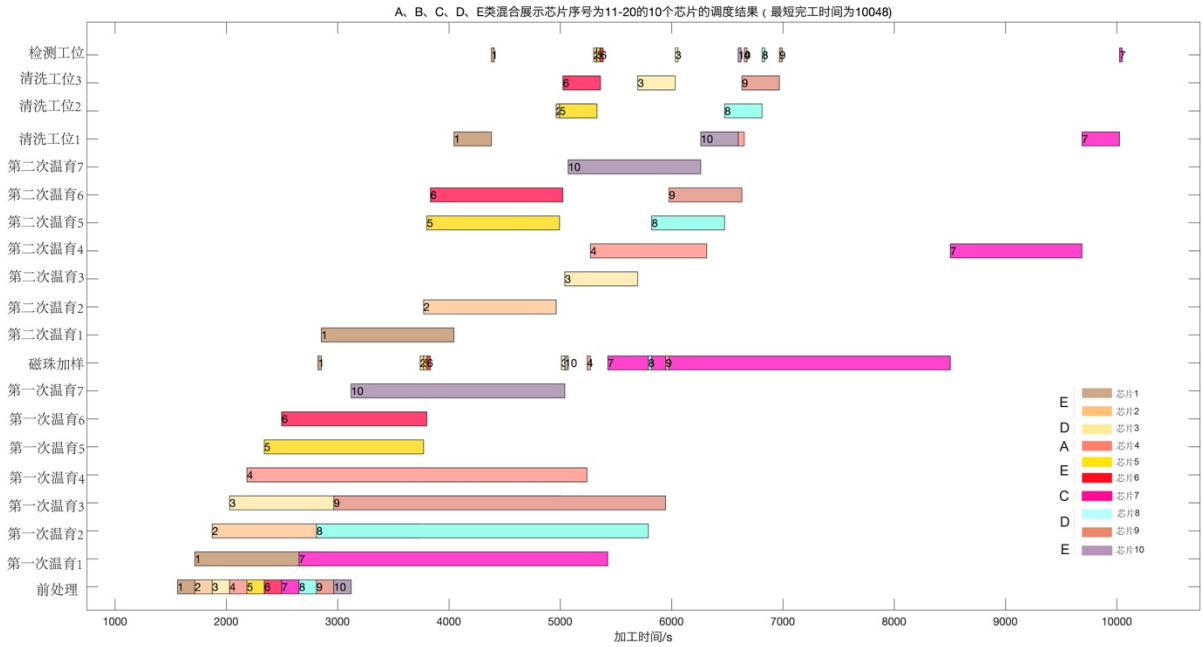

图 13　A、B、C、D、E 类序号为 11-20 的 10 个芯片处理流程甘特图

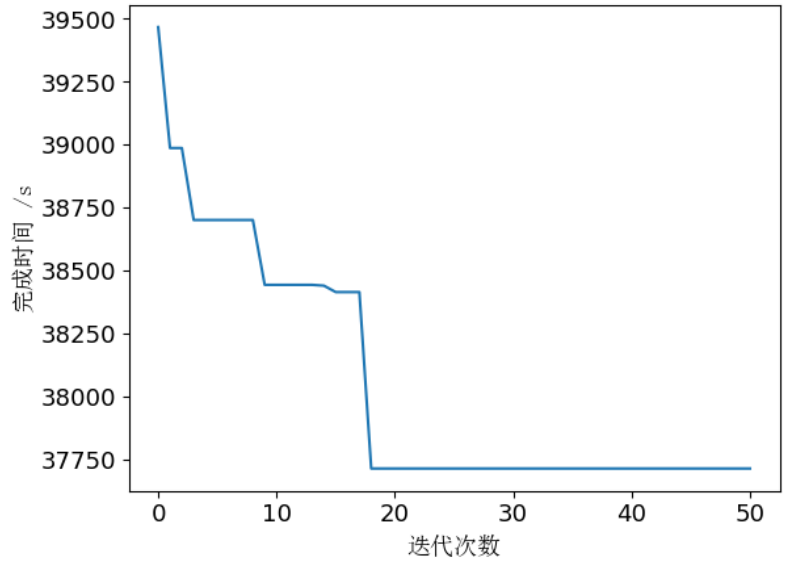

图 14　A、B、C、D、E 类序号为 11-20 的 10 个芯片处理算法收敛曲线图

综上所述，模型与算法在问题二和问题三的基础上，可以分别推广到芯片数量为m_i片，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。进一步推广到 N 种类型的芯片是可行的。

7 问题 5 模型建立与求解

7.1 问题 5 分析

问题五是中途加入第三组芯片，计算处理芯片的最短时间。问题五中需计算 A 类、B 类芯片前 30 分钟内通过检测的芯片总数，后添加 C 类芯片 20 片继续计时。过程中需记录每道工序开始的时间以及总共花费的时间，并获取给定芯片数量的最短总完工时间。问题五在问题三的基础上产生的变化是一类芯片中途加了进来，需要判断前 30 分钟类两类芯片的处理顺序及各芯片的工序开始时间，以及加入 20 片 C 类芯片后，剩余三类

芯片的先后顺序及各芯片的工序开始时间。经过分析，问题五的模型需满足如下硬约束，相较于问题 4 新增了不同芯片类型进仓时间约束：

(1)唯一性约束：每个芯片在每道工序上的加工只能被分配给一个工位。

(2)分配约束：分配给每道工序内各工位的各类型芯片数之和为需检测芯片数。

(3)工序处理顺序约束：当前工序未加工完成的芯片，下一工序不能开始处理。

(4)芯片处理时间约束：同一工序上，芯片完成时间取决于其在工位上的加工时间和开始时间。

(5)独占性约束：每个工位在同一时间最多仅能处理一个芯片。

(6)共享性约束：第一次温育与第二次温育共用同一大转盘上分布的卡槽。

(6)不同芯片类型进仓时间约束：前 30 分钟仅有 A、B 类型的芯片在工位上进行处理，第 30 分钟后 C 类芯片允许开始处理。即 A、B 类型的芯片允许进仓时间设为 0，C 类芯片的允许进仓时间设为 30min。

7.2 模型准备与建立

7.2.1 数据定义

(1)常量

q——*芯片类型，* $q \in 1,2$*，分别对应* A、B、C *类*

n_q——*对应* A、B、C *类芯片数量*

(2)变量

$S(i_k, j_q)$——q*类型芯片*j*在工序*k*的工位*i*上的开始加工时刻*

7.2.2 问题 5 模型建立

问题 5 需满足的约束相较于 MOD3 新增芯片进仓时间约束，模型三的目标函数中n为三个类型芯片总共的数量，目标函数为最小化$\mathrm{C}(m_c, n)$。

目标函数：最小化完工时间

$$\mathrm{MinC}(m_c, n) = \max\{C(m_{c-1}, n), C(m_c, n-1)\} + p(c, n) \tag{55}$$

约束条件：新增不同芯片类型的进仓时间约束，其他与问题三保持一致

(1)不同芯片类型的进仓时间约束：

$$\begin{cases} S(i_1, j_1) = 0 \\ S(i_1, j_2) = 0 \\ S(i_1, j_3) = 1800 \end{cases} \tag{56}$$

(2)芯片处理时间约束：同一工序上，芯片完成时间取决于其在工位上的加工时间和开始时间.

$$C(i_k, j_q) = S(i_k, j_q) + p(k, j_q), i \in M_k, j \in [1, n], k \in [1, c], q \in 1,2,3 \tag{57}$$

其中，$S(i_k, j_q)$满足取值如下.

$$S(i_k, j_q) = \max C(i_k, j_q - 1), C((i-1)_k, j_q), k \in [2, c] \tag{58}$$

即，芯片j_q在阶段k的开始时间，等于芯片j_q紧前工序的完工时间或同一工位紧前芯片$j_q - 1$的完工时间的最大值。

综上所述，可以建立问题 5 的数学模型。RHFSP 优化模型五的数学描述如下：

MOD5

$$\mathrm{MinC}(m_c,n)=\max\{C(m_{c-1},n),C(m_c,n-1)\}+p(c,n) \tag{59}$$

$$s.t.\begin{cases}\sum_{i=1}^{M_k}X_{ijk}^{q}=1,i\in M_k,j\in[1,n],k\in[1,c],q\in 1,2,3\\ \sum_{i=1}^{M_k}N_{ki}^{q}=n_q,k\in[1,c],q\in 1,2,3\\ C(i_k,j_q)+\Delta_k\leq S(i_{k+1},j_q),i\in M_k,j\in[1,n],k\in[1,c],q\in 1,2,3\\ C(i_k,j_q)=S(i_k,j_q)+p(k,j_q),i\in M_k,j\in[1,n],k\in[1,c],q\in 1,2,3\\ S(i_1,j_1)=0;\ S(i_1,j_2)=0;S(i_1,j_3)=1800\\ S(i_k,j)=\max\{C(i_k,j_q-1),C((i-1)_k,j_q)\},k\in[2,c],q\in\{1,2,3\}\\ \sum_{q=1}^{3}\sum_{j=1}^{n}X_{ijk}^{q}\leq 1,i\in M_k,k\in[1,c]\\ M_2=M_4\\ n=\sum_{q=1}^{3}n_q\\ X_{ijk}^{q}=\begin{cases}1,\text{若芯片 i 被安排在工序 j 的工位 k 上加工}\\ 0,\text{否则}\end{cases},\forall i,j,k;q\in 1,2,3\end{cases} \tag{60}$$

7.3 模型求解与分析

7.3.1 算法设计

问题 5 同样采用遗传算法求解，算法的流程与问题 1 相似，分为个体编码设计、解码设计、初始种群生成、适应度值函数计算、选择算子、交叉算子、变异算子以及精英保留策略。问题 5 需在约束模块新增不同芯片类型进仓时间约束，要求 A、B 类芯片的允许开始进仓时间为 0，C 类芯片的允许开始进仓时间为 1800s(30min)，通过在第一阶段遍历每个工件，判断当前完成时间是否小于 1800 和是否为 c 类工件从而调整 c 类工件位置以控制该约束。设置芯片的类型有 3 种，同时在初始编码时设置某类芯片的序号范围，以及设置不同类别芯片对应不同工序的不同加工时间，再进行求解，最后在最优解中由预定义的序号范围知所属的芯片类别。其中$\mathrm{C}(m_c,n)$为问题 5 的目标函数值。

7.3.2 问题 5 求解结果与分析

对于问题 5 的求解，按照 7.3.1 所述的算法流程进行，将遗传算法与该问题相结合，进行编程操作。最终求出处理 A 类、B 类，中途加进 C 类芯片的最短完工时间为 **26760s**。得到 A 类、B 类、C 类芯片各工序开始时间的调度结果如附录 A 所示，芯片进入时间调度的甘特图如图 15 所示，由于工序工位(工位)数较多，仅展示 10 个芯片的调度结果。图中展示的是 A、B 类先进仓、C 类在 30min 开始允许进仓，序号为 11-20 的 10 个芯片处理结果。不同序号的芯片用不同颜色表示，并记录了每个芯片对应的工序工位位置，右下角对芯片的序号、颜色、类型进行展示。

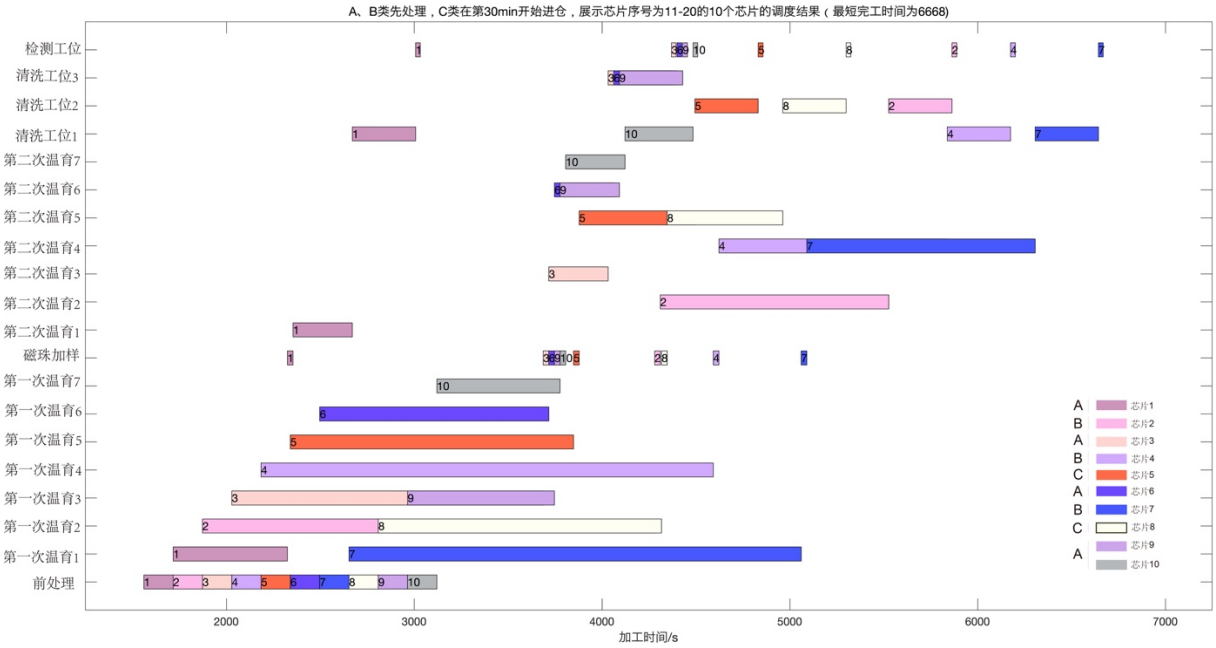

图 15　A、B 类先进仓、C 类在 30min 开始允许进仓，序号为 11-20 的 10 个芯片处理甘特图

8 RHFSP 模型评价与推广

8.1 模型的优点

(1) 本文模型针对化学发光分析仪的运行调度优化，相较于其他 RHFSP 模型，特别考虑了 4 个内容。1.两个不同工序的工位共用。2.工件的类型推广至 N。3.对于同一个工序，不同工件处理时间有差异。4.不同类型的工件进仓时间可以各异；

(2) 本文使用自适应 CHC 遗传算法实现上述的模型求解，精英选择策略选择用跨代策略，结合父代和子代，增加解的判断范围，不易陷入局部最优；重组操作选择异物种重组策略，使不同解个体间基因交叉融合，丰富物种多样性；变异概率选择自适应大变异，使其覆盖面更广泛，提升寻优能力。该算法具有编码解码快，收敛速度快，运行效率高等优点，对于求解 HSFP 模型非常适用；

(3) 模型充分结合实际，简化芯片的填入时间，以及假设每个芯片从上道工序到下道工序所需的对位所用时间都相同，考虑了芯片投放的顺序、芯片投放的开始时间，以及每个工序开始的时间等因素，得到合理的 RHFSP 模型,这样得到的模型贴合实际，具有较高的应用价值，除化学发光免疫分析仪的运行调度外，可以推广到多种可重入混合流水线调度问题，如车间作业调度等其他实际应用问题；

(4) 本文针对问题运用算法求解得到方案完工时间较短，现实使用化学发光免疫分析仪的情况下完全可以按照方案所给的芯片调度顺序及放置时间来进行操作，会大大提高分析仪分析芯片的数量，现有条件下能有效提高芯片的检测效率。

8.2 模型的不足

(5) 实际应用中，化学发光免疫分析仪芯片的放置同样需要消耗时间，且多种类型芯片的混用，处理前排序所需的时间也未在当前问题研究的范围内。本文未能考虑到这些因素的影响，一定程度上影响了模型计算完工时间的准确性；

(6) 本文提出的模型仅针对于化学发光免疫分析仪，由于时间问题没有对其他情况进行检验. 对于其他情形(如其他可重入混合流水线优化问题，可能需要重新对问题进行建模，再用相应算法进行求解。

8.3 模型的推广

1.考虑到现实生活中化学发光免疫分析仪所用芯片的类型不止三类，可以将模型扩充，使更多类型均可适用。2.化学发光免疫分析仪的两次温育转盘是共用的，但是算两道工序，在遗传算法的求解中可以通过存储转盘的空余位置来判断芯片的放入数量。

参考文献

[1] 王丽萍. 两种全自动化学发光免疫分析仪检测甲状腺功能 5 项指标的一致性评价[J].检验医学与临床,2021,18(15):2235-2237. DOI:10.3969/j.issn.1672-9455.2021.15.026.

[2] 赵世安,屈迟文.离散花朵授粉算法求解混合流水线调度问题[J].数学的实践与认识,2018,48(13):182-191.

[3] 崔春红.改进的混合遗传算法求解混合流水车间调度问题[D].内蒙古大学,2006.DOI:10.7666/d.y988314.

[4] 郑堃,练志伟,顾新艳等.采用改进两点交叉算子的改进自适应遗传算法求解不相关并行机混合流水车间调度问题[J].中国机械工程,2023,34(14):1647-1658+1671.

[5] 崔琪,吴秀丽,余建军.变邻域改进遗传算法求解混合流水车间调度问题[J].计算机集成制造系统,2017,23(09):1917-1927.DOI:10.13196/j.cims.2017.09.011.

[6] 董君,叶春明,万孟然.考虑可再生能源的可重入混合流水车间调度问题[J].计算机集成制造系统,2022,28(04):1112-1128.DOI:10.13196/j.cims.2022.04.014.

[7] 程博文,张宏顺.化学发光免疫分析技术的应用进展[J].中国工业医学杂志,2022,35(03):237-240.DOI:10.13631/j.cnki.zggyyx.2022.03.015.

[8] 杨红玮.化学发光免疫分析仪校准和质量控制探讨[J].设备管理与维修,2022(16):5-6.DOI:10.16621/j.cnki.issn1001-0599.2022.08D.03.

[9] 韦润聃, 全自动化学发光免疫分析仪. 四川省,成都宜乐芯生物科技有限公司,2022-04-13.

[10] 张晨磊.化学发光免疫分析法的研究进展[J].医疗装备,2021,34(14):191-192.

[11] 许智伟,吕聪,雷德明.新型教学优化算法可重入混合流水车间调度[J].控制工程,2020,27(10):1812-1819.DOI:10.14107/j.cnki.kzgc.20180478.

[12] 王琨. 面向混合流水车间调度的改进人工蜂群算法研究[D].武汉理工大学,2020.DOI:10.27381/d.cnki.gwlgu.2020.000726.

附录 A:问题调度结果

问题 1. A 型芯片的调度结果(单位：s)

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	156	312	920	949	1265	1602	1627
3	312	468	1076	1105	1421	1758	1783
4	468	624	1232	1261	1577	1914	1939
5	624	780	1388	1417	1733	2070	2095
6	780	936	1544	1573	1889	2226	2251
7	936	1092	1700	1729	2045	2382	2407
8	1092	1248	1856	1885	2201	2538	2563
9	1248	1404	2012	2041	2357	2694	2719
10	1404	1560	2168	2197	2513	2850	2875
11	1560	1716	2324	2353	2669	3006	3031
12	1716	1872	2480	2509	2825	3162	3187
13	1872	2028	2636	2665	2981	3318	3343
14	2028	2184	2792	2821	3137	3474	3499

由表可知 A 类型的芯片在单位时间内完成检测的有 **14** 个。

问题 1. B 型芯片的调度结果(单位：s)

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	156	312	2720	2749	3965	4302	4327

由表可知 B 类型的芯片**没有任何一个**在单位时间内完成检测。

问题 2. A、B 型芯片的调度结果(单位：s)

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	156	312	2720	2749	3965	4302	4327
3	B	312	468	2876	2905	4121	4483	4508
4	A	468	624	1232	1261	1577	1914	1939
5	B	624	780	3188	3217	4433	4770	4795
6	A	780	936	1544	1573	1889	2226	2251
7	A	936	1092	1700	1729	2045	2382	2407
8	B	1092	1248	3656	3685	4901	5238	5263
9	B	1248	1404	3812	3841	5057	5394	5419
10	B	1404	1560	3968	3997	5213	5550	5575
11	B	1560	1716	4124	4153	5369	5731	5756
12	B	1716	1872	4280	4309	5525	5862	5887
13	B	1872	2028	4436	4465	5681	6018	6043
14	A	2028	2184	3685	3715	4031	4368	4393
15	A	2184	2340	3715	3745	4061	4398	4423

16	B	2340	2496	4904	4933	6149	6486	6511
17	B	2496	2652	5060	5089	6305	6642	6667
18	A	2652	2808	3745	3775	4091	4428	4453
19	A	2808	2964	3775	3805	4121	4458	4483
20	B	2964	3120	5528	8285	9501	9838	9863
21	B	3120	3276	5684	10001	11217	11554	11579
22	A	3276	3432	4933	4963	5279	5616	5641
23	B	3432	3588	5996	14213	15429	15766	15791
24	A	3588	3744	4963	4993	5309	5646	5671
25	A	3744	3900	4993	5023	5339	5676	5701
26	A	3900	4056	5023	5053	5369	5706	5731
27	B	4056	4212	6620	6649	7865	8202	8227
28	B	4212	4368	6776	24353	25569	25921	25946
29	A	4368	4524	6025	14681	14997	15334	15359
30	A	4524	4680	6055	17801	18117	18454	18479
31	B	4680	4836	7244	7273	8489	8826	8851
32	B	4836	4992	7400	7429	8645	8982	9007
33	B	4992	5148	7556	7585	8801	9138	9163
34	B	5148	5304	7712	7741	8957	9294	9319
35	A	5304	5460	6085	18269	18585	18943	18968
36	B	5460	5616	8024	14997	16213	16561	16586
37	B	5616	5772	8180	18117	19333	19670	19695
38	B	5772	5928	8336	8365	9581	9918	9943
39	B	5928	6084	8492	8521	9737	10074	10099
40	A	6084	6240	7741	7771	8087	8424	8449
41	A	6240	6396	7771	7801	8117	8454	8479
42	A	6396	6552	7801	7831	8147	8484	8509
43	B	6552	6708	9116	9145	10361	10698	10723
44	B	6708	6864	9272	9301	10517	10854	10879
45	A	6864	7020	7831	7861	8177	8514	8539
46	A	7020	7176	7861	7890	8206	8543	8568
47	A	7176	7332	7940	7969	8285	8622	8647
48	A	7332	7488	8209	18585	18901	19238	19263
49	B	7488	7644	10052	10081	11297	11634	11659
50	B	7644	7800	10208	10237	11453	11790	11815
51	A	7800	7956	9301	9331	9647	9984	10009
52	A	7956	8112	9331	9361	9677	10014	10039
53	A	8112	8268	9361	18901	19217	19554	19579
54	B	8268	8424	10832	10861	12077	12414	12439
55	B	8424	8580	10988	11017	12233	12570	12595
56	B	8580	8736	11144	11173	12389	12726	12751
57	B	8736	8892	11300	11329	12545	12882	12907
58	A	8892	9048	9656	9685	10001	10338	10363
59	B	9048	9204	11612	12545	13761	14107	14132
60	B	9204	9360	11768	13816	15032	15369	15394
61	A	9360	9516	11173	11203	11519	11856	11881
62	B	9516	9672	12080	15977	17193	17530	17555
63	B	9672	9828	12236	17248	18464	18801	18826

64	B	9828	9984	12392	19565	20781	21118	21143
65	B	9984	10140	12570	12600	13816	14153	14178
66	A	10140	10296	11203	11297	11613	11950	11975
67	B	10296	10452	12860	12889	14105	14442	14467
68	B	10452	10608	13016	13045	14261	14598	14623
69	B	10608	10764	13172	13201	14417	14754	14779
70	B	10764	10920	13328	13357	14573	14910	14935
71	A	10920	11076	12421	20813	21129	21487	21512
72	A	11076	11232	12451	22529	22845	23182	23207
73	A	11232	11388	12481	23933	24249	24632	24657
74	B	11388	11544	13952	13981	15197	15534	15559
75	A	11544	11700	12511	25649	25965	26323	26348
76	A	11700	11856	12541	12570	12886	13223	13248
77	B	11856	12012	14420	14449	15665	16002	16027
78	A	12012	12168	13357	13387	13703	14040	14065
79	B	12168	12324	14732	14761	15977	16314	16339
80	B	12324	12480	14888	14917	16133	16470	16495
81	B	12480	12636	15044	19217	20433	20770	20795
82	A	12636	12792	13400	13429	13745	14082	14107
83	B	12792	12948	15356	15385	16601	16938	16963
84	B	12948	13104	15512	15541	16757	17094	17119
85	A	13104	13260	13868	13897	14213	14550	14575
86	B	13260	13416	15824	15853	17069	17406	17431
87	B	13416	13572	16002	16032	17248	17585	17610
88	A	13572	13728	14336	14365	14681	15018	15043
89	A	13728	13884	14492	14521	14837	15174	15199
90	B	13884	14040	16448	25569	26785	27122	27147
91	B	14040	14196	16604	25965	27181	27518	27543
92	B	14196	14352	16760	16789	18005	18342	18367
93	A	14352	14508	15853	15883	16199	16536	16561
94	B	14508	14664	17072	17101	18317	18654	18679
95	A	14664	14820	15883	15913	16229	16586	16611
96	A	14820	14976	15913	21129	21445	21799	21824
97	A	14976	15132	15943	22845	23161	23515	23540
98	A	15132	15288	15973	24249	24582	24919	24944
99	A	15288	15444	17101	17131	17447	17784	17809
100	A	15444	15600	17131	17161	17477	17814	17839
101	B	15600	15756	18164	18193	19409	19746	19771
102	B	15756	15912	18320	18349	19565	19902	19927
103	A	15912	16068	17161	17191	17507	17844	17869
104	A	16068	16224	17191	17221	17537	17874	17899
105	B	16224	16380	18788	18817	20033	20370	20395
106	B	16380	16536	18944	18973	20189	20526	20551
107	B	16536	16692	19100	19333	20549	20886	20911
108	A	16692	16848	17456	17485	17801	18138	18163
109	B	16848	17004	19412	21445	22661	22999	23024
110	B	17004	17160	19568	19597	20813	21150	21175
111	A	17160	17316	17924	17953	18269	18606	18631

112	A	17316	17472	18193	18223	18539	18876	18901
113	A	17472	17628	18236	18265	18581	18918	18943
114	B	17628	17784	20192	20221	21437	21774	21799
115	B	17784	17940	20348	20377	21593	21930	21955
116	B	17940	18096	20504	20533	21749	22086	22111
117	B	18096	18252	20660	20689	21905	22242	22267
118	A	18252	18408	19441	23161	23477	23814	23839
119	A	18408	18564	19471	24565	24881	25231	25256
120	A	18564	18720	19501	19531	19847	20184	20209
121	B	18720	18876	21284	21313	22529	22866	22891
122	B	18876	19032	21440	24881	26097	26434	26459
123	A	19032	19188	20689	20719	21035	21372	21397
124	B	19188	19344	21752	21781	22997	23334	23359
125	B	19344	19500	21908	21937	23153	23490	23515
126	A	19500	19656	20719	20749	21065	21402	21427
127	A	19656	19812	20749	20779	21095	21432	21457
128	A	19812	19968	20779	20809	21125	21462	21487
129	A	19968	20124	20809	23477	23793	24130	24155
130	B	20124	20280	22688	22717	23933	24270	24295
131	B	20280	20436	22844	22873	24089	24426	24451
132	B	20436	20592	23000	23029	24245	24607	24632
133	B	20592	20748	23156	23185	24522	24859	24884
134	A	20748	20904	21937	21967	22283	22620	22645
135	A	20904	21060	21967	21997	22313	22650	22675
136	B	21060	21216	23624	23653	24869	25206	25231
137	B	21216	21372	23780	23809	25025	25362	25387
138	A	21372	21528	22136	22165	22481	22818	22843
139	A	21528	21684	22292	22321	22637	22974	22999
140	A	21684	21840	22448	22477	22793	23130	23155
141	B	21840	21996	24404	24433	25649	26021	26046
142	A	21996	22152	23809	23839	24155	24492	24517
143	B	22152	22308	24716	24745	25961	26298	26323
144	A	22308	22464	23839	23869	24185	24522	24547
145	A	22464	22620	23869	23899	24215	24552	24577
146	B	22620	22776	25184	25213	26429	26766	26791
147	A	22776	22932	23899	23929	24245	24582	24607
148	A	22932	23088	23929	23958	24274	24657	24682
149	A	23088	23244	23958	23988	24426	24763	24788
150	A	23244	23400	24008	24037	24492	24829	24854
151	A	23400	23556	24164	24193	24552	24889	24914
152	A	23556	23712	24320	24349	24665	25002	25027
153	A	23712	23868	25213	25243	25559	25896	25921
154	A	23868	24024	25243	25273	25589	25946	25971
155	A	24024	24180	25273	25303	25619	25971	25996
156	A	24180	24336	25303	25333	25649	25996	26021
157	A	24336	24492	25333	25362	25678	26046	26071
158	A	24492	24648	25362	25392	25708	26071	26096
159	A	24648	24804	25412	25441	25896	26233	26258

160	A	24804	24960	25568	25597	25913	26258	26283

由表可知 A、B 类型的芯片完成检测的最短用时为 **27543**s。

问题 3. A、B、C 型芯片的调度结果(单位：s)

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	A	0	156	764	796	1112	1449	1474
2	C	156	312	1820	1852	2468	2805	2830
3	B	312	468	2876	2908	4124	4461	4486
4	B	468	624	3032	3064	4280	4617	4642
5	A	624	780	1388	1420	1736	2073	2098
6	C	780	936	2444	2476	3092	3429	3454
7	A	936	1092	1700	1732	2048	2385	2410
8	B	1092	1248	3656	3688	4904	5241	5266
9	A	1248	1404	2476	2509	2825	3162	3187
10	A	1404	1560	2509	2541	2857	3194	3219
11	C	1560	1716	3688	3721	4337	4674	4699
12	B	1716	1872	4280	4312	5528	5865	5890
13	A	1872	2028	2636	2668	2984	3321	3346
14	B	2028	2184	4592	7254	8470	8807	8832
15	B	2184	2340	4748	7820	9036	9373	9398
16	C	2340	2496	4312	4345	4961	5298	5323
17	A	2496	2652	4345	4377	4693	5030	5055
18	A	2652	2808	4377	4410	4726	5063	5088
19	B	2808	2964	5372	9380	10596	10946	10971
20	C	2964	3120	4780	8444	9060	9398	9423
21	A	3120	3276	4410	4856	5172	5509	5534
22	A	3276	3432	4442	5480	5796	6133	6158
23	A	3432	3588	4475	5948	6264	6601	6626
24	A	3588	3744	4508	4540	4856	5193	5218
25	C	3744	3900	5408	12188	12804	13141	13166
26	B	3900	4056	6464	19988	21204	21541	21566
27	A	4056	4212	4820	8912	9228	9565	9590
28	B	4212	4368	6776	21860	23076	23413	23438
29	A	4368	4524	5132	5164	5480	5817	5842
30	C	4524	4680	6188	6220	6836	7173	7198
31	B	4680	4836	7244	7276	8492	8832	8857
32	A	4836	4992	5600	5632	5948	6285	6310
33	C	4992	5148	6808	24356	24972	25309	25334
34	A	5148	5304	5912	17492	17808	18145	18170
35	C	5304	5460	7276	7309	7925	8262	8287
36	A	5460	5616	6841	6873	7189	7526	7551
37	B	5616	5772	8180	13232	14448	14785	14810
38	B	5772	5928	8336	17136	18352	18689	18714
39	A	5928	6084	6873	6906	7222	7559	7584
40	A	6084	6240	6906	6938	7254	7591	7616
41	A	6240	6396	7309	7341	7657	7994	8019

42	A	6396	6552	7341	10268	10584	10921	10946
43	B	6552	6708	9116	9148	10364	10701	10726
44	A	6708	6864	7472	7504	7820	8157	8182
45	A	6864	7020	7628	11048	11364	11701	11726
46	A	7020	7176	7784	11828	12144	12481	12506
47	C	7176	7332	8840	22592	23208	23545	23570
48	A	7332	7488	8096	8128	8444	8781	8806
49	A	7488	7644	8368	18224	18540	18895	18920
50	C	7644	7800	9308	9340	9956	10293	10318
51	A	7800	7956	8564	8596	8912	9249	9274
52	C	7956	8112	9620	9652	10268	10605	10630
53	C	8112	8268	9776	9808	10424	10761	10786
54	A	8268	8424	9032	9064	9380	9717	9742
55	B	8424	8580	10988	11020	12236	12573	12598
56	A	8580	8736	9344	9376	9692	10029	10054
57	C	8736	8892	10400	10432	11048	11385	11410
58	B	8892	9048	11456	11488	12704	13041	13066
59	B	9048	9204	11612	17808	19024	19366	19391
60	C	9204	9360	11020	11053	11669	12006	12031
61	B	9360	9516	11924	11956	13172	13509	13534
62	C	9516	9672	11180	11212	11828	12165	12190
63	C	9672	9828	11644	11677	12293	12630	12655
64	A	9828	9984	11053	11085	11401	11738	11763
65	C	9984	10140	11677	11709	12325	12662	12687
66	B	10140	10296	12704	12736	13952	14289	14314
67	C	10296	10452	11960	11992	12608	12945	12970
68	B	10452	10608	13016	13048	14264	14601	14626
69	B	10608	10764	13172	13204	14420	14757	14782
70	B	10764	10920	13328	13360	14576	14913	14938
71	C	10920	11076	12584	12616	13232	13569	13594
72	A	11076	11232	11840	11872	12188	12525	12550
73	B	11232	11388	13796	24972	26315	26669	26694
74	B	11388	11544	13952	13984	15200	15554	15579
75	C	11544	11700	13208	13240	13856	14193	14218
76	B	11700	11856	14264	14296	15512	15849	15874
77	C	11856	12012	14296	14329	14945	15282	15307
78	C	12012	12168	14329	14576	15192	15529	15554
79	A	12168	12324	13240	13273	13589	13926	13951
80	A	12324	12480	13273	18540	18856	19193	19218
81	B	12480	12636	15044	15076	16292	16629	16654
82	A	12636	12792	14361	16292	16608	16945	16970
83	B	12792	12948	15356	21284	22500	22837	22862
84	B	12948	13104	15512	23936	25152	25489	25514
85	B	13104	13260	15668	25652	26868	27205	27230
86	A	13260	13416	14394	19568	19884	20221	20246
87	B	13416	13572	15980	16012	17228	17565	17590
88	C	13572	13728	16012	16045	16661	16998	17023
89	B	13728	13884	16292	16324	17540	17877	17902

90	C	13884	14040	16324	16357	16973	17310	17335
91	B	14040	14196	16604	16636	17852	18189	18214
92	C	14196	14352	16357	23208	23824	24161	24186
93	C	14352	14508	16389	16422	17038	17375	17400
94	C	14508	14664	16422	16454	17070	17407	17432
95	C	14664	14820	16487	16520	17136	17473	17498
96	B	14820	14976	17384	17416	18632	18969	18994
97	C	14976	15132	16640	18856	19472	19809	19834
98	A	15132	15288	16454	16487	16803	17140	17165
99	B	15288	15444	17852	17884	19100	19437	19462
100	C	15444	15600	17108	17140	17756	18093	18118
101	C	15600	15756	17416	17449	18065	18402	18427
102	B	15756	15912	18320	18352	19568	19905	19930
103	C	15912	16068	17576	17608	18224	18561	18586
104	C	16068	16224	17884	17917	18533	18870	18895
105	C	16224	16380	17917	17949	18565	18920	18945
106	A	16380	16536	17144	17176	17492	17829	17854
107	C	16536	16692	18200	19884	20500	20837	20862
108	C	16692	16848	18356	18388	19004	19341	19366
109	A	16848	17004	18232	21204	21520	21857	21882
110	A	17004	17160	18265	23076	23392	23729	23754
111	B	17160	17316	19724	19756	20972	21309	21334
112	B	17316	17472	19880	19912	21128	21465	21490
113	B	17472	17628	20036	20068	21284	21621	21646
114	A	17628	17784	18392	18424	18740	19077	19102
115	A	17784	17940	18548	18580	18896	19233	19258
116	B	17940	18096	20504	20536	21752	22089	22114
117	A	18096	18252	18860	18892	19208	19545	19570
118	A	18252	18408	19016	19048	19364	19701	19726
119	A	18408	18564	19172	19204	19520	19857	19882
120	A	18564	18720	19328	19360	19676	20013	20038
121	B	18720	18876	21284	21316	22532	22869	22894
122	A	18876	19032	19640	19672	19988	20325	20350
123	C	19032	19188	21316	21349	21965	22302	22327
124	B	19188	19344	21752	23824	25040	25377	25402
125	A	19344	19500	21349	21381	21697	22034	22059
126	C	19500	19656	21381	21414	22030	22367	22392
127	A	19656	19812	21414	21520	21836	22173	22198
128	A	19812	19968	21446	23392	23708	24045	24070
129	C	19968	20124	21784	21817	22433	22770	22795
130	B	20124	20280	22688	22720	23936	24273	24298
131	C	20280	20436	21944	21976	22592	22929	22954
132	C	20436	20592	22100	22132	22748	23085	23110
133	B	20592	20748	23156	23188	24404	24741	24766
134	A	20748	20904	21512	21544	21860	22198	22223
135	C	20904	21060	22568	22600	23216	23570	23595
136	C	21060	21216	23188	23221	23837	24186	24211
137	C	21216	21372	23221	23253	23869	24211	24236

138	C	21372	21528	23253	23286	23902	24239	24264
139	B	21528	21684	24092	24124	25340	25677	25702
140	B	21684	21840	24248	24280	25496	25833	25858
141	B	21840	21996	24404	24436	25652	25989	26014
142	B	21996	22152	24560	24592	25808	26145	26170
143	B	22152	22308	24716	24748	25964	26307	26332
144	B	22308	22464	24872	24904	26282	26619	26644
145	B	22464	22620	25028	25060	26347	26694	26719
146	A	22620	22776	23384	23416	23732	24070	24095
147	B	22776	22932	25340	25372	26588	26925	26950
148	C	22932	23088	25372	25405	26021	26382	26407
149	C	23088	23244	25405	25437	26155	26507	26532
150	A	23244	23400	24008	24040	24356	24693	24718
151	C	23400	23556	25437	26188	26804	27141	27166
152	A	23556	23712	24320	24352	24668	25005	25030
153	A	23712	23868	25470	25502	25818	26170	26195
154	C	23868	24024	25532	25564	26301	26644	26669
155	A	24024	24180	25564	25597	25913	26250	26275
156	A	24180	24336	25597	25629	25945	26282	26307
157	A	24336	24492	25629	25662	25978	26332	26357
158	A	24492	24648	25662	25694	26010	26357	26382
159	A	24648	24804	25694	25727	26145	26482	26507
160	A	24804	24960	25727	25760	26250	26587	26612

由表可知 A、B、C 类型的芯片全部完成检测的最短用时为 **27230s**。

问题 5. 先有 A、B 型芯片后有 C 型芯片的调度结果(单位：s)

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	2594	3810	4147	4172
2	B	156	312	2720	2750	3966	4303	4328
3	B	312	468	2876	2906	4122	4459	4484
4	A	468	624	1232	1262	1578	1915	1940
5	A	624	780	1388	1418	1734	2071	2096
6	B	780	936	3344	3374	4590	4927	4952
7	B	936	1092	3500	3530	4746	5083	5108
8	B	1092	1248	3656	3686	4902	5239	5264
9	A	1248	1404	2012	2042	2358	2695	2720
10	A	1404	1560	2168	2198	2514	2851	2876
11	A	1560	1716	2324	2354	2670	3007	3032
12	B	1716	1872	4280	4310	5526	5863	5888
13	A	1872	2028	3686	3716	4032	4369	4394
14	B	2028	2184	4592	4622	5838	6175	6200
15	C	2184	2340	3848	3878	4494	4831	4856
16	A	2340	2496	3716	3746	4062	4399	4424
17	B	2496	2652	5060	5090	6306	6643	6668
18	C	2652	2808	4316	4346	4962	5299	5324
19	A	2808	2964	3746	3776	4092	4429	4454

20	A	2964	3120	3776	3806	4122	4484	4509
21	A	3120	3276	4346	4376	4692	5029	5054
22	C	3276	3432	5090	5120	5736	6073	6098
23	B	3432	3588	5996	6026	7242	7579	7604
24	C	3588	3744	5252	5282	5898	6235	6260
25	C	3744	3900	5408	5438	6054	6391	6416
26	A	3900	4056	5438	7662	7978	8315	8340
27	B	4056	4212	6620	6650	7866	8203	8228
28	B	4212	4368	6776	6806	8022	8359	8384
29	A	4368	4524	5468	10782	11098	11435	11460
30	A	4524	4680	5498	20454	20770	21128	21153
31	A	4680	4836	5528	20922	21238	21575	21600
32	B	4836	4992	7400	7430	8646	8983	9008
33	A	4992	5148	5756	21390	21706	22043	22068
34	B	5148	5304	7712	7742	8958	9295	9320
35	B	5304	5460	7868	7898	9114	9451	9476
36	C	5460	5616	7124	7154	7770	8107	8132
37	B	5616	5772	8180	8210	9426	9763	9788
38	B	5772	5928	8336	8366	9582	9919	9944
39	A	5928	6084	7154	7184	7500	7837	7862
40	B	6084	6240	8648	8678	9894	10231	10256
41	B	6240	6396	8804	8834	10050	10387	10412
42	B	6396	6552	8960	8990	10206	10543	10568
43	A	6552	6708	7316	7346	7662	7999	8024
44	A	6708	6864	7472	7502	7818	8155	8180
45	B	6864	7020	9428	21706	22922	23270	23295
46	B	7020	7176	9584	9614	10830	11167	11192
47	A	7176	7332	8834	8864	9180	9517	9542
48	B	7332	7488	9896	9926	11142	11479	11504
49	A	7488	7644	8864	11098	11414	11751	11776
50	C	7644	7800	9926	9956	10572	10909	10934
51	A	7800	7956	8894	20770	21086	21423	21448
52	B	7956	8112	10520	10550	11766	12103	12128
53	A	8112	8268	8924	21238	21554	21891	21916
54	B	8268	8424	10832	10862	12078	12415	12440
55	B	8424	8580	10988	21554	22770	23107	23132
56	B	8580	8736	11144	11174	12390	12727	12752
57	A	8736	8892	9956	9986	10302	10639	10664
58	C	8892	9048	10556	11414	12030	12367	12392
59	B	9048	9204	11612	11642	12858	13195	13220
60	A	9204	9360	9986	10016	10332	10669	10694
61	B	9360	9516	11924	11954	13170	13507	13532
62	B	9516	9672	12080	12110	13326	13663	13688
63	A	9672	9828	10436	10466	10782	11119	11144
64	B	9828	9984	12392	12422	13638	13975	14000
65	A	9984	10140	10748	21086	21402	21752	21777
66	B	10140	10296	12704	13326	14542	14879	14904
67	B	10296	10452	12860	14574	15790	16127	16152

68	B	10452	10608	13016	15978	17194	17531	17556
69	B	10608	10764	13172	17382	18598	18935	18960
70	B	10764	10920	13328	13358	14574	14911	14936
71	A	10920	11076	11954	11984	12300	12637	12662
72	A	11076	11232	11984	12014	12330	12667	12692
73	B	11232	11388	13796	23934	25150	25487	25512
74	A	11388	11544	13202	18630	18946	19304	19329
75	B	11544	11700	14108	25182	26398	26735	26760
76	A	11700	11856	13232	19993	20309	20654	20679
77	B	11856	12012	14420	14450	15666	16003	16028
78	A	12012	12168	13262	22686	23002	23370	23395
79	B	12168	12324	14732	14762	15978	16315	16340
80	B	12324	12480	14888	14918	16134	16471	16496
81	B	12480	12636	15044	15074	16290	16627	16652
82	A	12636	12792	14450	14480	14796	15133	15158
83	B	12792	12948	15356	15386	16602	16939	16964
84	B	12948	13104	15512	21402	22618	22955	22980
85	A	13104	13260	14480	14510	14826	15163	15188
86	A	13260	13416	14510	14540	14856	15193	15218
87	A	13416	13572	14540	14570	14886	15223	15248
88	B	13572	13728	16136	16166	17382	17719	17744
89	A	13728	13884	14570	14600	14916	15253	15278
90	B	13884	14040	16448	20309	21525	21862	21887
91	A	14040	14196	15542	15572	15888	16225	16250
92	B	14196	14352	16760	23002	24218	24555	24580
93	B	14352	14508	16916	16946	18162	18499	18524
94	B	14508	14664	17072	17102	18318	18655	18680
95	A	14664	14820	15572	15602	15918	16255	16280
96	B	14820	14976	17384	17414	18630	18967	18992
97	A	14976	15132	15740	18946	19283	19641	19666
98	C	15132	15288	17102	17132	17748	18085	18110
99	B	15288	15444	17852	17882	19098	19435	19460
100	A	15444	15600	17132	17162	17478	17815	17840
101	B	15600	15756	18164	18194	19410	19747	19772
102	A	15756	15912	17162	17192	17508	17845	17870
103	B	15912	16068	18476	18506	19722	20059	20084
104	B	16068	16224	18746	18777	19993	20330	20355
105	C	16224	16380	18506	18536	19249	19586	19611
106	B	16380	16536	18944	18974	20190	20527	20552
107	C	16536	16692	18536	18566	19279	19616	19641
108	A	16692	16848	18566	18596	18912	19249	19274
109	A	16848	17004	18596	18626	18942	19279	19304
110	B	17004	17160	19568	19598	20814	21153	21178
111	B	17160	17316	19724	22922	24138	24475	24500
112	A	17316	17472	18626	18656	18972	19329	19354
113	A	17472	17628	18656	18686	19002	19354	19379
114	A	17628	17784	18686	18716	19032	19379	19404
115	A	17784	17940	18716	18746	19062	19404	19429

116	A	17940	18096	19754	19784	20100	20437	20462
117	B	18096	18252	20660	20690	21906	22243	22268
118	C	18252	18408	19916	19946	20562	20899	20924
119	A	18408	18564	19946	19976	20292	20629	20654
120	B	18564	18720	21128	21158	22374	22711	22736
121	A	18720	18876	19976	20006	20322	20679	20704
122	B	18876	19032	21440	21470	22686	23023	23048
123	A	19032	19188	20006	20036	20352	20704	20729
124	B	19188	19344	21752	21782	22998	23345	23370
125	A	19344	19500	20108	20138	20454	20791	20816
126	B	19500	19656	22064	22094	23310	23647	23672
127	A	19656	19812	20420	20450	20766	21103	21128
128	A	19812	19968	20576	20606	20922	21259	21284
129	B	19968	20124	22532	22562	23778	24115	24140
130	B	20124	20280	22688	22718	23934	24271	24296
131	A	20280	20436	21044	21074	21390	21727	21752
132	B	20436	20592	23000	23030	24246	24583	24608
133	B	20592	20748	23156	23186	24402	24739	24764
134	A	20748	20904	22562	22592	22908	23245	23270
135	B	20904	21060	23468	23498	24714	25051	25076
136	A	21060	21216	22592	22622	22938	23295	23320
137	C	21216	21372	23498	23528	24144	24500	24525
138	B	21372	21528	23936	23966	25182	25519	25544
139	A	21528	21684	22622	22652	22968	23320	23345
140	B	21684	21840	24248	24278	25494	25844	25869
141	A	21840	21996	22652	22770	23086	23423	23448
142	A	21996	22152	23528	23558	23874	24211	24236
143	B	22152	22308	24716	24746	25962	26299	26324
144	C	22308	22464	24746	24776	25392	25729	25754
145	C	22464	22620	24776	24806	25422	25759	25784
146	C	22620	22776	24806	24836	25452	25789	25814
147	C	22776	22932	24836	24866	25482	25819	25844
148	C	22932	23088	24866	24896	25512	25869	25894
149	C	23088	23244	24896	24926	25542	25894	25919
150	C	23244	23400	24926	25150	25766	26103	26128

由表可知先有 A 类型芯片，后有 B、C 类型芯片全部完成检测的最短用时为 **26760s**。

附录 B:主要程序

代码环境	操作系统: Windows10 编程语言: Python 3.7.1 (Anaconda Navigator 1.9.2) 编辑器: PyCharm 2022.3 (Professional Edition) 代码详见: Codes.zip

代码清单 1 芯片-工位信息矩阵生成模块

```
def Generate multi(State,Machine,first list,second list,num list):
    Job=sum(num list)
    PT = []
    for i in range(State):
        Si = []  # 第i各加工阶段
        if i==0:#工序0，即前处理+对齐
            t 0 = 156
            for j in range(Machine[i]):
                S0 = [t 0 for k in range(Job)]
                Si.append(S0)
            PT.append(Si)
        elif i==1:#第一次温育+对齐
            for j in range(Machine[i]):
                S0 = []
                for k in range(len(num list)):
                    for  in range(num list[k]):
                        S0.append(first list[k]+8)
                Si.append(S0)
            PT.append(Si)
        elif i==2:#磁珠加样+对齐
            t cizhu=random.uniform(21, 25)+8
            for j in range(Machine[i]):
                S0 = [t cizhu for k in range(Job)]
                Si.append(S0)
            PT.append(Si)
        elif i==3:#第二次温育+对齐
            for j in range(Machine[i]):
                S0 = []
                for k in range(len(num list)):
                    for  in range(num list[k]):
                        S0.append(second list[k]+16)
                Si.append(S0)
            PT.append(Si)
        elif i==4:#清洗+对齐
            t clean = 337
            for j in range(Machine[i]):
                S0 = [t clean for k in range(Job)]
                Si.append(S0)
            PT.append(Si)
        elif i==5:#检测
            t jiance = 25
            for j in range(Machine[i]):
                S0 = [t jiance for k in range(Job)]
                Si.append(S0)
            PT.append(Si)
        else:
            print('error')
    return PT
```

代码清单 2 解码模块

```
class Item:
    def __init__(self):
        self.start = []
        self.end = []
        self._on = []
        self.T = []
        self.last_ot = 0
```

```
            self.L = 0

        def update(self, s, e, on, t):#(s,e,machine,t) (0,5,0,5)
            self.start.append(s)#s为开始时间  max(last_od, last_Md[Machine]) max(上一个)
            self.end.append(e)#e为结束时间 max(last_od, last_Md[Machine]) + M_time[Machine]
即开始+运行
            self._on.append(on)#运行的机器序号
            self.T.append(t)#运行时间
            self.last_ot = e #last_ot为该工件在上一工序的结束时间
            self.L += t#运行总时间


    class Scheduling:
        def __init__(self, J_num, Machine, State, PT):
            self.M = Machine
            self.J_num = J_num
            self.State = State
            self.PT = PT
            self.Create_Job()#生成self.jobs,列表，元素为每个工件的item()类
            self.Create_Machine()#生成self.Machines列表，元素为每个工序的每个机器的item()类
[[item0,item1],[item0,item1]]
            self.fitness = 0

        def Create_Job(self):#为每个工件生成一个类，self.jobs为所有工件类的列表
            self.Jobs = []
            for i in range(self.J_num):
                J = Item()
                self.Jobs.append(J)

        def Create_Machine(self):#为每个工序的并行机器生成一个类
            self.Machines = []
            for i in range(len(self.M)):  # 突出机器的阶段性，即各阶段有哪些机器
                State_i = []
                for j in range(self.M[i]):
                    M = Item()
                    State_i.append(M)
                self.Machines.append(State_i)

        def i_decode(self,i,s):

        def select_m(self,i,s):

        def do(self,i,s,start,end,time,Machine):

        # 每个阶段的解码
        def Stage_Decode(self, CHS, Stage):
            for i in CHS:#[2,1,0]
                last_od = self.Jobs[i].last_ot# self.Jobs[i]
                last_Md = [self.Machines[Stage][M_i].last_ot for M_i in range(self.M[Stage])]
                last_ML = [self.Machines[Stage][M_i].L for M_i in range(self.M[Stage])]
```

```
                M_time = [self.PT[Stage][M_i][i] for M_i in range(self.M[Stage])]
                O_et = [max(last_od, last_Md[_]) + M_time[_] for _ in range(self.M[Stage])]
                if O_et.count(min(O_et)) > 1 and last_ML.count(last_ML) > 1:
                    Machine = random.randint(0, self.M[Stage])
                elif O_et.count(min(O_et)) > 1 and last_ML.count(last_ML) < 1:
                    Machine = last_ML.index(min(last_ML))#
                else:
                    Machine = O_et.index(min(O_et))#
                s, e, t = max(last od, last Md[Machine]), max(last od, last Md[Machine]) + M
time[Machine], M_time[Machine]
                self.Jobs[i].update(s, e, Machine, t)#
                self.Machines[Stage][Machine].update(s, e, i, t)
                if e > self.fitness:
                    self.fitness = e

        # 解码
        def Decode(self, CHS):
            #0
            # for i in CHS:  # [2,1,0]
            #     if self.Machines[0][0].last_ot<1800:
            #         if i>=130:#在前1800遇到了C类，则把该工件放至末尾
            #             CHS.remove(i)
            #             CHS.append(i)
            #         else:
            #             self.i_decode(i,0)
            #     else:
            #         self.i_decode(i,0)
            self.Stage_Decode(CHS,0)
            Job_end = [self.Jobs[i].last_ot for i in range(self.J_num)]
    # 所有工件在上一工序的结束时间
            CHS = sorted(range(len(Job_end)), key=lambda k: Job_end[k], reverse=False)
    # 由小到大排序Job_end [2,0,1]
            #i=1,2,3
            self.three_Decode(CHS)
            Job_end = [self.Jobs[i].last_ot for i in range(self.J_num)]
    # 所有工件在上一工序的结束时间
            CHS = sorted(range(len(Job_end)), key=lambda k: Job_end[k], reverse=False)
    # 由小到大排序Job_end [2,0,1]
            #i=4
            i=4
            self.Stage_Decode(CHS,i)#对CHS=[2,1,0] i=工序号解码
            Job_end = [self.Jobs[i].last_ot for i in range(self.J_num)]
            CHS = sorted(range(len(Job_end)), key=lambda k: Job_end[k], reverse=False)
            i=5
            self.Stage_Decode(CHS,i)#对CHS=[2,1,0] i=工序号解码
            Job_end = [self.Jobs[i].last_ot for i in range(self.J_num)]
            CHS = sorted(range(len(Job_end)), key=lambda k: Job_end[k], reverse=False)

        def three_Decode(self,CHS):
            done_second=0
```

```
            first_end_time=[]
            first_end_item=[]
            cizhu_end_item=[]#存储已经磁珠加样但没有二温的
            #1.从前处理拿一个结束时间最小的工件送到温育1
            i=0
            item=CHS[i]
            e1=self.i_decode(item,1)
            first_end_item.append(item)
            first_end_time.append(e1)
            #2.判断它结束之前还有哪些工件可以进入温育1
            while i<len(CHS)-1:
                i=i+1
                id=CHS[i]
                start,end,time,Machine=self.select_m(id,1)
                if start < e1:
                    self.do(id,1,start,end,time,Machine)
                    first_end_item.append(id)
                    first_end_time.append(end)
                else:
                    break
            #3
            while done_second<len(CHS):
                if len(first_end_time)!=0:
                    #3.1选择最早结束温育1的工件进入磁珠加样
                    index=first_end_time.index(min(first_end_time))
                    item=first_end_item[index]
                    e2=self.i_decode(item,2)
                    cizhu_end_item.append(item)#进入已加样列表
                    del first_end_time[index]
#进入了磁珠，就不要在判断第一次温育哪个结束得早的列表里存记录了
                    del first_end_item[index]
                    #3.2判断磁珠加样结束前是否还可以有其他工件进入温育1，
不用判断是否有其他工件进入磁珠，因为磁珠只有一个卡，现在它自己占着
                    while i<len(CHS):
                        id = CHS[i]
                        start, end, time, Machine = self.select_m(id, 1)
                        if start < e2:
                            self.do(id, 1, start, end, time, Machine)
                            first_end_item.append(id)
                            first_end_time.append(end)
                        else:
                            break
                        i=i+1
                    #3.3item进入温育2
                    e3=self.i_decode(item,3)
                    done_second=done_second+1
                    cizhu_end_item.remove(item)#进入了温育2就删掉
                    #3.4判断温育2结束前，是否还有其他工件可以进入温育1和磁珠
                    while i<len(CHS):#温育1
```

```
                    id=CHS[i]
                    start, end, time, Machine = self.select_m(id, 1)
                    if start < e3:
                        self.do(id, 1, start, end, time, Machine)
                        first_end_item.append(id)
                        first_end_time.append(end)
                    else:
                        break
                    i=i+1

                del_index=[]
                for it in range(len(first_end_time)) :#磁珠
                    if first_end_time[it]<e3:
                        del_index.append(it)
                        self.i_decode(first_end_item[it],2)
                        cizhu_end_item.append(first_end_item[it])
                del_index.sort(reverse=True)
                for d in del_index:
                    del first_end_time[d]
                    del first_end_item[d]
            else:#把未二温的给二温
                for ii in range(len(cizhu_end_item)):
                    self.i_decode(cizhu_end_item[ii],3)
                    done_second=done_second+1

            print(done_second)
```

代码清单 3 遗传算法模块

```
class GA:
    def __init__(self, J_num, State, Machine, PT):
        self.State = State
        self.Machine = Machine
        self.PT = PT
        self.J_num = J_num
        self.Pm = 0.2
        self.Pc = 0.9
        self.Pop_size = 50

    # 随机产生染色体
    def RCH(self):
        Chromo = [i for i in range(self.J_num)]
        random.shuffle(Chromo)
        return Chromo

    # 生成初始种群
    def CHS(self):
```

```
        CHS = []
        for i in range(self.Pop_size):
            CHS.append(self.RCH())
        return CHS

    # 选择
    def Select(self, Fit_value):
        Fit = []
        for i in range(len(Fit_value)):
            fit = 1 / Fit_value[i]
            Fit.append(fit)
        Fit = np.array(Fit)
        idx = np.random.choice(np.arange(len(Fit_value)), size=len(Fit_value), replace=True,
                               p=(Fit) / (Fit.sum()))#([0,1,2,...,99],size=100,p=比例[])
    # fit值越大，抽取的概率越大，从[0,1,2,..,99]随机抽取
        return idx

    # 交叉
    def Crossover(self, CHS1, CHS2):
        T_r = [j for j in range(self.J_num)]
        r = random.randint(2, self.J_num)  # 在区间[1,T0]内产生一个整数r
        random.shuffle(T_r)
        R = T_r[0:r]  # 按照随机数r产生r个互不相等的整数
        # 将父代的染色体复制到子代中去，保持他们的顺序和位置
        H1 = [CHS1[_] for _ in R]
        H2 = [CHS2[_] for _ in R]
        C1 = [_ for _ in CHS1 if _ not in H2]
        C2 = [_ for _ in CHS2 if _ not in H1]
        CHS1, CHS2 = [], []
        k, m = 0, 0
        for i in range(self.J_num):
            if i not in R:
                CHS1.append(C1[k])
                CHS2.append(C2[k])
                k += 1
            else:
                CHS1.append(H2[m])
                CHS2.append(H1[m])
                m += 1
        return CHS1, CHS2

    # 变异
    def Mutation(self, CHS):
        Tr = [i_num for i_num in range(self.J_num)]
        # 机器选择部分
        r = random.randint(1, self.J_num)  # 在变异染色体中选择r个位置
        random.shuffle(Tr)
        T_r = Tr[0:r]
        K = []
        for i in T_r:
```

```
            K.append(CHS[i])
        random.shuffle(K)
        k = 0
        for i in T_r:
            CHS[i] = K[k]
            k += 1
        return CHS

    def main(self):
        BF = []#一个种群中最好的结果
        x = [_ for _ in range(self.Pop_size + 1)]
        C = self.CHS()#生成初始种群
        Fit = []
        for C_i in C:
            s = Sch(self.J_num, self.Machine, self.State, self.PT)#Scheduling类
            s.Decode(C_i)
            Fit.append(s.fitness)
        best_C = None
        best_fit = min(Fit)
        BF.append(best_fit)
        for i in range(self.Pop_size):
            C_id = self.Select(Fit)#返回的id是打乱的个体序号 会有重复
            C = [C[_] for _ in C_id]#id打乱后重新排布的个体
            #交叉和变异，得到新的大小为100的种群
            for Ci in range(len(C)):
                if random.random() < self.Pc:#交叉概率
                    _C = [C[Ci]]
                    CHS1, CHS2 = self.Crossover(C[Ci], random.choice(C))
                    _C.extend([CHS1, CHS2])
                    Fi = []
                    for ic in _C:#_C：第一个为父代，第二个和第三个交叉后的结果
                        s = Sch(self.J_num, self.Machine, self.State, self.PT)
                        s.Decode(ic)
                        Fi.append(s.fitness)
                    C[Ci] = _C[Fi.index(min(Fi))]#选出适应度最小(总时间最小)的新的个体
                    Fit.append(min(Fi))
                elif random.random() < self.Pm:
                    CHS1 = self.Mutation(C[Ci])
                    C[Ci] = CHS1
            Fit = []
            Sc = []
            for C_i in C:
                s = Sch(self.J_num, self.Machine, self.State, self.PT)
                s.Decode(C_i)
                Sc.append(s)
                Fit.append(s.fitness)
            if min(Fit) < best_fit or min(Fit) == best_fit:
                best_fit = min(Fit)
                best_C = Sc[Fit.index(min(Fit))]
            BF.append(best_fit)
```

```
        return best_C,BF,x



if __name__ == "__main__":
    first_list=[]
    second_list=[]
    num_list=[]
    for i in range(5):
        first_list.append(random.randint(600,3600))
        second_list.append(random.randint(300,1800))
        num_list.append(random.randint(30,50))
    Job=sum(num_list)
    #Job = 160#工件数
    State = 6 #工序数
    Machine =[1,40,1,40,8,1]#每个工序对应的机器数
    # State = 5 #工序数
    # Machine =[1,3,1,3,1]#每个工序对应的机器数
    PT = Generate_multi(State, Machine,first_list,second_list,num_list)
    g = GA(Job, State, Machine, PT)
    b_c, BF, x = g.main()
    plt.plot(x, BF)
    font1 = {'family': 'SimSun',
             'weight': 'normal',
             'size': 13,
             }  # Arial是字体形式
    plt.ylabel('完成时间 /s', font1)  # x轴坐标名称及字体样式
    plt.xlabel('迭代次数', font1)  # y轴坐标名称及字体样式
    plt.xticks(fontsize=13)  # x轴刻度字体大小
    plt.yticks(fontsize=13)  # y轴刻度字体大小
    plt.savefig('results/5_ABC_epoch50.png', bbox_inches='tight')
    plt.show()
    print(first_list)
    print(second_list)
    print(num_list)
    #b_c.Gantt()
    b_c.printinfo()
```