


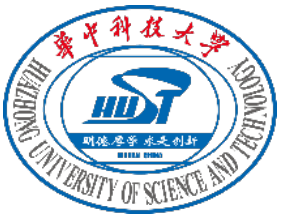

中国研究生创新实践系列大赛

中国光谷·“华为杯”第十九届中国研究生数学建模竞赛

学　　校	中南大学
参赛队号	C22105330245
队员姓名	1.史瞻铭
	2.汪梦祥
	3.黄小彦

中国研究生创新实践系列大赛

中国光谷·“华为杯”第十九届中国研究生数学建模竞赛

题　目　　PBS 调度优化问题建模

摘　　要：

涂装-总装缓存调序区（PBS）的调度优化问题是典型的**混流排序问题**，高效的调度算法对提高汽车生产效率，降低生产成本具有重要意义。本文针对多车辆在 PBS 多车道情况下的调度问题，基于**逆向求解和动态规划**的思想，以**最优出车序列和最短调度时间**为目标，建立了一种 PBS 调度优化模型，并使用**模拟退火算法和遗传算法**对模型进行求解。

针对问题一：对于优化目标 1-4 进行分析，考虑约束 1-12 的影响，本文建立了**出车顺序、动力类型、驱动类型**等指标，将 PBS 调度优化问题简化为**排序问题和分配问题**，随后基于逆向求解和动态规划的思想，建立了一种 PBS 调度优化模型。该模型充分考虑了进车顺序、车道选择的影响。本文综合使用了模拟退火和遗传算法对该模型进行求解，并**对比了两种算法**的优劣，发现模拟退火算法更适合用于**脱敏的大数据集的建模**。具体求解结果如下：采用模拟退火算法对附件一、二进行求解，得到的**序列-时间联合分布矩阵**分别列于**表 5-2 和表 5-3**，**最终得分**分别为 **32.086 分和 63.720 分**。

- 优化目标 1、2：分别以汽车的**动力和驱动类型为排序依据**，受数据影响较大，附件二的数据可以分别相对**提高 50.00%-50.32%和 20.00%-33.34%**的分数。
- 优化目标 3、4：分别以**最少返回道的使用**和**最短调度时间为目标**，本文在分配车道时，进行**逆向求解**，以最优出车顺序为标准，建立**组别优先级**，组内车辆越多优先选择通过时间短的车道，相较未考虑出车顺序的通过时间可以**减少 61.43%-67.48%，**返回道使用次数减至 **4 次，总得分提高 53.94-55.81 分**。

针对问题二:去除了约束 6 和 7，意味着**送车横机可以选择性的接车**，进一步减少了**进车顺序**对**车道选择**的影响。在实现优化目标下的最优出车序列时，可以更少的使用返回道，调度完成时间更短。结果表明：采用模拟退火算法对附件一、附件二进行求解得到的序列-时间联合分布矩阵分别列于**表 6-1 和表 6-2**，**最终得分**分别为 **33.646 分和 64.972 分**。

相比问题一的计算结果，问题二中**返回道使用次数**可以优化为 **0** 次，**最短调度时间减少 3.51%-12.22%，总得分提高 1.93%-4.64%**。

最后，我们通过**自定义最优情况数据集**对模型的有效性及适用性进行验证，**模型得分 99.367/100**。并对提出的模型进行全面的评价：本文建立的模型能够合理解决 PBS 的调度优化问题，能够良好适用于**大数据集**的情况，且模型计算的结果准确有效。

关键词：PBS；调度优化模型；混流排序；模拟退火算法；遗传算法

目录

一、 问题重述

1.1 问题背景

随着社会的发展，人们对于个性化和定制化产品的需求正在提升，我国也顺应科技的进步发布了《中国制造 2025》，强调突出制造业的创新能力、信息化能力与原有的工业化制造能力深度结合。汽车制造业作为传统制造业的代表，时代的发展和消费者的需求都在促使着汽车制造业发生变革，汽车制造厂商也开始转型，提升产品的多样化，但也正面临着生产周期变长、成本提高的风险。并且汽车装配各阶段工艺复杂，物料种类涉及的范围广，要制定合理且有效的生产计划极为困难。企业必须要改进自身的生产管理模式以适应新的发展，实行以客户为导向的生产和销售策略的思想随之产生，要求汽车厂商在同一生产多种型号和款式的车，这使得产品生产的序列排序难度急剧上升[1]。在激烈的市场环境中，中国汽车作为世界汽车行业重要的组成部分，汽车制造企业需提升自身的竞争力，通过优化生产和管理升级达到可持续发展这一目的。

汽车在混合生产线中需经过焊装车间、涂装车间、总装车间，每个车间又包含多个工艺的加工线，存在不同的生产偏好，所以合理的投产序列就显得尤为重要，在 1961 年，Kilbridge 和 Wester 首次提出了混装产品排序问题[2]，即对每个生产周期内的汽车投产序列规划后得出最优生产方案。并且由于各车间的约束不同，各车间无法按照同一序列连续生产，相邻两个车间通常需要通过重排序来改变产品序列[3]，这就需要在两个车间之间建立一个具有调存功能的缓存区（Painted Body Store，汽车制造涂装-总装缓存调序区 PBS），车辆从上游车间离开后不直接进入下游车间，而是经过调整去满足下游车间的约束，再进入到下游车间继续生产。

1.2 问题提出

基于上述的研究背景，汽车的涂装-总装缓存调序区（PBS）的进出车流程如下图 1.1 所示。

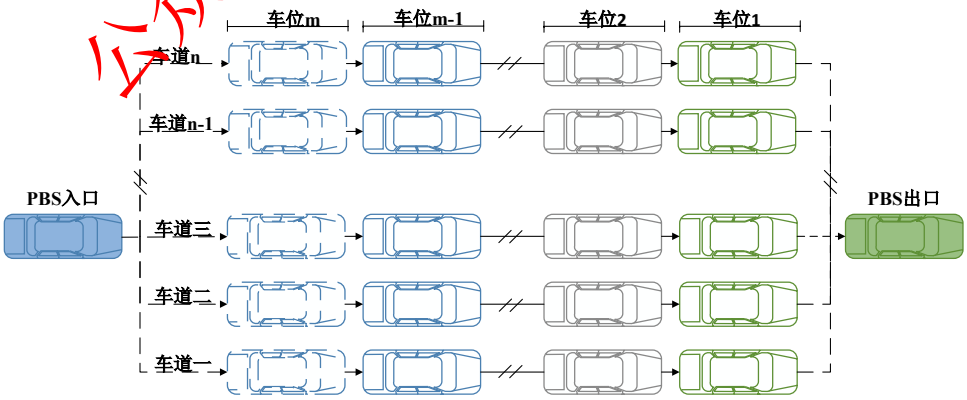

图 1.1 涂装-总装缓存调序区（PBS）示意图

PBS 的 7 个相关时间数据说明，①车身由横移机卸载到各进车道 10 停车位或返回道 1 停车位，以及由各进车道 1 停车位或返回道 10 停车位装载到横移机上的时间均不考虑；②车身在涂装-PBS 出车口处装载到接车横移机上，以及在 4 车道出口中间位置处，车身由送车横移机上卸载到 PBS-总装接车口的时间均不考虑；③在进车过程中（1-6 进车道），任意车身到达涂装-PBS 出车口时，处于最中间位置，正对进车道 4，接车横移机把车身从该位置运送至不同进车道 10 停车位并返回至初始位置，消耗时间分别为[18, 12, 6, 0, 12, 18]

秒；④在出车过程中（1-6 进车道），送车横移机从中间初始位置出发，把各进车道 1 停车位的车身运送至总装-PBS 接车口，处于最中间位置，正对进车道 4，消耗时间分别为[18, 12, 6, 0, 12, 18]秒；⑤在车进返回道过程中（1-6 进车道），在有需要时，送车横移机从中间初始位置出发，把各进车道 1 停车位的车身，运送至返回道 1 停车位，并返回至初始位置，消耗时间分别为[24, 18, 12, 6, 12, 18]秒；⑥在车出返回道过程中（1-6 进车道），接车横移机从中间初始位置出发，把返回道 10 停车位的车身运送至任意进车道 10 停车位并返回至初始位置，消耗时间分别为[24, 18, 12, 6, 12, 18]秒；⑦在车移动过程中，任意车身在进或返回道中，从某一停车位移动至后一停车位，消耗时间为 9 秒。

基于上述研究背景和题目所给时间数据，需解决以下 2 个问题：

(1) **问题 1：优化调度方案一**

严格按照 PBS 约束说明及相关时间数据说明，根据涂装出车序列，考虑 PBS 区域调度能力及限制，建立 PBS 优化调度模型，使得总装进车序列尽可能满足总装生产需求，根据附件 1 和 2 的数据输出优化调度方案和得分结果。

(2) **问题 2：优化调度方案二**

如果去除 PBS 约束说明中第 6、7 两条约束，其余约束不变，根据涂装出车序列，考虑 PBS 区域调度能力及限制，建立 PBS 优化调度模型，使得总装进车序列尽可能满足总装生产需求，根据附件 1 和 2 的数据输出优化调度方案和得分结果。

二、 模型假设与符号说明

2.1 模型基本假设

根据题目已有约束条件和本文对题意的解读，可做出以下假设：

(1) 假设数据对各种变量划分的标准是可靠的；
(2) 假设个别数据的缺失不会对计算结果造成大的影响；
(3) 假设车身在移动过程中，未完全进入下一车位时，释放当前车位；
(4) 当各车道 1 号停车位有车时，接车横移机可选择性接车；
(5) 送车横移机不能将返回道的车身送入 PBS-总装接车口；
(6) 车身在进车道和返回道的移动方向不得改变；
(7) 接车横移机和送车横移机上同一时刻分别最多有一个车身；
(8) 接车横移机和送车横移机在完成任意动作后必须返回中间初始位置，才可以执行下一步动作；
(9) 接车横移机和送车横移机在执行过程中均不能被打断；
(10) 当返回道 10 停车位有车身，同时接车横移机空闲时，优先处理返回道 10 停车位上的车身；
(11) 当若干进车道 1 停车位有车身等候，同时送车横移机空闲时，优先处理最先到达 1 停车位的车身；
(12) 如果任意进车道 1 停车位有车身，那么送车横移机不能为空闲状态；
(13) 进车道和返回道每个时刻最多容纳 10 个车身，每个停车位最多容纳 1 个车身；
(14) 同一车道内，多个车身在不同停车位上的移动可以不同步进行；
(15) 当某车身所在停车位的下一停车位出现空位时，车身必须立即开始向下一停车位移动；
(16) 车身在进车道和返回道不同停车位之间移动的过程中，不能被调度；
(17) PBS 出车序列倾向于从小到大排布；

(18) 在遵循最优出车序列的前提下，车身在 PBS 的进车和出车过程倾向于进入耗时最短的车道；
(19) 为简化模型，假设各进车道上的车身到达 1 号停车位后都需要等待一段时间。

2.2 符号说明

本文定义了如下 32 个使用次数较多的符号，其余符号在使用时注明。

表 2-1 符号说明

序号	符号	含义
1	A	A型车
2	B	B型车
3	C	进、出车队列长度
4	D	表示返回车道使用次数
5	F	目标函数
6	h	混动车身
7	i,j	每辆车在进站序列中的位置
8	k	每辆车在出站序列中的位置
9	m	PBS车道数
10	n	每条车道的车位数
11	q	权重
12	r	燃油车身
13	S	车辆总数
14	T	调度完成时间
15	u	两驱车
16	v	四驱车
17	W	双射函数
18	π	PBS进车序列
19	π'	PBS出车序列
20	GA	遗传算法
21	SA	模拟退火算法
22	e_{ki}	0-1 变量，e_{ki}=1 表示进站车辆 i 被分配到出站序列 k 的位置，反之 e_{ki}=0
23	z_{ij}	0-1 变量，z_{ij} =1 表示 PBS 中车辆 i,j 前后相邻排布，反之 z_{ij} =0
24	p_{ij}	PBS 中车辆 i,j 前后相邻排布时，车辆 i 后同一车道内的车辆数
25	φ_{hr}	0-1 变量，φ_{hr} =1 表示每连续两辆 h 之间的 S_r=2，反之φ_{hr} =0
26	φ_{uv}	0-1 变量，φ_{uv} =1 表示每一分块中 $S_u/S_v\neq1$，反之φ_{uv} =0
27	T_0	模拟退火算法初始温度

28	k	退火速率
29	N_{max}	算法终止条件，外层循环的最大迭代次数
30	L	每个温度下，内层循环的最大迭代次数
31	d	外层循环的第 d 次迭代
32	f	优化目标得分（f_1、f_2、f_3、f_4）

三、 解题流程图

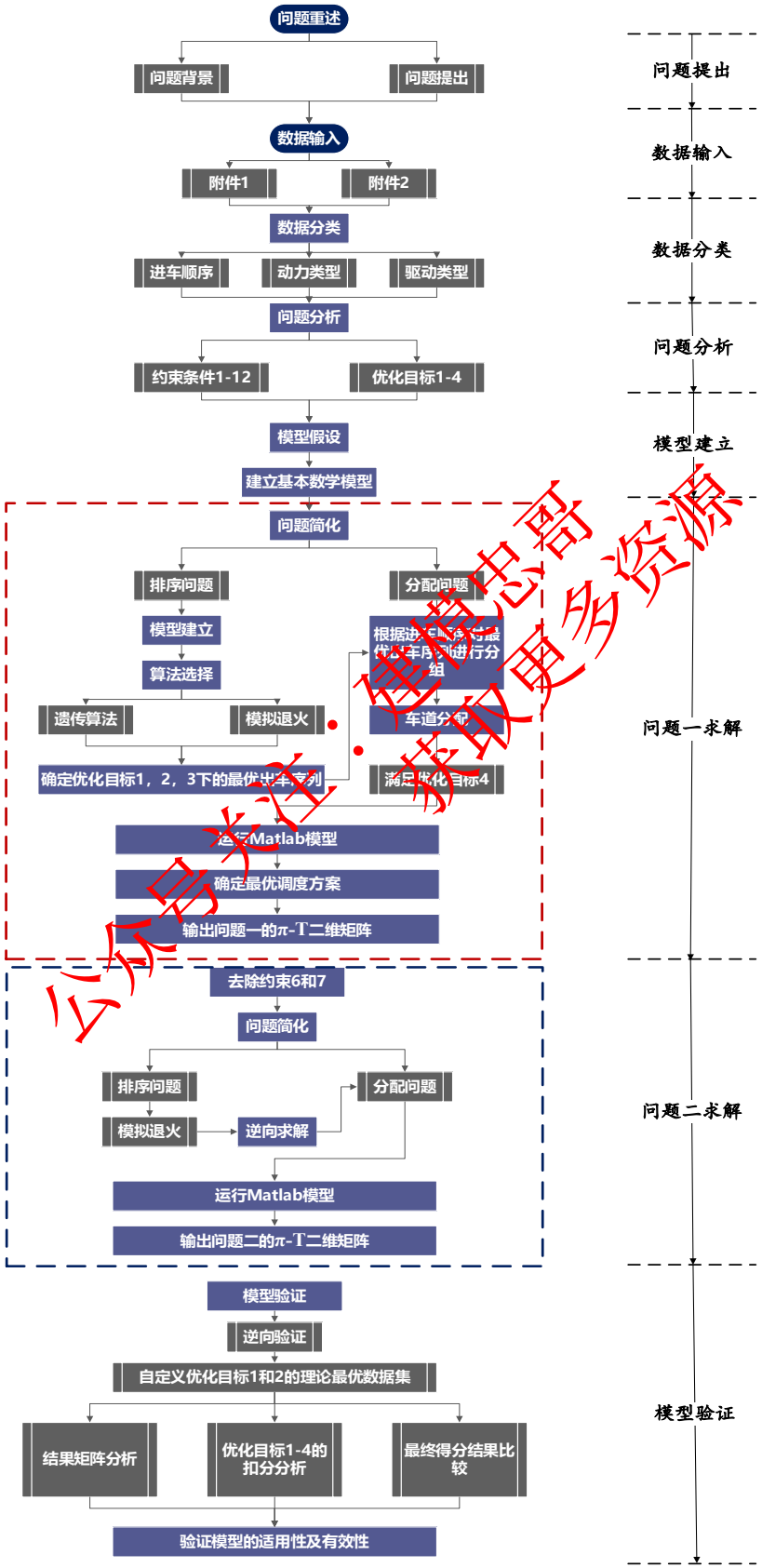

图 3.1 本文技术路线图

四、　优化目标分析与模型建立

4.1 优化目标分析

在满足已有约束条件的基础上，需满足以下四个优化目标，并在加权后，建立模型具体量化目标，求解模型使得分尽可能的高，各优化目标权重系数分别为 0.4、0.3、0.2、0.1。

（1） 优化目标 1：在出车序列中，混动车型间隔 2 台非混车型为优，若间隔不等于 2 则扣 1 分。据题意，如下图 4.1、4.2 所示，当下列情况存在，进车序列如下图的情况，则经过重排序后，存在唯一最优解。

① 当燃油车数大于或等于混动车数时，最优出站序列为：

$$S_r \geq 2\times(S_h-1) \tag{4-1}$$

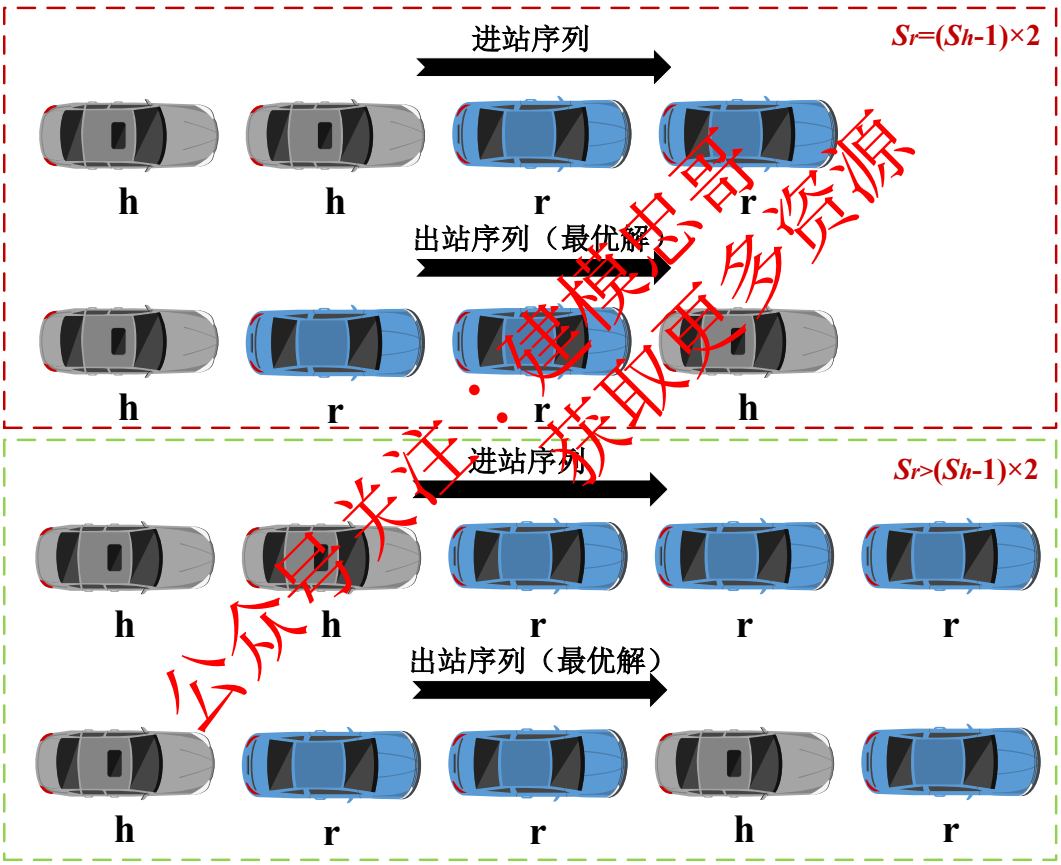

图 4.1 存在唯一最优解情况

② 当燃油车数小于混动车数时，最优出站序列为：

$$S_r < 2\times(S_h-1) \tag{4-2}$$

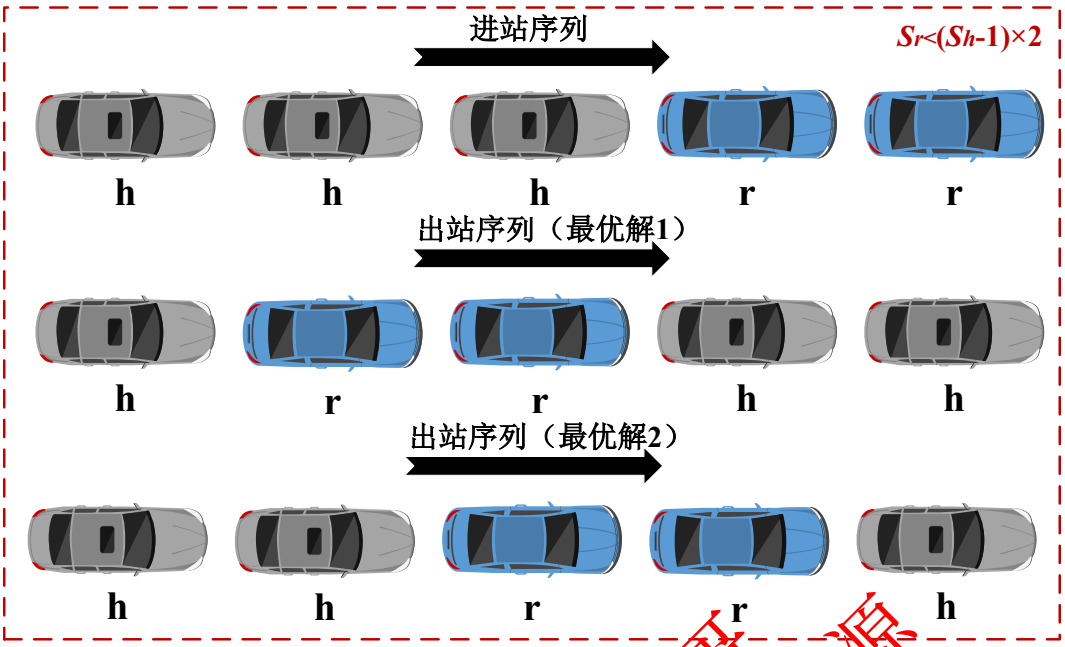

图 4.2 存在多种最优解情况

（2） 优化目标 2：对出车序列进行分块，每一分块中的四驱车型与两驱车型倾向于 1:1 出车序列，如果序列以 *v* 开头，则根据从 *u* 变为 *v* 将序列进行分块；如果序列以 *u* 开头，则根据从 *v* 变为 *u* 将序列进行分块，若分块不满足 1:1，则扣 2 分。

① 若出车序列中两驱数量多于四驱数量，则可按图 4.3 排序方式得出最优解：

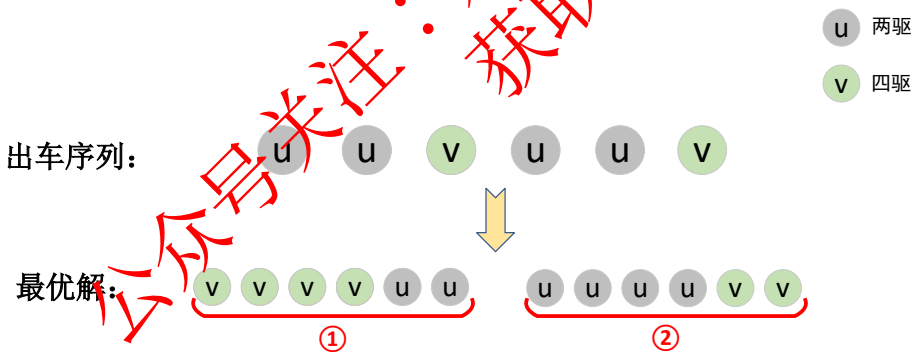

图 4.3 两驱数量多于四驱数量

② 若出车序列中两驱数量少于四驱数量，则可按图 4.4 排序方式得出最优解：

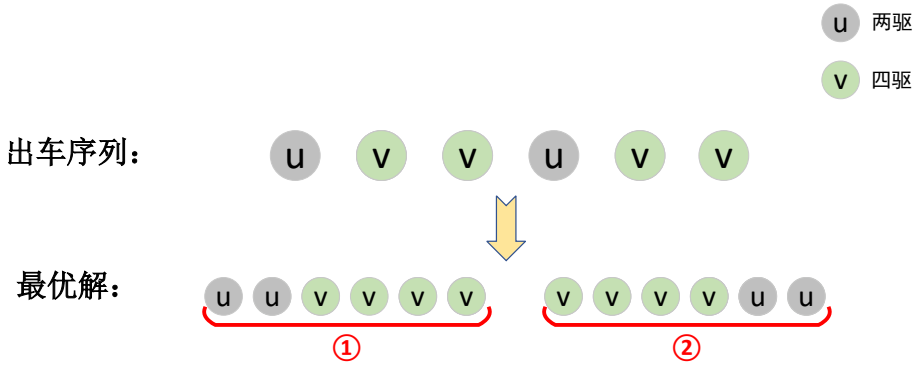

图 4.4 两驱数量少于四驱数量

③ 若出车序列中两驱数量等四驱数量，则可按图 4.5 排序方式得出最优解：

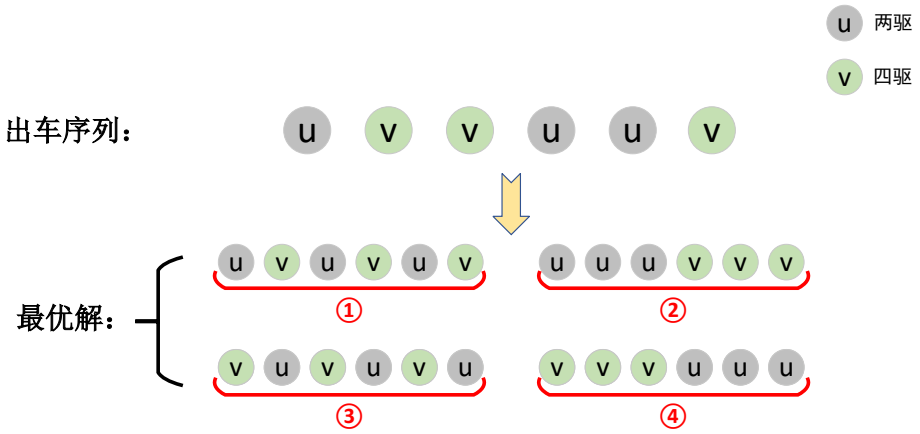

图 4.5 两驱数量等于四驱数量

（3）优化目标 3：尽量不使用返回道，每使用一次扣 1 分。

（4）优化目标 4：倾向总调度时间越短越好。当全部车都从 4 车道进入时，此时的出车时间是最短的，理论最快完成时间为 9C+72。

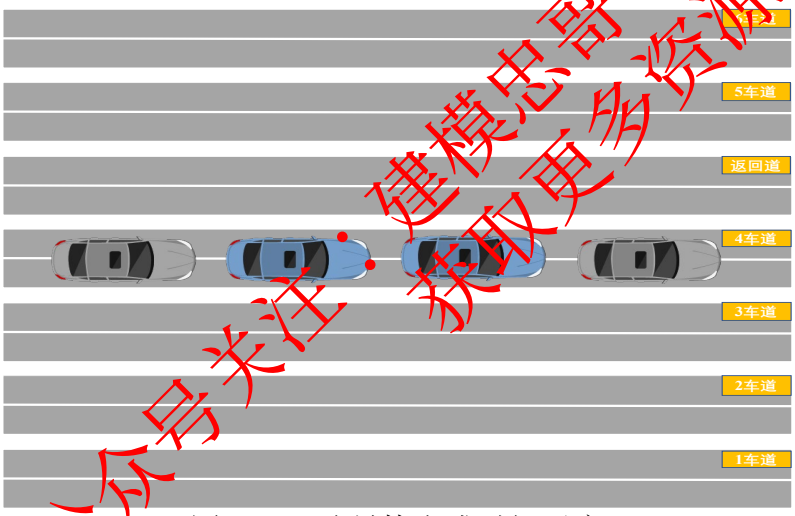

图 4.6 理论最快完成时间示意

4.2 数学模型

序列重排之后进出车辆之间的对应关系可以建立模型来表达，前文中已将模型中相关符号进行定义，现将题目给出的数据表中的进车顺序 C、车型（A、B）、动力（h、r）、驱动（u、v）转换为模型输入形式，如表 4-1 所示。

表 4-1 模型输入

进车顺序	车型	动力	驱动
1	A	h	u
2	B	h	u
3	A	r	u
…	…	…	…
C-2	B	r	u
C-1	A	r	v
C	B	r	v

根据问题描述和表 2-1 内给出的参数变量符号，进车序列和出车序列在 PBS 内的排布方案和在出车序列中的位置关系可以采用以下数学模型来表示：

目标函数：

（1） 目标函数得分最大化

$$F=\max\left\{f_1q_1+f_2q_2+f_3q_3+f_4q_4\right\} \tag{4.1}$$

其中：q 为权重，q_1=0.4，q_2=0.3，q_3=0.2，q_4=0.1

（2） 优化目标一得分

$$f_1=100-\sum\varphi_{hr} \tag{4.2}$$

（3） 优化目标二得分

$$f_2=100-\sum\varphi_{uv} \tag{4.3}$$

（4） 优化目标三得分

$$f_3=100-D \tag{4.4}$$

（5） 优化目标四得分

$$f_4=100-0.01\times(T-9C-72) \tag{4.5}$$

约束条件： (4.6)

（1） 进出车队列长度为四驱与二驱之和或混动与燃油之和

$$C=\begin{cases}S_h+S_r\\S_u+S_v\end{cases} \tag{4.7}$$

（2） PBS 进车序列

$$\pi=\{1,\ldots,C\} \tag{4.8}$$

（3） 进车序列π通过函数 Ω 映射到出车序列π'

$$\pi\xrightarrow{\Omega}\pi' \tag{4.9}$$

（4） PBS 出车序列

$$\pi'=\{\Omega(1),\ldots,\Omega(C)\} \tag{4.10}$$

（5） 规定进出车序列车辆之间存在一一对应关系

$$\begin{cases}\sum_{k=1}^{C}e_{ki}=1,\quad i=1,2,\ldots,C\\\sum_{i=1}^{C}e_{ki}=1,\quad k=1,2,\ldots,C\end{cases} \tag{4.11}$$

（6） 要求每一辆车停放在该条车道的最前端（或该车道前方相邻位置都有一辆进车位置靠前的车）

$$\sum_{i=0}^{j-1}z_{ij}=1,\quad j=1,2,\ldots,C \tag{4.12}$$

（7） 每辆车后相邻位置最多只能安排给一辆进车序列靠后的车辆停放

$$\sum_{j=i+1}^{C}z_{ij}\le 1,\quad i=1,2,\ldots,C \tag{4.13}$$

（8） 车所占车道不允许超过缓冲区的总车道数

$$\sum_{j=1}^{C}z_{0j}\le m \tag{4.14}$$

（9） P_{ij} 的值等于车 i 后同一车道内的车辆数

$$\begin{cases}\sum_{i=0}^{j-1} p_{ij} - \sum_{i=j+1}^{C} p_{ji} = 1, \quad j = 1,2,...,C-1 \\ \sum_{i=0}^{C-1} p_{iC} = 1\end{cases} \tag{4.15}$$

（10）PBS 中每一车道所放置的车数不得超过其容量

$$z_{ij} \le p_{ij} \le n \cdot z_{ij}, \quad i = 0,...,C-1; j = i+1,...,C \tag{4.16}$$

（11）Z_{ij} 为整数变量

$$z_{ij} \in \{0,1\}, \quad i,j = 0,...,C; i < j \tag{4.17}$$

（12）e_{ki} 为整数变量

$$e_{ki} \in \{0,1\}, \quad k,i = 0,...,C \tag{4.18}$$

（13）X 表示位置矩阵

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1T} \\ x_{21} & x_{22} & \cdots & x_{2T} \\ \cdots & \cdots & \cdots & \cdots \\ x_{C1} & x_{C2} & \cdots & x_{CT} \end{bmatrix} \tag{4.19}$$

这里给出了重排序问题的数学建模表达式，由于两向中约束条件不一致，具体的模型相关变动将在后文中给出。

4.3 基于混流生产模式的重排序方法

在建立模型的基础上，本文给出了四种重排序方法，将其应用到汽车的涂装-PBS-总装的生产流程中。

4.3.1 分组重排序法

分组重排序法其实是在具体的总装车间排序目标下，利用贪婪规则来构建最优出车序列的，在对每组的车辆进行重排分区时，假设出车序列有 k-1 辆车，按照贪心算法从该组中选择第 k 辆车。

在完成汽车的涂装后，到达 PBS 区域内，首先需将到达的车辆序列拆分成为连续的 n 个子序列，并且每个子序列包含 m 辆车，各组的车辆可以依次占据 PBS 中同一列上的所有车位，从首个位置向后逐步确定每一出车序列位置所对应的车辆，最后得到一个完整的最优出车序列。对于出车序列位置$k(k = 1,\dots,S)$，如果$k \in [l-m+1, lm](l = 1,\dots,n)$，则第 k 辆车的候选集包含进车第 l 个子序列内所有没有被选择的车辆。由此可得出，构建出车序列的阶段相当于每个子序列内部车辆重新排列的阶段。

如果进车序列$\pi = \{1,\dots,9\}$，PBS 容量为3×3，即有 3 条三个车位的车道。按照分组重排方法将该序列拆分为三个组，然后对每组中的车辆进行重排得到，图中每辆车身上的第一个数字表示该车的进车顺序，圆圈内的数字为该辆车的出车序列位置，假设重排后得到三个子序列$\pi_1' = \{3,1,2\}, \pi_2' = \{6,4,5\}, \pi_3' = \{8,9,7\}$，如图 4.7 所示。

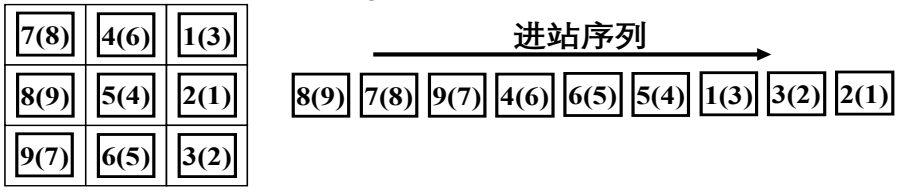

图 4.7 分组重排序示意图

4.3.2 基于步进式算法的重排序法

步进式重排序法是基于贪婪思想逐步构建的出车序列，当有一辆车被最佳释放时，需同时确认该车在 PBS 中的放置位置。为了满足题目假设，该车以及在缓冲区内的车需满足先进先出原则，对车身的排布进行可行性验证，找到最佳方案。

将所有未重排的车作为候选集，依照启发式规则从该候选集中选择车辆，如果该车未通过可行性验证，则选择候选集中的其他车辆，只有满足可行性验证的才是最佳出车序列。首先应当选择非空车道，如果当前还有 i 辆车需要进入，j 为某非空车道最后一辆车，那么 i 要进入这条车道必须要大于 j，否则 i 无法在 j 之后进入车道，当有多个可供选择的备选车道时，选择 *min(i-j)*这条车道；但如果没有满足条件的非空车道时，则任选一条空车道进入。之后进行可行性验证：

当前所选车辆能否进入 PBS 内，剩余的未重排车辆是否存在至少一种满足先进先出的填充方法，这两个条件都必须满足，才能证明可行性验证成功。假设 l 为当前车道，i_l 为车道 l 内的最后一辆车进车序列位置序号，*p(il)*为未重排车辆中进车序列位置序号大于 i_l 的车数，*g(l)*为当前车道 l 上剩余的车位数，若车道为空，则 i_l 为 0。所有剩余的未重排车辆都可以在满足先进先出的情况下获得在 PBS 中的一个车位，建立以下不等式：

$$p(i_l) \geq \sum_{k\in\{k|\geq i_l\}} g(k), \forall l \in \{1,\dots,M\} \tag{4.20}$$

这就保证了当前已构建的部分进车序列是可以实现的，在任意车道 1 上，若满足$i_k \geq i_l$的车道的剩余车位数之和大于可进入 PBS 的未重排车辆数 p(il)，就说明 PBS 内无法被填满。

若有进车序列$\pi = \{1,2,3,4,5,6,7,8,9\}$，缓冲区 PBS 容量为$3 \times 3$。假设按照上述步进式重排序法来构建出车序列，进站序列第 4 辆车的出车序列为 1，进车序列第 3 辆车的出车序列为 2，如图 4.8 所示。按照启发式规则优先选择车 2 作为出车序列的第 3 辆车，此时可以发现，在车道 1 上，进车序列位置序号大于 4 的未重排车辆数为 5，车道 1 剩余车位数为 2，不满足不等式 4.20；在车道 2 上，进车序列位置序号大于 3 的未重排车数为 5，车道 1 和车道 2 的剩余车位数之和为 4，不满足不等式 4.20；在车道 3 上，所有满足$i_k \geq i_3$的车道剩余车位数之和为 6，而进车序列位置号大于 2 的未重排车辆数为 5，不满足不等式 4.20；此种情况下车辆 1 无法进入缓冲区 PBS，车辆 2 的可行性检验失败。

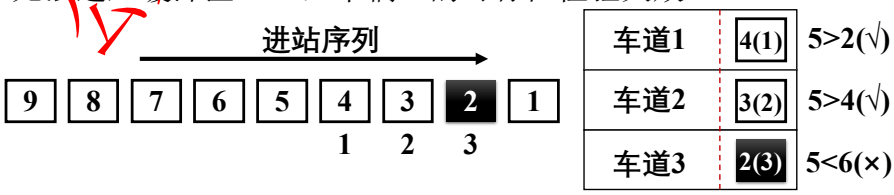

图 4.8 车辆 2 的可行性检验失败

基于贪婪的启发式规则，应使得车辆通行时尽可能地满足可行性验证，当第一个选择的车道验证失败时，可按照前文中的填充方案继续选择其他车道，继续下一步的可行性验证。

若有进车序列$\pi = \{1,2,3,4\}$，缓冲区 PBS 容量为2×2。假设按照上述步进式重排序法来构建出车序列，对进站序列第 3 辆车进行可行性验证，验证通过后 3 车的出车序列为 1，剩余候选集中的车辆按优先顺序排列为{2,1,4}，优先选择车 2 作为出车序列的第 2 辆车，不满足不等式 4.20；则按照优先顺序，选择车 1 进行可行性验证，验证通过确定车 1 为出车序列的第一辆车。依此类推后，最后得到出车序列以及车辆在 PBS 中的排列方案。

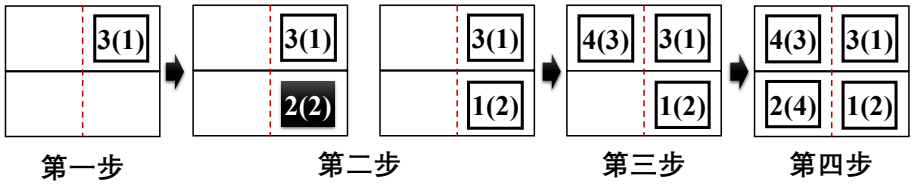

图 4.9 步进式重排序示意图

4.3.3 基于遗传算法的重排序法

遗传算法(GA)是数学中最常用来解决最优化问题的算法，Grenfenstette 提出的遗传算法从代表问题可能潜在解集的一个种群开始，对种群反复进行选择、交叉已经变异操作，直接对结构对象进行操作，估计各个体的适应值，采用概率化的方式进行全局最优解的搜寻，根据“适者生存、优胜劣汰”的进化规则，使得群体越来越向最优解的方向进化[4][5][6]。

算法的基本运算流程包括染色体编码、初始化种群设置、建立适应度函数对个体进行评价、执行遗传操作、算法的终止规则、染色体解码，具体的算法流程图如图 4.10 所示。车身序列可以被简便地编码为染色体，只要给出适应度函数的定义，遗传算法就能直接对车身序列质量进行筛选，每轮迭代都在原本的种群上更新得到表现更好的种群，基本遗传算法的时间复杂度为$O(n^2)$，能够在较短的运算时间内给出较为优质的可行解。

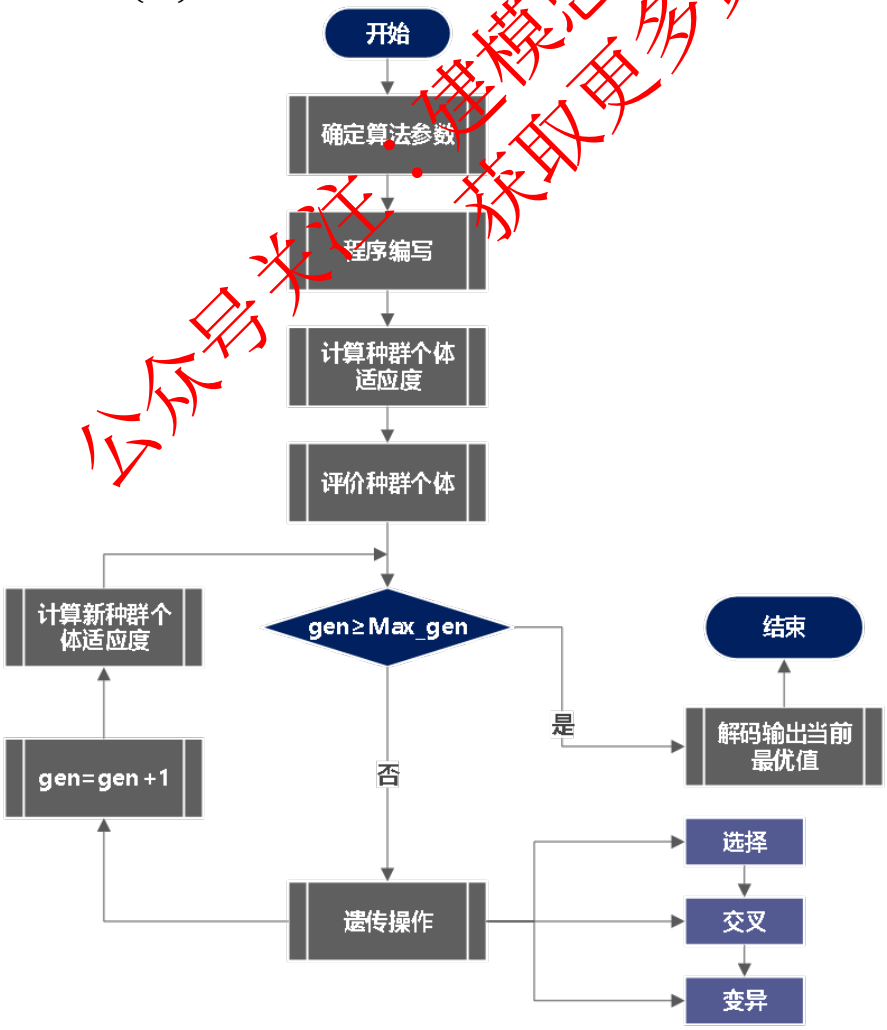

图 4.10 遗传算法流程图

在运行步骤当中，合理的种群规模大小很关键，会对遗传算法的性能产生影响[7]，较大数量的种群可以遍历所有可行解，较小数量的种群计算量小，算法可以以很快的速度收敛，但可能结果是局部最优解，所以一个合适的种群规模变得尤为重要，最优的种群规模应当是决策变量数量的 4 倍到 6 倍，一般在 20~100 之间[8]；其次是交叉概率，来衡量种群个体是否进行交叉操作的可能性大小，从种群中随机选出部分个体做交叉操作产生子代新个体，交叉概率一般的取值范围为 0.4~0.99；变异概率是个体是否进行变异操作的评价标准，变异概率太大可能会产生不可行解，太小则不易跳出局部最优而重复迭代，变异概率的取值一般为 0.0001~0.5；最大迭代次数是遗传算法预先设定的一种运行终止规则[9]，若当前已达到最大迭代次数，则退出循环终止操作，得到最优解。

4.3.4 基于模拟退火算法的重排序法

模拟退火算法的出发点是基于物理退火过程，包含加温、等温和冷却过程，模拟金属材料退火过程，金属内部粒子随着温度的升高由有序变得无序，同时粒子内能升高，其冷却过程却是缓慢冷却，粒子由无序转为有序状态，最后在常温时达到基态，此时内能最小[10]。其中内循环所使用的 Metropolis 准则，在一定的概率范围内接受恶化解，从而跳离局部最优陷阱，达到全局最优。具体的模拟退火算法流程图 4.11 所示。

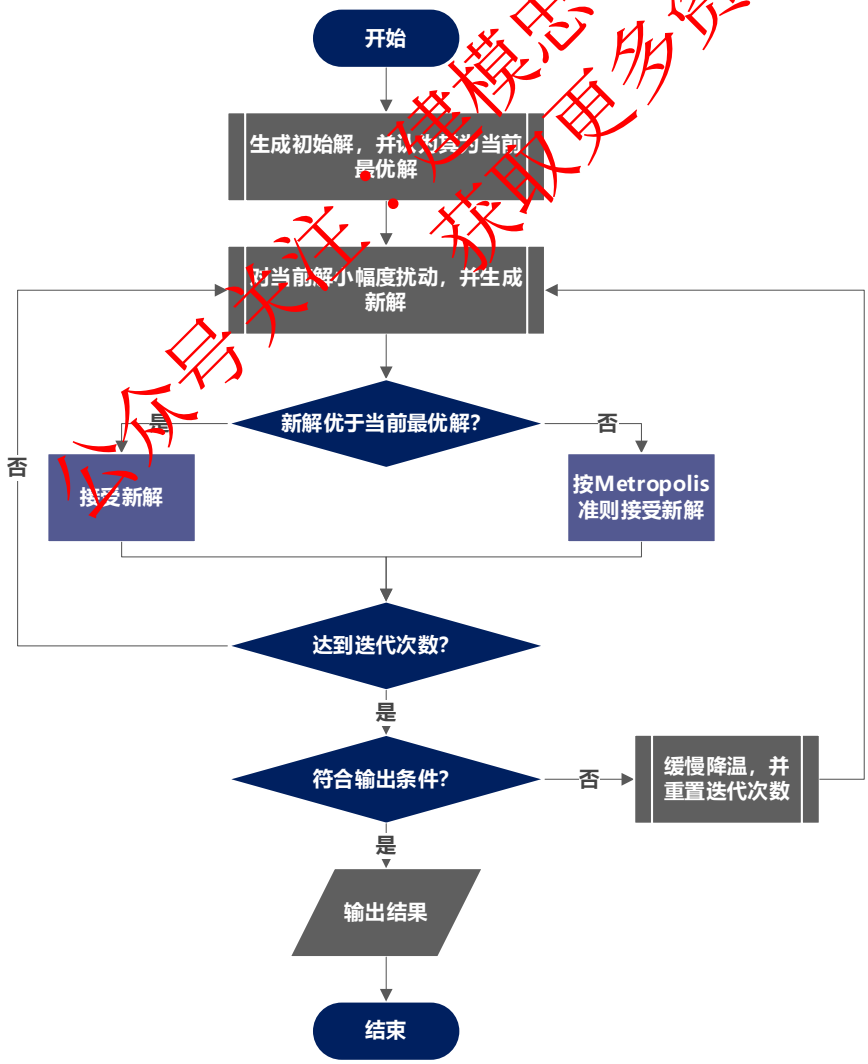

图 4.11 模拟退火算法流程图

在给定问题的初始状态和退火的初始状态，包括设定初始序列 D 和初始温度 T_0，通过束搜索算法可以得到不错的初始解，能在一个较好的初始状态下搜索更优的解；在一定的温度下对解进行随机搜索，每次内循环会在当前解的基础上产生一个新的序列，根据 Metropolis 准则进行判定是否接受新解；之后冷却，通过降温函数来降低温度，终止温度是判断外循环是否终止的条件[11]。

模拟退火算法目前已经广泛应用于混流装配、图像处理、投产制造等领域[12]，与其他智能算法相比，模拟退火算法的优势在于局部搜索能力强以及计算结果较为简单，能匹配解决各类问题，但是每次都是对解空间内的单个解不断进行迭代寻找最优解，算法的运行速度会比其他算法要慢，耗费时间更长。

4.4 模型分析与排序比较

本文的总数学模型包括序列重排后进出车辆之间对应关系的分配模型和 PBS 内车辆布排方案的车辆路径模型，建立了描述问题的整数规划模型。在各种排序方法的比较当中，分组重排序法是将涂装出车序列拆分为多个较小的子序列，在每个小组当中，组内的车辆会依次地进行连续重排，之后每组的车辆排序好后形成最终的总装进车序列；在步进式重排序法中，总装进车序列和车辆在 PBS 内的排布方案是同时进行的，进行可行性验证，逐步确认填充方案，依此类推后，最后得到出车序列以及车辆在 PBS 中的排列方案；在基于遗传算法的重排序法中，车身涂装序列可以被编码为染色体，在给出适应度函数的定义下，算法筛选车身序列的质量，达到最大迭代次数，输出最优染色体，后解码得到结果；在基于模拟退火算法的重排序法中，利用全局优化方法，先得到初始总装进车序列，在每次内循环后，搜索新的更优的解，根据 Metropolis 准则进行判定，最终得到全局最优解。

可以发现，在以上四种重排序方法中，遗传算法和模拟退火算法是考虑较为全面的，且为了使得研究方向和使用方法更加精确，本文通过关键词共现来揭示有关汽车排序相关研究的关键内容，以 CNKI 数据库为数据源，以关键词“混流排序”来检索，设置检索时间为 2012 年-2022 年，借助可视化分析工具 Citespace 对文章的关键词进行共现与聚类，从关键词的角度对研究汽车排序问题的内容进行探讨。Citespacce 是可视化文献分析软件，能够揭露现有研究的热点和它们之间的关系，最终的关键词共现时序分析结果见图 4.12。关键词出现的频次越多，聚类越靠前，图中的文字越大，在学者的研究方向中，使用较多的算法包括遗传算法和模拟退火算法。本文将在基于题目的算例要求下，将模拟退火算法和遗传算法分别融入到数学模型当中，对比这两种算法的优劣，并选出最优算法。

图 4.12 “混流排序”的关键词共现分析图

五、问题一求解

5.1 问题一的分析

问题一的主要任务是建立合理的 PBS 优化调度模型，使得 PBS 出车序列尽可能满足总装生产需求。此外，还需要将模型应用于附件 1 和附件 2 给出的 PBS 进车序列清单，输出评分结果和调度过程中每个车身在每一秒所处的区域。

为了解决该难题，我们可以根据题目给出的 PBS 约束说明、相关时间数据说明以及 4 个优化目标，制定出模型约束条件，并提出目标函数，以此建立 PBS 优化调度的数学模型。完成模型建立后，需要进行求解。求解的过程就是 PBS 调度方案不断完善优化的过程，最终使得目标函数的得分最高。为此，我们可以采用逆向思维，以输出结果为导向，将解题步骤分解为 2 个阶段：阶段一，需要求得 PBS 最优出车序列；阶段二，要在阶段一结果的基础上，返求 PBS 进车方案，从而得到完整的优化调度方案，并计算得到附件 1 和附件 2 的得分结果。

对于阶段一，我们可以采用模拟退火算法，获得最优出车序列。同时，可以通过遗传算法进行比较。模型的目的是使得出车序列减分最少，即模型的优化目标函数得分最多。这里，有两点需要进行说明。第一，由于附件给出的进车序列是从小到大按顺序排列的，所以，在获得最优出车序列的过程中，我们假设出车序列也应尽量满足从小到大排布，以减少返回道的使用次数，符合优化目标 3。第二，由于优化目标 4 权重最小，因此优先考虑优化目标 1~3。

对于阶段二，分析假设 18 以及 PBS 相关时间数据说明，我们可以发现，当考虑到优化目标 4 要求耗时最短时，横移机接送车身进入各个进车道和返回道的优先级是有区别的。

然后，我们可以先根据进车顺序对最优出车序列进行分组，组内车辆最多的组别优先分配优先级最高的进车道。在此基础上，可以通过 Matlab 编程实现具体的 PBS 调度方案，输出每个车身在每一秒所处的区域，并计算出考虑 4 个优化目标的最终得分结果。

问题一的解题思路如图 5.1 所示。

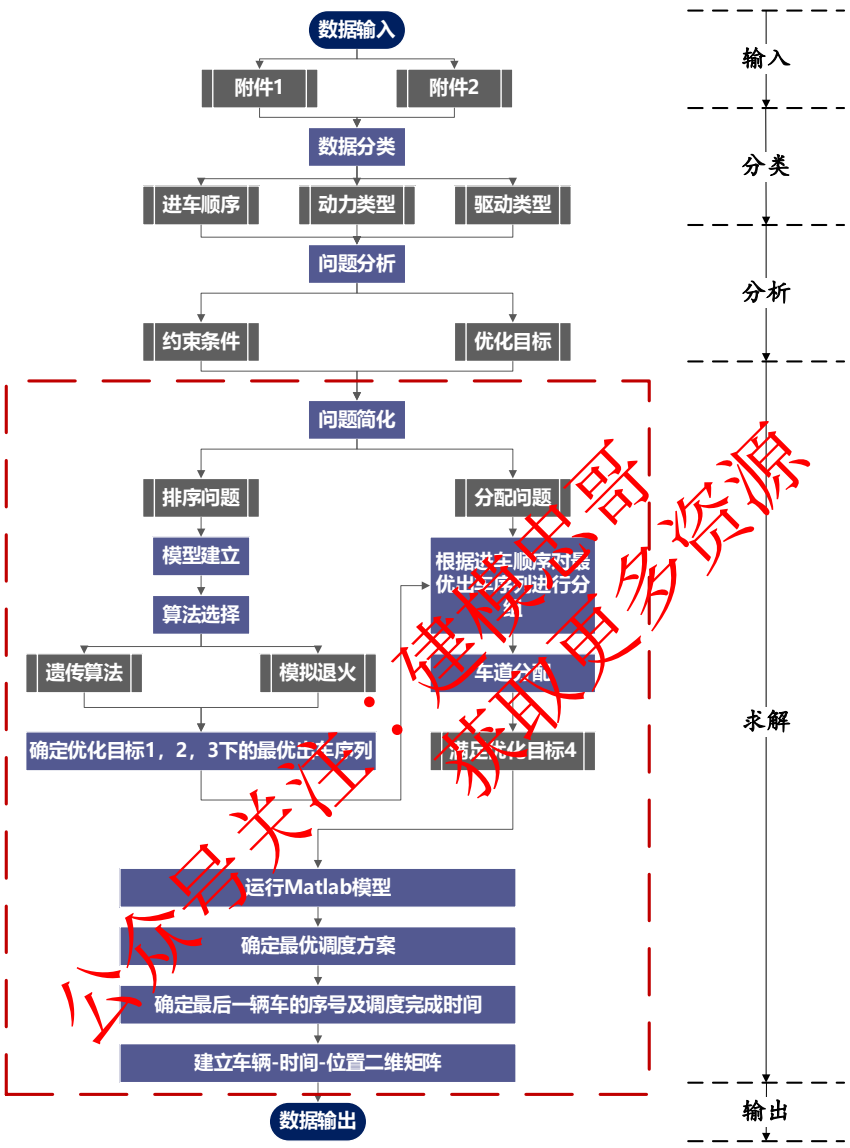

图 5.1 问题一解题思路图

5.2 数学模型的建立

5.2.1 基于模拟退火算法的数学模型

针对问题一的求解过程可以分为两个阶段：

(一)阶段一：最优出车序列的获取

1) 采用模拟退火算法获取 PBS 的最优出车序列，需要进行算法参数设置。这里，我们将初始温度 T_0 设置为 100；退火速率 k 设置为 0.97；算法终止条件，即外层循环的迭代次数 N_{max} 设置为 1000 次；每个温度下的迭代次数，即内层循环的迭代次数 L 设置为 500 次；系统能量E，即根据优化目标 1~3 评判后扣除的分数，越低越好。

表 5-1 模拟退火算法计算参数

序号	符号	单位	值
1	T_0	℃	100
2	k	/	0.97
3	N_{max}	次	1000
4	L	次	500

2) 解的形式是一个数字序列，我们首先通过 Matlab 函数 randperm()将附件给出的进车序列π随机打乱，从而产生一个初始解π'_0，并计算出对应系统能量 E_0。

3) 对初始解π'_0施加随机扰动，生成一个新解π'_1，并计算对应系统能量 E_1。随机扰动的施加可以采用交换法、移位法和倒置法，如图 5.2 所示。

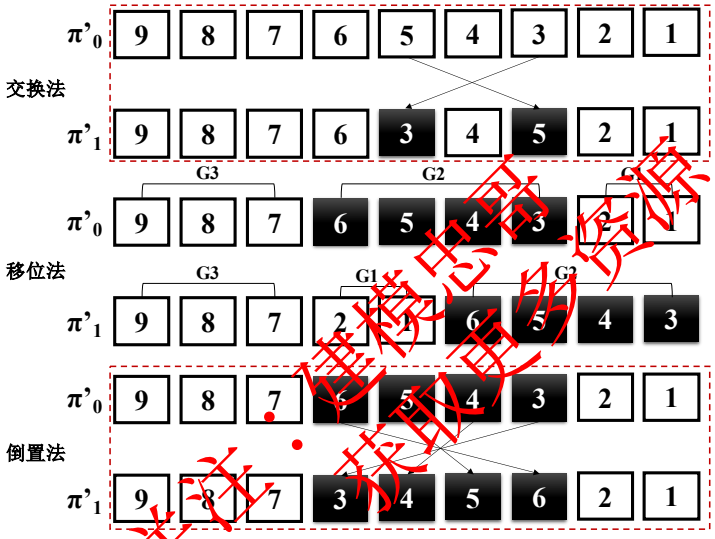

图 5.2 随机扰动施加方法

4) 按照 Metropolis 准则，判断是否接受新解。

$$P=\begin{cases}1, & E(d+1)<E(d)\\ \exp(-\dfrac{E(d+1)-E(d)}{T_w}), & E(d+1)\geq E(d)\end{cases} \tag{5.1}$$

$$E=q_1\sum\varphi_{hr}+q_2\sum\varphi_{uv}+q_3D \tag{5.2}$$

5) 在同一温度下，内层循环重复 500 次扰动和接受过程。

6) 当外层循环的迭代次数等于 1000 次后，算法终止，得到最优解。

(二)阶段二：PBS 调度方案及得分

1) 根据不同情况下进车道的优先级顺序先对最优出车顺序进行分组。再按照各组内车身数量的多少分配进车道，车身数量较多的组优先划分到优先级顺序最高的进车道。车道优先级顺序如下：

i. 当车身不需要进入返回道时，车身进入 6 个进车道的优先级顺序为：进车道 4＞ 进车道 3＞ 进车道 2＝ 进车道 5＞ 进车道 1＝ 进车道 6。

ii. 当某些序号的车身需要进入返回道时，车身进入 6 个进车道的优先级顺序为：进车道 4＞ 进车道 3＝ 进车道 5＞ 进车道 2＝ 进车道 6＞ 进车道 1。

2) 分组结束后，根据出车顺序，返求每个车身的进入对应车道的初始时间。为简化模型，我们这里假设每辆车身到达 1 号停车位后都等待 9 s，然后再通过送车横移机到达 PBS 出车口。知道车身的进车时间、进车道编号以及出车时间后，就可以获得每个车身在每一秒所处的区域，以及优化调度方案的得分结果。

5.2.2 基于遗传算法的数学模型

(一)阶段一：最优出车序列的获取

通过遗传算法解决 PBS 最优出车序列的过程如下：

1) 读取涂装线车身序列，将所有车身动力和所有车身驱动分别进行整数编码，令迭代次数 i=0，随机生成初始种群，内含有 N 条染色体，每条染色体代表问题的一个解。
2) 将进车序列作为初始染色体，产生初始种群，种群中的染色体是由初始染色体中的基因点随机交换生成，计算种群内每条染色体的适应度值 Fit，判断算法是否满足设定的终止条件，如果不满足则更新迭代次数 i，令 i=i+1，生成选择种群 $Selext(i+1)$。
3) 进行交叉操作，采用双点交叉，随机选取两个染色体，随机选定交叉点，两层基因链同时进行交叉操作，生成交叉种群 $Pc(i+1)$。
4) 进行变异操作，在一条染色体上随机选择两个基因点，交换两个基因点的值，两层基因链同时进行变异，生成变异种群 $Pm(i+1)$。
5) 进行选择操作，首先将当前代中适应度最高的两个个体保留到下一代种群中，再按照轮盘赌选择方式生成新一代种群。
6) 在新一代种群的基础上，返回步骤 3。
7) 输出适应度最高的染色体，解码得到最优出车序列，算法结束。

(二)阶段二：PBS 调度方案及得分

此部分与模拟退火算法阶段二相同，详见 5.2.1 节。

5.3.两种算法模型求解

本文使用两种算法分别在 Matlab 软件编程运行之后输出最优序列，采用 Origin 制图得到结果对比分析。遗传算法和模拟退火算法在附件一和附件二中的数据对比存在差异，在满足四个优化目标的基础上，从 PBS 原始进车序列转变为 PBS 最优出车序列，面对此类汽车重排序分配问题，采用模拟退火算法得到的结果会优于遗传算法。

为保证得分最高，需要在满足车辆排序的基础上，使得原始进车顺序和最优出车顺序不要相差太大，在 x=y 上下波动幅度不要太大，这样能够尽可能少的使用返回道和减少运输时间，如下图 5.3 所示。从附件一数据来看，模拟退火的异常值点比遗传算法的点更少，基本为一条直线；从附件二数据来看，两种算法的情况基本一致。当面对数据基础条件更差时，模拟退火算法会更具有适用性和高效性。

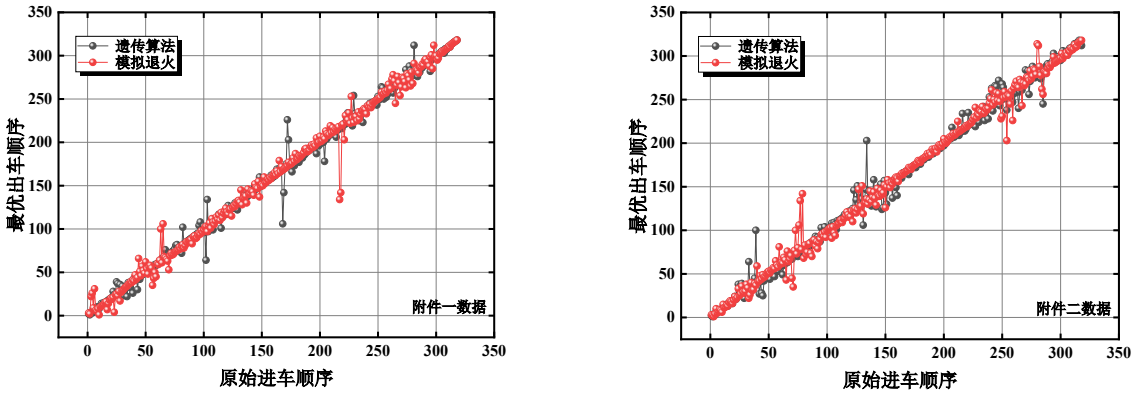

图 5.3 SA 和 GA 最优出车顺序比较

如图 5.4 所示，两种算法在迭代相同次数的基础上，当仅考虑 PBS 内的进出车重排序时，遗传算法的扣分数比模拟退火算法的扣分数多，在附件一的数据上的区别更为明显，发现**模拟退火算法**更适合用于**脱敏的大数据集的建模**，在**各个目标的得分情况下均能优于遗传算法**。

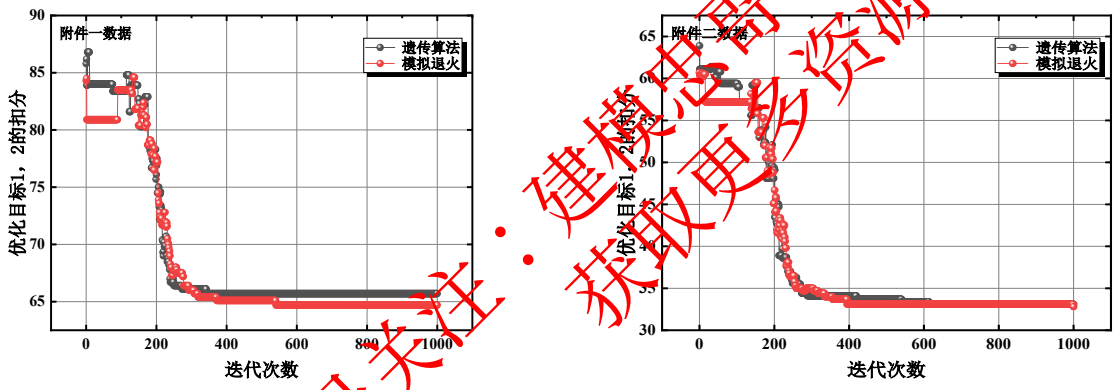

图 5.4 优化目标 1&2 条件下 SA 和 GA 在 1000 次迭代中的扣分演化过程

如图 5.5 所示，当涂装出车序列进入缓冲区内，若满足混动车型间隔两台非混动车型、四驱车型和两驱车型 1:1 出车序列这两个优化目标，先不考虑出车顺序，如图中的黑色坐标点，第 2 辆进入的车很可能需要到第 200 辆才能出车，此种做法是不合理的，将浪费大量的时间资源和多次无效使用返回道。再同时考虑总调度时间和返回道使用次数这两个优化目标，当考虑进车顺序时，如图中的红色坐标点，尽可能满足先进先出原则，根据进车顺序对最优出车序列进行分组，车辆最多的组优先选择通过时间最短的车道。可以看出，红色的点组成了一条接近于 x=y 的线，在附件一和附件二两种算法输出结果的对比中，图中模拟退火算法的红色异常值点比遗传算法的突出点要少，与 x=y 线的拟合效果更好，尤其在附件一中效果更为显著。综上，考虑四个优化目标时，模拟退火算法优于遗传算法。

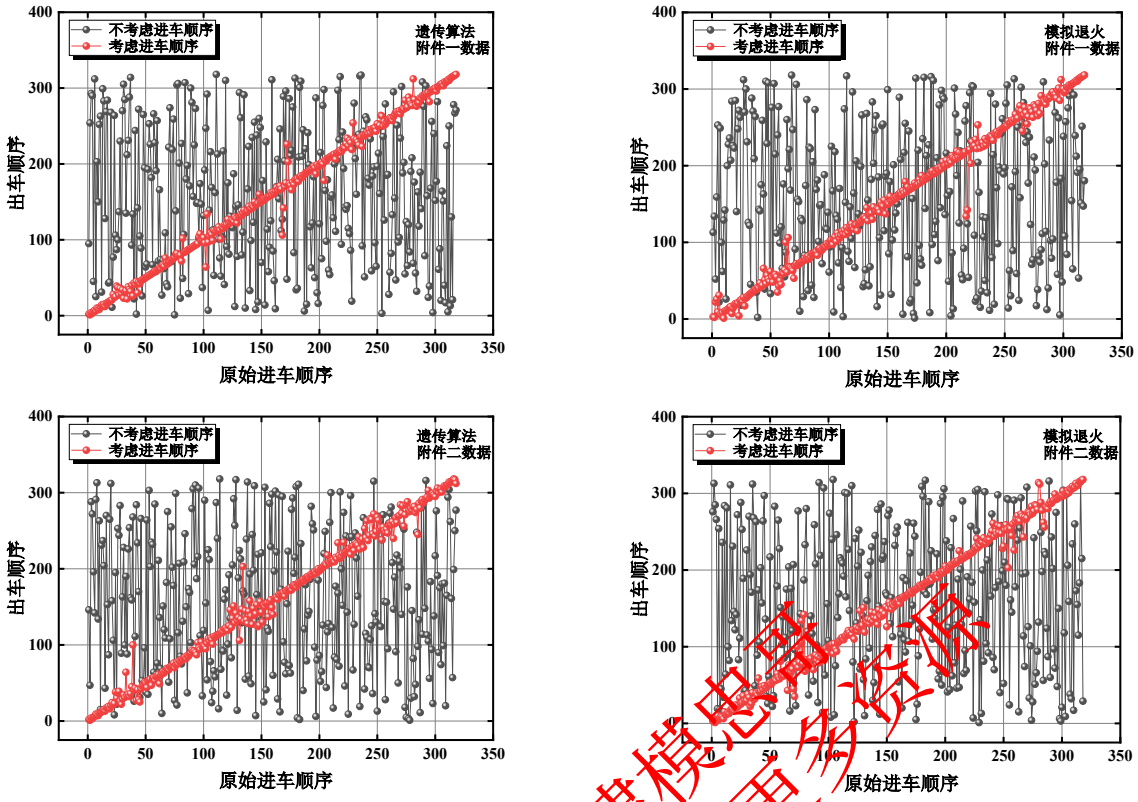

图 5.5 SA 和 GA 对数据处理前后对比

如图 5.6 所示，附件一数据使用模拟退火算法时，若不考虑进车顺序，重排序分配后的总分为-21.854 分，若考虑进车顺序，则需满足全部优化目标，在生产排序中可将资源更为有效利用，得分为 32.086 分，分数提高了 53.94 分；使用遗传算法时，不考虑进车顺序时的得分比模拟退火的时间要高，为-21.312 分，而满足全部优化目标后得分仅为 31.944 分，分数提高了 53.256 分，和模拟退火相比优化程度低了 0.684 分，不具有优越性。当使用附件二时，数据是经过调整和优化后的，总体得分都要高于附件一，当使用模拟退火时，若不考虑进车顺序，重排序分配后的总分为 7.908 分，若考虑进车顺序，则需满足全部优化目标，在生产排序中可将资源更为有效利用，得分为 63.72 分，分数提高了 55.812 分；使用遗传算法时，不考虑进车顺序时的得分比模拟退火的时间要高，为 9.568 分，而满足全部优化目标后得分为 63.4 分，分数提高了 53.832 分，和模拟退火相比优化程度低了 1.98 分。优化目标 3、4：分别以最少返回道的使用和最短调度时间为目标，本文在分配车道时，进行逆向求解，以最优出车顺序为标准，建立组别优先级，组内车辆越多优先选择通过时间短的车道，相较未考虑出车顺序的通过**时间可以减少 61.43%-67.48%，返回道使用次数减至 4 次，总得分提高 53.94-55.81 分。**

由此可见，无论是附件一还是附件二，在满足全部优化目标的前提下，模拟退火算法均要优于遗传算法。

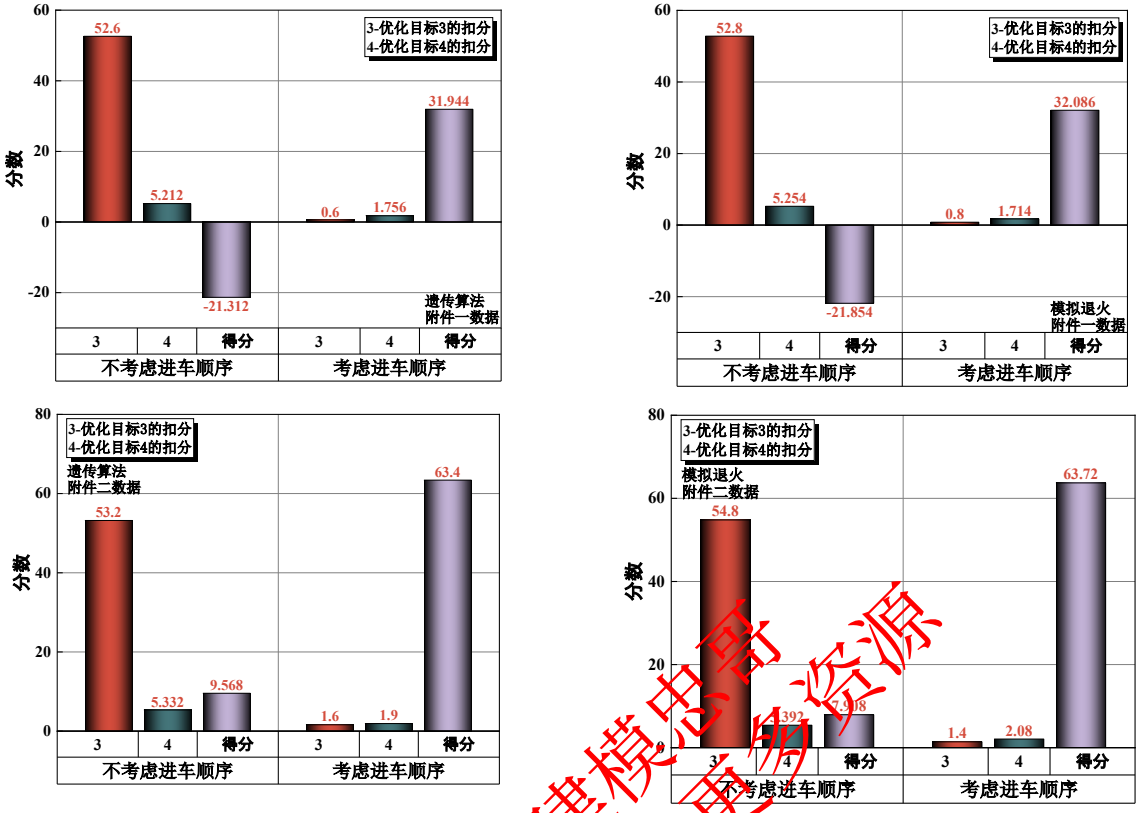

图 5.6 仅考虑优化目标 1&2 的出车顺序和算法迭代后的最优出车顺序比较

当考虑出车顺序时，最主要的影响因素是返回道的使用和总调度时间的不同，这是生产过程中必须考虑的关键因素。在现实生活中，大多是类似于附件一的数据，而本文运用的模拟退火算法针对于这种无序且不平稳的数据，在**各个目标的得分情况下均能优于遗传算法**，能够确定出最优调度方案。故得出结论，基于此算例，**本文使用的模拟退火算法优于遗传算法**。

5.4 问题一结果

根据模型运算结果可得，**问题一附件一的得分为 31.944，附件二的得分为 63.4**。表 5-2 为模拟退火算法求解多目标混流生产重排序问题一的最优调度结果，详见附件 result11.xlsx。如第 0 秒 1 车身被装载到接车横移机上，接着被卸载到 4 进车道 10 号车位，第 3 秒 2 车身被装载到接车横移机上，接着被卸载到 3 进车道 10 号车位....

表 5-2 附件 1 优化后的序列(π)-时间(T)联合分布矩阵(result11.xlsx)

T / π	0	1	2	3	...	4687	4688	4689
1	410	410	410	410	...	3	3	3
2	1	1	1	310	...	3	3	3
3	0	0	0	0	...	3	3	3
4	0	0	0	0	...	3	3	3
5	0	0	0	0	...	3	3	3
6	0	0	0	0	...	3	3	3
...	...	...	...	...	...	...	...	...
315	0	0	0	0	...	3	3	3
316	0	0	0	0	...	3	3	3

317	0	0	0	0	...	3	3	3
318	0	0	0	0	...	2	2	3

表 5-3 为模拟退火算法求解附件 2 问题一的最优调度结果，详见附件 result12.xlsx。

表 5-3 附件 2 优化后的序列(π)-时间(T)联合分布矩阵(result12.xlsx)

π \ T	0	1	2	3	...	4831	4832	4833
1	410	410	410	410	...	3	3	3
2	1	1	1	310	...	3	3	3
3	0	0	0	0	...	3	3	3
4	0	0	0	0	...	3	3	3
5	0	0	0	0	...	3	3	3
6	0	0	0	0	...	3	3	3
...	...	...	...	...	...	...	...	...
315	0	0	0	0	...	3	3	3
316	0	0	0	0	...	3	3	3
317	0	0	0	0	...	3	3	3
318	0	0	0	0	...	2	2	3

六、 问题二求解

6.1 问题二的分析

问题二是在问题一的基础上，考虑去除 PBS 约束说明中的第 6、7 两条约束后，再确定新的 PBS 优化调度方案。约束说明中的第 6、7 两条约束的意思是当返回道 10 停车位或者进车道 1 停车位有车身时，横移机要优先对其进行处理。而问题二去除这两个约束后，横移机就可以优先处理需要进入或者离开进车道优先级高的车身，使得这些满足要求的车身等待时间减少，从而更大限度地节约时间。

问题二的解题思路和问题一类似，同样可以分为两个阶段。问题二和问题一在阶段一都是求取最优出车序列，考虑优化目标 1~3，因此求解过程不变。对于阶段二，需要输出 PBS 调度方案和得分结果。由于约束条件改变，所以阶段二的问题假设要有所变化。问题的阶段二，我们假设各进车道上的车身到达 1 号停车位后都需要等待一段时间。在此处，这个假设我们可以变成：各车道车身到达 1 号停车位后等待的时间，根据车身所在车道的优先级顺序而有所不同。车道优先级越高，车身等待时间越短。这样，车身在 PBS 调度过程中会优先进入优先级高的车道，并优先从该车道离开。从整体来看，会更加满足优化目标 4 的要求。

6.2 数学模型的建立

6.2.1 基于模拟退火算法的数学模型

针对问题二的求解过程同样分为两个阶段，如下：

(一)阶段一：最优出车序列的获取

约束 6 和约束 7 的去除，并不影响最优出车序列的求解过程，所以，此处阶段一求解过程与问题一一致，详见 5.2.1 节。

(二)阶段二：PBS 调度方案及得分

1) 根据不同情况下进车道的优先级顺序对最优出车顺序进行分组。按照各组内车身数量的多少分配进车道，车身数量较多的组优先划分到优先级顺序最高的进车道。
2) 根据改变后的假设，我们将 1~6 号进车道上的车身到达 1 号停车位后的等待时间分别设置为：[9s, 6s, 0s, 3s, 6s, 9s]。其余求解步骤不变，最终得到 PBS 优化调度方案和得分结果。

6.2.2 基于遗传算法的数学模型

通过遗传算法求解最优出车序列的过程与 5.2.2 节相同。

针对 PBS 调度方案的求解，同样需要将 1~6 号进车道上的车身到达 1 号停车位后的等待时间分别设置为：[9s, 6s, 0s, 3s, 6s, 9s]。其余求解步骤不变，最终得到 PBS 优化调度方案和得分结果。

6.3 问题二结果分析

6.3.1 问题一、二对比

因问题二模型的求解过程与问题一类似，所以此处只进行最终结果的展示与分析，**问题一附件一的总得分为 32.806**，各优化目标扣分项依次为 63.6、1.8、0.8、1.714，**附件二的得分为 63.72**，各优化目标扣分项依次为 31.6、1.2、1.4、2.08；**问题二附件一的得分为 33.646**，各优化目标扣分项依次为 63.2、1.5、0、1.654，**附件二的得分为 64.972**，各优化目标扣分项依次为 31.6、1.2、0.4、1.828。相比问题一的计算结果，问题二中**返回道使用次数**可以优化为 **0 次**，**最短调度时间 3.51%-12.22%**，**总得分提高 1.93%-4.64%**。

如图 6.1 所示，当使用附件一、附件二数据时，1、2、3、4 项为各优化目标的扣分数，问题一的扣分均比问题二的扣分数要高，在去除约束 6、7 之后，模型的约束条件变得放松，横移机就可以优先处理需要进入或者离开进车道优先级高的车身，故问题二的总得分比问题一要高些。优化目标 1、2:分别以汽车的动力和驱动类型为排序依据，受数据影响较大，附件二数据可以分别相对提高 **50.00%-50.32%和 20.00%-33.34%**的分数。

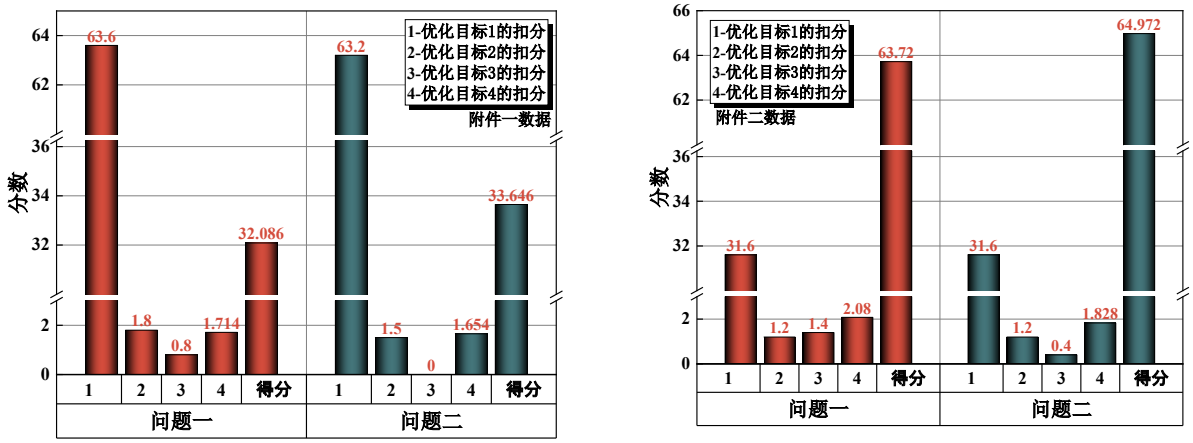

图 6.1 问题一，二的评分对比

6.3.2 问题二结果

如图 6.2 所示，基于模拟退火算法建立的数学模型可以较好的匹配此类混流重排序问题，在尽量满足四个目标函数的要求上，可以发现 result21.xlsx 和 result22.xlsx 中的结果可以很好的拟合成一条直线，说明涂装出车序列到达缓冲区后，也能够按照要求尽快地完成重排序成为总装进车序列，达到最优调度。

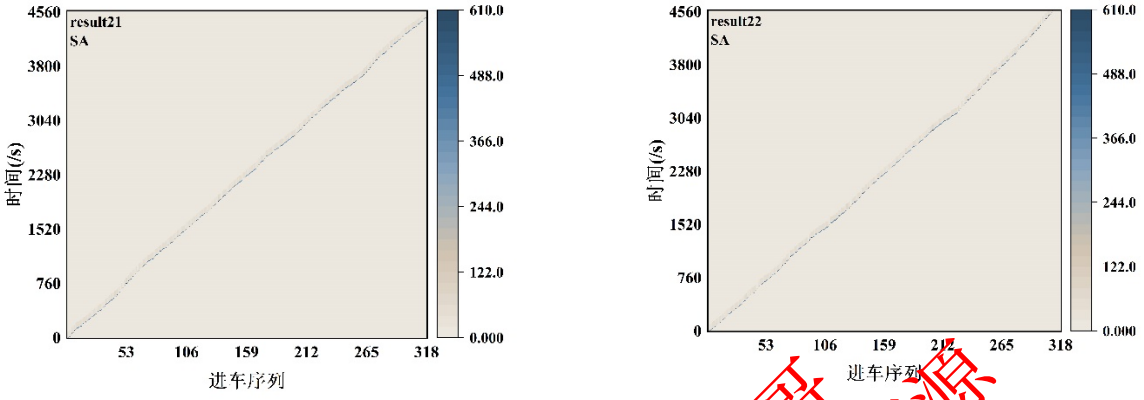

图 6.2 问题二结果矩阵对比

根据模型运算结果可得，**问题二附件一的得分为 33.646，附件二的得分为 64.972**。表 6-1 为模拟退火算法求解多目标混流生产重排序问题二的最优调度结果，详见附件 result21.xlsx。如第 0 秒 1 车身被装载到接车横移机上，接着被卸载到 4 进车道 10 号车位，第 3 秒 2 车身被装载到接车横移机上，接着被卸载到 3 进车道 10 号车位....

表 6-1 附件 1 优化后的序列（π）-时间（T）联合分布矩阵（result21.xlsx）

π \ T	0	1	2	3	...	4585	4586	4587
1	410	410	410	410	...	3	3	3
2	1	1	1	310	...	3	3	3
3	0	0	0	0	...	3	3	3
4	0	0	0	0	...	3	3	3
5	0	0	0	0	...	3	3	3
6	0	0	0	0	...	3	3	3
...	...	...	...	...	...	...	...	...
315	0	0	0	0	...	3	3	3
316	0	0	0	0	...	3	3	3
317	0	0	0	0	...	3	3	3
318	0	0	0	0	...	2	2	3

表 6-2 为模拟退火算法求解附件 2 问题二的最优调度结果，详见附件 result22.xlsx。

表 6-2 附件 2 优化后的序列（π）-时间（T）联合分布矩阵（result22.xlsx）

π \ T	0	1	2	3	...	4759	4760	4761
1	410	410	410	410	...	3	3	3
2	1	1	1	310	...	3	3	3
3	0	0	0	0	...	3	3	3
4	0	0	0	0	...	3	3	3
5	0	0	0	0	...	3	3	3

6	0	0	0	0	...	3	3	3
...	...	...	...	...	...	...	...	...
315	0	0	0	0	...	3	3	3
316	0	0	0	0	...	3	3	3
317	0	0	0	0	...	3	3	3
318	0	0	0	0	...	2	2	3

七、 模型的验证

为进一步验证本文建立数学模型的有效性，自定义一个满足优化目标 1、2 的理论最优情况的数据表，见表 7-1。通过运行上述基于模拟退火算法建立的模型，得到的结果矩阵如表 7-2 所示。图 7.1 为采用表 7-2 数据绘制的云图。

由图表可知，所有车辆在同一时间的位置不存在重叠现象，车辆交替选用通过时间最短的 4 车道和 3 车道，最终完成时间为 195 秒，最终评分达到 99.372/100.000。

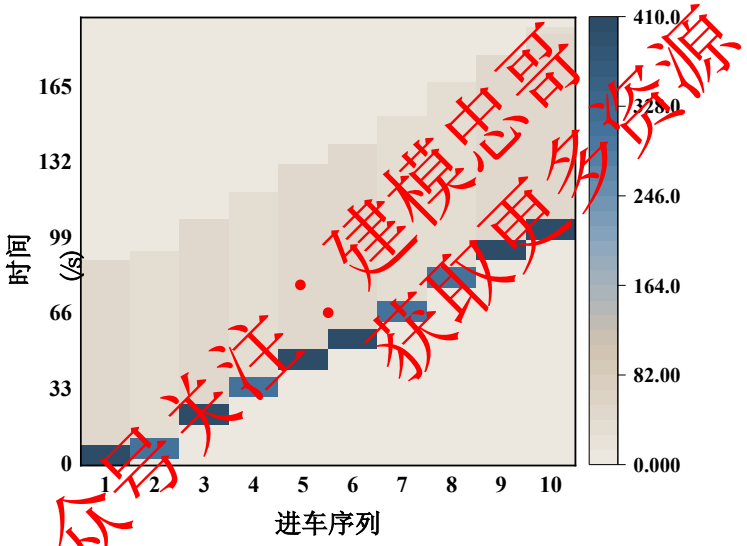

图 7.1 最优情况下结果矩阵

表 7-1 最优车身情况

进车顺序	车型	动力	驱动
1	*A*	*r*	*u*
2	*B*	*r*	*v*
3	*A*	*h*	*u*
4	*A*	*r*	*v*
5	*A*	*h*	*u*
6	*A*	*r*	*v*
7	*A*	*r*	*v*
8	*A*	*r*	*v*
9	*A*	*h*	*u*
10	*A*	*h*	*u*

表 7-2 附件 1 最优情况下的序列（π）-时间（T）联合分布矩阵

T / π	0	1	…	50	51	…	100	101	…	150	151	…	194	195
1	410	410	...	45	45	...	3	3	...	3	3	...	3	3
2	1	1	...	35	35	...	3	3	...	3	3	...	3	3
3	0	0	...	47	47	...	41	41	...	3	3	...	3	3
4	0	0	...	38	38	...	33	33	...	3	3	...	3	3
5	0	0	...	410	410	...	44	44	...	3	3	...	3	3
6	0	0	...	0	0	...	45	45	...	3	3	...	3	3
7	0	0	...	0	0	...	36	36	...	31	31	...	3	3
8	0	0	...	0	0	...	38	38	...	33	32	...	3	3
9	0	0	...	0	0	...	49	49	...	44	44	...	3	3
10	0	0	...	0	0	...	410	410	...	45	45	...	41	3

为进一步探究模型的运算过程，通过编写代码提取模型在优化目标 1-4 下的扣分情况，如图 7.2 所示。由图可知，理论最优情况下的数据在优化目标 1 和 3 的扣分为 0，这意味着车身的混动及燃油比例刚好符合每连续两辆混动车身之间存在两辆非混动车身，且返回道使用次数为 0。模型的主要扣分项在优化目标 2 和 4，分别扣 0.6 分和 0.033 分。

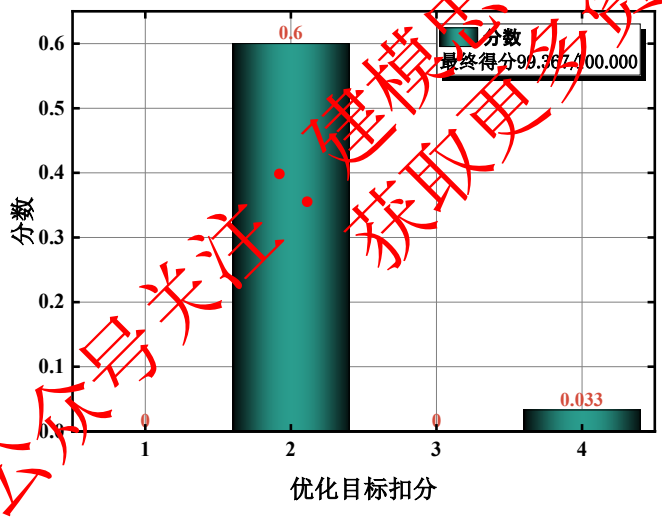

图 7.2 最优情况下各优化目标扣分情况

为探究扣分原因，进一步提取模型计算中产生的最优排序结果及通道选择情况，见图 7.3，图 7.4。

如图 7.3 所示，在设计理论最优情况数据表时，仅考虑混动和燃油车身数量满足式（1），四驱和二驱整体满足 1:1，但未考虑由于混动车身不存在 4 驱的情况，在分块时，最后 4 辆车分为两组（G3,G4），因此需要扣 2 分，考虑权重，最终扣 0.6 分。

其次，由于题目给出的理论最短完成时间 9C+72，并未考虑全部车辆选择 4 车道时，车辆每次前移需要的 9 秒时间，因此在实际调度过程中不可能达到理论最短完成时间。结合图 7.4 可知，10 辆车中存在 4 辆车走 3 车道的情况，最终完成时间 195 秒，考虑权重最终扣 0.033 分。

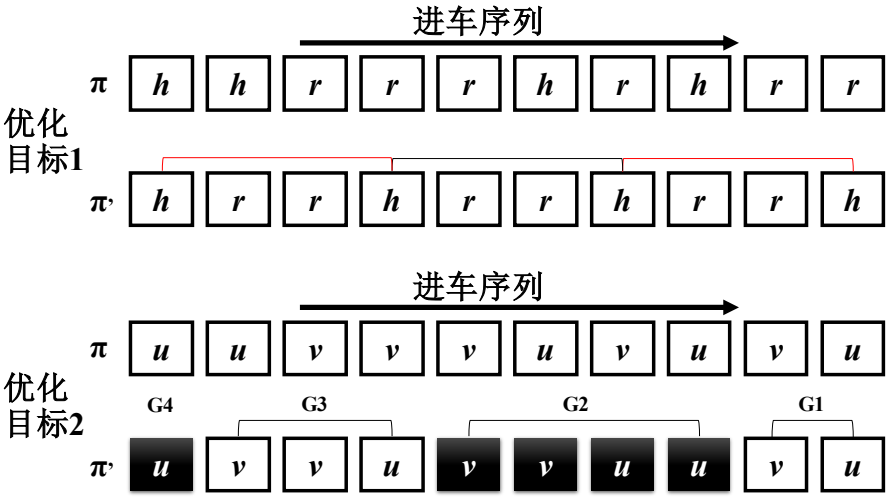

图 7.3 最优情况下优化目标 1，2 的排序方式

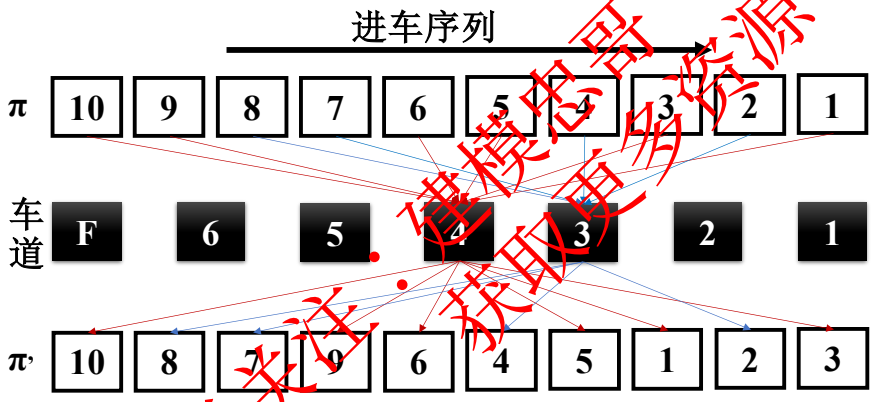

图 7.4 最优情况下车道分配

综上所述，本文**基于模拟退火算法建立的模型具有良好的有效性及适用性**，问题一、二的求解过程及运算结果是有效可靠的。

八、 模型的评价

8.1 模型优点

（1）针对汽车的 PBS 优化调度问题，本文根据题目给出的 PBS 约束说明、时间数据说明，以及为简化模型而提出的假设条件，成功建立了涵盖多方面要素的数学模型。该模型的求解过程以优化目标为导向，**采用逆向思维**，先求解 PBS 最优出车序列。然后在此基础上，返求 PBS 优化调度方案，这大大简化了 PBS 调度问题。

（2）此外，在采用**模拟退火算法和遗传算法**求解最优出车序列的过程中，本文假设出车序列也应尽量满足从小到大排布，从而避免对进车序列的过度打乱，最终得以**节约大量时间**。

（3）从对附件 1 和附件 2 的应用情况来分析，建立的数学模型很好地满足了 4 个优化目标的要求，得分较高，并且在模型验证中，也能够将结果良好运算出，这说明了模型的可靠性。

8.2 模型不足

（1）在 PBS 的调度方案中，本文对各车道的优先级进行了排序，使得车身倾向于选择优先级较高的车道，这可能缩小了模型解集的范围，也就是降低了调度方案的多样性。

（2）对于不同进车道上车身等待时间的设置，在客观性和精确性上有所不足，还需深入挖掘。

（3）由于时间上的限制和知识层面的欠缺，题目给出的诸多约束并未完全考虑到，模型还需进一步完善。

九、 参考文献

[1] Boysen N, Flidner M, Scholl A. Sequencing mixed-model assembly lines: Survey, classification and model critique[J]. European Journal of Operational Research, 2009, 192(2): 349-373.

[2] Wester L，Kilbridge M. The assembly line model-mix sequencing problem[C].In：Proceedings of the Third International Conference on Operations Research. Paris, 1964.247-260.

[3] 庞博高雅. 汽车混流装配线排序研究[D].东北林业大学,2018.

[4] Goldberg D E. Genetic Algorithm in search[J]. Optimization and Machine Learning, 1989 (11): 3-26.

[5] Holand J.H. Adaptation in Natural and Artificial Systems [M].Massachusetts: MITPress.1975.

[6] 马永杰,云文霞.遗传算法研究进展[J].计算机应用研究,2012,29(04):1201-1206+1210.

[7] 孙宝凤,申琇秀,龙书玲,卢昭宇.混流装配线的双目标投产排序决策模型[J].计算机集成制造系统,2017,23(07):1481-1491.DOI:10.13196/j.cims.2017.07.013.

[8] 刘晓霞. 种群规模对遗传算法性能影响的研究[D].华北电力大学（河北）,2010.

[9] 刘佳楠. 基于遗传算法的混流装配线平衡与排产问题研究[D].青岛理工大学,2020.DOI:10.27263/d.cnki.gqudc.2020.000160.

[10] 武良丹,张小凤,贺西平.基于模拟退火算法的超声回波参数估计[J].应用声学,2007(05):313-317.

[11] 姜东,唐秋华,李梓响,吴玲.多目标模拟退火算法求解混装线平衡与排序[J].机械设计与制造,2018(09):189-192.DOI:10.19356/j.cnki.1001-3997.2018.09.051.

[12] 张九龙,王晓峰,芦磊,牛鹏飞,程亚南.改进的模拟退火算法求解规则可满足性问题[J].现代电子技术,2022,45(05):122-128.DOI:10.16652/j.issn.1004-373x.2022.05.021.

十、 附录

问题一的代码清单

```
%% 模拟退火
clc
clear
tic
data=xlsread('D:\DESKTOP\2022 数学建模\代码\附件 1.xlsx');
data(:,1:4)=[];
[a,b]=size(data);
n=a;

T0 = 100;
T = T0;
maxgen = 1000;
Lk = 500;
alpfa = 0.97;

path0 = randperm(n);
data_1=data(path0,:);
[result0,score0] = jisuan(data_1);

min_result = result0;
RESULT = zeros(maxgen,1);
score=[score0,];

for iter = 1 : maxgen
    for i = 1 : Lk
        path1 = gen_new_path(path0);
        data_2=data_1(path1,:);
        hundongchexuhao=find(data_2(:,2)==3);
        [a1,b1]=size(hundongchexuhao);
        k=0;
        for j=2:a1
            if hundongchexuhao(j)-hundongchexuhao(j-1)~=3;
                k=k+1;
            end
        end
        y1=0.4*k;

        [a2,b2]=size(data_2);
        I=[];
        for l=2:a2
            if data_2(l,3)~=data_2(l-1,3);
                I=[I,l];
            end
        end

        if data_2(1,3)==data_2(end,3)
```

```
        I=[1,I];
        I=I';
        I1=[0;I(1:end-1)];
        k2=1;
        I_2=I-I1;
        I_2(1)=[];
        [i2,j2]=size(I_2);
        z=1;
        for m=1:i2/2
            if I_2(z)~=I_2(z+1)
                k2=k2+1;
            end
            z=z+2;
        end
end
if data_2(1,3)~=data_2(end,3)

        k2=0;
        I=[1,I,data_2(end,3)];
        I=I';
        I1=[0;I(1:end-1)];
        I_2=I-I1;
        I_2(1)=[];
        [i2,j2]=size(I_2);
        z=1;
        for m=1:i2/2
            if I_2(z)~=I_2(z+1)
                k2=k2+1;
            end
            z=z+2;
        end
end

y2=0.3*k2;
%% 目标 3 计算
ss=1:1:n;
ss=ss';
chazhi=sum(abs(data_2(:,4)-ss))/500;
y3=chazhi;

y=y1+y2+y3;
%%
result1 = y;

if result1 < result0
        path0 = path1;
        result0 = result1;
else
        p = exp(-(result1 - result0)/T);
```

```
                if rand(1) < p
                    path0 = path1;
                    result0 = result1;
                end
            end

            if result1 < min_result
                min_result = result1;
                haojieguo=data_2;
                best_path = path1;
                Z_Y=[y1,y2,y3];
            end
        end
        RESULT(iter) = min_result;
        T = alpfa*T;
        score=[score,Z_Y(1)+Z_Y(2)];
end
path1= path1';
best_path=best_path';
zuiyoujie=min_result;
zuiyoupaixu=haojieguo;
toc
y1= Z_Y(1);
y2= Z_Y(2);


z=0;
kong_1=[];
for i=1:n
        if haojieguo(i,4)>z
            kong_1=[kong_1;i];
            z=haojieguo(i,4);
        end

end
diyilie=haojieguo(kong_1,:);
[l1,~]=size(diyilie);
zhuan_1=haojieguo;
zhuan_1(kong_1,:)=[];


[ll,kk]=size(zhuan_1);
z=0;
kong_2=[];
for i=1:ll
        if zhuan_1(i,4)>z
            kong_2=[kong_2;i];
            z=zhuan_1(i,4);
        end
end
```

```
dierlie=zhuan_1(kong_2,:);
[l2,~]=size(dierlie);
zhuan_2=zhuan_1;
zhuan_2(kong_2,:)=[];


[pl,~]=size(zhuan_2);
z=0;
kong_3=[];
for i=1:pl
    if zhuan_2(i,4)>z
        kong_3=[kong_3;i];
        z=zhuan_2(i,4);
    end
end
disanlie=zhuan_2(kong_3,:);
[l3,~]=size(disanlie);
zhuan_3=zhuan_2;
zhuan_3(kong_3,:)=[];


[pk,~]=size(zhuan_3);
z=0;
kong_4=[];
for i=1:pk
    if zhuan_3(i,4)>z
        kong_4=[kong_4;i];
        z=zhuan_3(i,4);
    end
end
disilie=zhuan_3(kong_4,:);
[l4,~]=size(disilie);
zhuan_4=zhuan_3;
zhuan_4(kong_4,:)=[];


[pz,~]=size(zhuan_4);
z=0;
kong_5=[];
for i=1:pz
    if zhuan_4(i,4)>z
        kong_5=[kong_5;i];
        z=zhuan_4(i,4);
    end
end
diwulie=zhuan_4(kong_5,:);
[l5,~]=size(diwulie);
zhuan_5=zhuan_4;
zhuan_5(kong_5,:)=[];
```

```
[px,~]=size(zhuan_5);
z=0;
kong_6=[];
for i=1:px
    if zhuan_5(i,4)>z
        kong_6=[kong_6;i];
        z=zhuan_5(i,4);
    end
end
diliulie=zhuan_5(kong_6,:);
[l6,~]=size(diliulie);
zhuan_6=zhuan_5;
zhuan_6(kong_6,:)=[];


d_fanhuidao=zhuan_6;
[lf,~]=size(d_fanhuidao);
T=[l1,l2,l3,l4,l5,l6];
T=sort(T);
zongtime=T(1)*(2*18)+T(2)*(2*18)+T(3)*(2*12)+T(4)*(2*12)+T(5)*(2*6)+lf*(2*6)+9*n+7
2;


tongdao_3=diyilie(:,4);
tongdao_4=dierlie(:,4);
tongdao_2=disanlie(:,4);
tongdao_5=disilie(:,4);
tongdao_1=diwulie(:,4);
tongdao_6=diliulie(:,4);
tongdao_7=d_fanhuidao(:,4);
% d1=diyilie(:,4);
% d2=dierlie(:,4);
% d3=disanlie(:,4);
% d4=disilie(:,4);
% d5=diwulie(:,4);
% d6=diliulie(:,4);
% d7=d_fanhuidao(:,4);
zuijiaxuhao=haojieguo(:,4);
dajuzhen=zeros(n,zongtime);
zzz1=0;
zzz2=0;
zzz3=0;
zzz4=0;
zzz5=0;
zzz6=0;
zzz7=0;
ZK=[];
AAAAAAA=1;
SS=[];
```

```
for i=1:10
    dajuzhen(1,(i-1)*9+1:i*9)=str2double(sprintf('%d%d', 4, 11-i));
end
for i=1:10
    dajuzhen(2,(i-1)*9+4:i*9+4)=str2double(sprintf('%d%d',3, 11-i));
end
dajuzhen(2,1:3)=1;
dajuzhen(2,i*9+5:i*9+7)=2;

zzk=0;
shangyigeshijian=0;


for i=3:n
    ts1=length(find(tongdao_1<i));
    ts2=length(find(tongdao_2<i));
    ts3=length(find(tongdao_3<i));
    ts4=length(find(tongdao_4<i));
    ts5=length(find(tongdao_5<i));
    ts6=length(find(tongdao_6<i));
    ts7=length(find(tongdao_7<i));
    aa1=length(find(tongdao_1==i));
    aa2=length(find(tongdao_2==i));
    aa3=length(find(tongdao_3==i));
    aa4=length(find(tongdao_4==i));
    aa5=length(find(tongdao_5==i));
    aa6=length(find(tongdao_6==i));
    aa7=length(find(tongdao_7==i));
    TTT=[aa1,aa2,aa3,aa4,aa5,aa6,aa7];
    AAA=[1,2,3,4,5,6,7];
    ZZZ=max(TTT.*AAA);
    ZK=[ZK;ZZZ];

    if ZZZ==1||ZZZ==6
        qianzhui=18;
    elseif ZZZ==2||ZZZ==5
        qianzhui=12;
    elseif ZZZ==3||ZZZ==7
        qianzhui=6;
    elseif ZZZ==4
        qianzhui=0;
    end

    if ZZZ==4
        zzz4=zzz4+1;
        Q=zzz4;
    end
    if ZZZ==3
        zzz3=zzz3+1;
        Q=zzz3;
```

```
    end
    if ZZZ==7
        zzz7=zzz7+1;
        Q=zzz7;
    end
    ou=9;
    % if i==1
    %     opk=0;
    % end
    %     if i~=1&&qianzhui<9        %&&ZK(i)==ZK(i-1)
    %         opk=9;
    % %       else
    % %           opk=-qianzhui+qianzhui/2;
    %     end
    WWW=i-2;

shijianqidian=ts1*18+ts2*12+ts3*6+ts4*0+ts5*6+ts6*12+ts7*18+1+qianzhui/2+(     WWW-
1)*9;
    if  shijianqidian<Q*9+qianzhui/2+1;
        shijianqidian=Q*9+qianzhui/2+1;
    end
    %shijianzhongjian=ts1*18+ts2*12+ts3*6+ts4*0+ts5*6+ts6*12+ts7*18+1+90-1;
    shijianzhongjian=shijianqidian+90;
    dajuzhen(i,shijianqidian:(shijianqidian+ou-1))=str2double(sprintf('%d%d', ZZZ, 10));
    dajuzhen(i,shijianqidian+ou:(shijianqidian+2*ou-1))=str2double(sprintf('%d%d',     ZZZ,
9));
    dajuzhen(i,shijianqidian+2*ou:(shijianqidian+3*ou-1))=str2double(sprintf('%d%d',  ZZZ,
8));
    dajuzhen(i,shijianqidian+3*ou:(shijianqidian+4*ou-1))=str2double(sprintf('%d%d',  ZZZ,
7));
    dajuzhen(i,shijianqidian+4*ou:(shijianqidian+5*ou-1))=str2double(sprintf('%d%d',  ZZZ,
6));
    dajuzhen(i,shijianqidian+5*ou:(shijianqidian+6*ou-1))=str2double(sprintf('%d%d',  ZZZ,
5));
    dajuzhen(i,shijianqidian+6*ou:(shijianqidian+7*ou-1))=str2double(sprintf('%d%d',  ZZZ,
4));
    dajuzhen(i,shijianqidian+7*ou:(shijianqidian+8*ou-1))=str2double(sprintf('%d%d',  ZZZ,
3));
    dajuzhen(i,shijianqidian+8*ou:(shijianqidian+9*ou-1))=str2double(sprintf('%d%d',  ZZZ,
2));
    dajuzhen(i,shijianqidian+9*ou:(shijianqidian+10*ou-1))=str2double(sprintf('%d%d',
ZZZ, 1));
    if i==1
        dajuzhen(i,(shijianzhongjian+qianzhui/2+1):zongtime)=3;
    end
    if i~=1
        dajuzhen(i,(shijianqidian-qianzhui/2):(shijianqidian-1))=1;
        dajuzhen(i,(shijianzhongjian):(shijianzhongjian+qianzhui/2))=2;
        dajuzhen(i,(shijianzhongjian+qianzhui/2):zongtime)=3;
    end
```

```
        zzk=ZZZ;
        shangyigeshijian=shijianqidian;
end
[zzzz,klkl]=size(dajuzhen);
for i=1:klkl
        if dajuzhen(end,i)==3;
                lk=i;
                break
        end
end
dajuzhen(:,lk:end)=[];
for i=1:2
        for j=1:(lk-1)
                if dajuzhen(i,j)==0;
                        dajuzhen(i,j)=3;
                end
        end
end

[aaaa,bbbb]=size(d_fanhuidao);
y4=0.2*aaaa;
y5=0.1*(0.01*(lk-9*n-72));
zuizhongpingfen=100-(y1+y2+y4+y5);


xlswrite('D:\DESKTOP\2022 数学建模\代码\结果文件\测试结果\1.xlsx',dajuzhen);
```

生成新路径函数文件的代码清单

```
function path1 = gen_new_path(path0)
% path0:  原来的路径
n = length(path0);

p1 = 0.33;
p2 = 0.33;
r = rand(1);
if    r< p1
        c1 = randi(n);
        c2 = randi(n);
        path1 = path0;
        path1(c1) = path0(c2);
        path1(c2) = path0(c1);
elseif r < p1+p2
        c1 = randi(n);
        c2 = randi(n);
        c3 = randi(n);
        sort_c = sort([c1 c2 c3]);
        c1 = sort_c(1);    c2 = sort_c(2);    c3 = sort_c(3);    % c1 < = c2 <=    c3
        tem1 = path0(1:c1-1);
```

```
        tem2 = path0(c1:c2);
        tem3 = path0(c2+1:c3);
        tem4 = path0(c3+1:end);
        path1 = [tem1 tem3 tem2 tem4];
else
        c1 = randi(n);
        c2 = randi(n);
        if c1>c2
                tem = c2;
                c2 = c1;
                c1 = tem;
        end
        tem1 = path0(1:c1-1);
        tem2 = path0(c1:c2);
        tem3 = path0(c2+1:end);
        path1 = [tem1 fliplr(tem2) tem3];
end
end
```

目标函数的代码清单

```
function [y,score0]=jisuan(x)

data=x;
hundongchexuhao=find(data(:,2)==3);
[a,b]=size(data);
n=a;
[a1,b1]=size(hundongchexuhao);
k=0;
for i=2:a1
        if hundongchexuhao(i)-hundongchexuhao(i-1)~=3;
                k=k+1;
        end
end
y1=0.4*k;

[a2,b2]=size(data);
I=[];
for i=2:a2
        if data(i,3)~=data(i-1,3);
                I=[I,i];
        end
end

if data(1,3)==data(end,3)

        I=[1,I];
        I=I';
        I1=[0;I(1:end-1)];
        k2=1;
        I_2=I-I1;
```

```
        I_2(1)=[];
        [i2,j2]=size(I_2);
        z=1;
        for i=1:i2/2
            if I_2(z)~=I_2(z+1)
                k2=k2+1;
            end
            z=z+2;
        end
end
if data(1,3)~=data(end,3)

        k2=0;
        I=[1,I,data(end,3)];
        I=I';
        I1=[0;I(1:end-1)];
        I_2=I-I1;
        I_2(1)=[];
        [i2,j2]=size(I_2);
        z=1;
        for i=1:i2/2
            if I_2(z)~=I_2(z+1)
                k2=k2+1;
            end
            z=z+2;
        end
end
y2=0.3*k2;

ss=1:1:n;
ss=ss';
chazhi=sum(abs(data(:,4)-ss))/500;
y3=chazhi;
y=y1+y2+y3;
score0=y1+y2;
end
```

遗传算法的代码清单

```
clc
clear
data=xlsread('D:\DESKTOP\2022 数学建模\代码\附件 1.xlsx');
maxgen=1000;
pct=0.9;
pmt=0.1;
w1=0.4;w2=0.3;w3=0.2; w4=0.1;
[pop]=initpop(data);
[F]=calobjvalue(pop);
[fitvalue]=calfitvalue(F);
bestindividual=zeros(maxgen,data);
```

```
bestobjvalue=zeros(maxgen,1);
bestfit=zeros(maxgen,1);
generation=1;
while   generation<=maxgen
      sumf=sum(fitvalue);
      Zavg=sumf/popsize;
      [Zmax,fmax]=max(fitvalue);
      pcc=pct*2^(-10*(Zmax-Zavg)/Zmax);
      pmm=pmt*exp(-10*(Zmax-Zavg)/Zmax);
      if   pcc>=0.5
            pc=pcc;
      else
            pc=0.5;
      end
      if    pmm>=0.01
            pm=pmm;
      else
            pm=0.01;
      end
      [L]=selection(pop,fitvalue);
      [N]=crossover(L,pc);
      [O]=mutation(N,pm);
      pop=O;
      [F]=calobjvalue(pop);
      pxx=length(F);
      for   i=1:pxx
            fitvalue(i)=1./F(i);
      end
      [px,py]=size(pop);
      bestindividual(generation,:)=pop(1,:);
      bestfit(generation)=fitvalue(1);
      bestobjvalue(generation)=F(1);
      for   i=2:px
            if   fitvalue(i)>bestfit(generation)
                  bestindividual(generation,:)=pop(i,:);
                  bestfit(generation)=fitvalue(i);
                  bestobjvalue(generation)=F(i);
            end
      end
      generation=generation+1;
end
z=0;
kong_1=[];
for i=1:n
      if haojieguo(i,4)>z
            kong_1=[kong_1;i];
            z=haojieguo(i,4);
      end

end
```

```
diyilie=haojieguo(kong_1,:);
[l1,~]=size(diyilie);
zhuan_1=haojieguo;
zhuan_1(kong_1,:)=[];


[ll,kk]=size(zhuan_1);
z=0;
kong_2=[];
for i=1:ll
    if zhuan_1(i,4)>z
        kong_2=[kong_2;i];
        z=zhuan_1(i,4);
    end
end
dierlie=zhuan_1(kong_2,:);
[l2,~]=size(dierlie);
zhuan_2=zhuan_1;
zhuan_2(kong_2,:)=[];


[pl,~]=size(zhuan_2);
z=0;
kong_3=[];
for i=1:pl
    if zhuan_2(i,4)>z
        kong_3=[kong_3;i];
        z=zhuan_2(i,4);
    end
end
disanlie=zhuan_2(kong_3,:);
[l3,~]=size(disanlie);
zhuan_3=zhuan_2;
zhuan_3(kong_3,:)=[];



[pk,~]=size(zhuan_3);
z=0;
kong_4=[];
for i=1:pk
    if zhuan_3(i,4)>z
        kong_4=[kong_4;i];
        z=zhuan_3(i,4);
    end
end
disilie=zhuan_3(kong_4,:);
[l4,~]=size(disilie);
zhuan_4=zhuan_3;
zhuan_4(kong_4,:)=[];
```

```
[pz,~]=size(zhuan_4);
z=0;
kong_5=[];
for i=1:pz
    if zhuan_4(i,4)>z
        kong_5=[kong_5;i];
        z=zhuan_4(i,4);
    end
end
diwulie=zhuan_4(kong_5,:);
[l5,~]=size(diwulie);
zhuan_5=zhuan_4;
zhuan_5(kong_5,:)=[];



[px,~]=size(zhuan_5);
z=0;
kong_6=[];
for i=1:px
    if zhuan_5(i,4)>z
        kong_6=[kong_6;i];
        z=zhuan_5(i,4);
    end
end
diliulie=zhuan_5(kong_6,:);
[l6,~]=size(diliulie);
zhuan_6=zhuan_5;
zhuan_6(kong_6,:)=[];



d_fanhuidao=zhuan_6;
[lf,~]=size(d_fanhuidao);
T=[l1,l2,l3,l4,l5,l6];
T=sort(T);
zongtime=T(1)*(2*18)+T(2)*(2*18)+T(3)*(2*12)+T(4)*(2*12)+T(5)*(2*6)+lf*(2*6)+9*n+7
2;


tongdao_3=diyilie(:,4);
tongdao_4=dierlie(:,4);
tongdao_2=disanlie(:,4);
tongdao_5=disilie(:,4);
tongdao_1=diwulie(:,4);
```

```
tongdao_6=diliulie(:,4);
tongdao_7=d_fanhuidao(:,4);

zuijiaxuhao=haojieguo(:,4);
dajuzhen=zeros(n,zongtime);
zzz1=0;
zzz2=0;
zzz3=0;
zzz4=0;
zzz5=0;
zzz6=0;
zzz7=0;
ZK=[];
AAAAAAA=1;
SS=[];
for i=1:10
    dajuzhen(1,(i-1)*9+1:i*9)=str2double(sprintf('%d%d', 4, 11-i));
end
for i=1:10
    dajuzhen(2,(i-1)*9+4:i*9+4)=str2double(sprintf('%d%d',3, 11-i));
end
dajuzhen(2,1:3)=1;
dajuzhen(2,i*9+5:i*9+7)=2;

zzk=0;
shangyigeshijian=0;


for i=3:n
    ts1=length(find(tongdao_1<i));
    ts2=length(find(tongdao_2<i));
    ts3=length(find(tongdao_3<i));
    ts4=length(find(tongdao_4<i));
    ts5=length(find(tongdao_5<i));
    ts6=length(find(tongdao_6<i));
    ts7=length(find(tongdao_7<i));
    aa1=length(find(tongdao_1==i));
    aa2=length(find(tongdao_2==i));
    aa3=length(find(tongdao_3==i));
    aa4=length(find(tongdao_4==i));
    aa5=length(find(tongdao_5==i));
    aa6=length(find(tongdao_6==i));
    aa7=length(find(tongdao_7==i));
    TTT=[aa1,aa2,aa3,aa4,aa5,aa6,aa7];
    AAA=[1,2,3,4,5,6,7];
    ZZZ=max(TTT.*AAA);
    ZK=[ZK;ZZZ];

    if ZZZ==1||ZZZ==6
        qianzhui=18;
```

```
    elseif ZZZ==2||ZZZ==5
        qianzhui=12;
    elseif ZZZ==3||ZZZ==7
        qianzhui=6;
    elseif ZZZ==4
        qianzhui=0;
    end

    if ZZZ==4
        zzz4=zzz4+1;
        Q=zzz4;
    end
    if ZZZ==3
        zzz3=zzz3+1;
        Q=zzz3;
    end
    if ZZZ==7
        zzz7=zzz7+1;
        Q=zzz7;
    end
    ou=9;

    WWW=i-2;

shijianqidian=ts1*18+ts2*12+ts3*6+ts4*0+ts5*6+ts6*12+ts7*18+1+qianzhui/2+(    WWW-
1)*9;
    if  shijianqidian<Q*9+qianzhui/2+1;
        shijianqidian=Q*9+qianzhui/2+1;
    end
    %shijianzhongjian=ts1*18+ts2*12+ts3*6+ts4*0+ts5*6+ts6*12+ts7*18+1+90-1;
    shijianzhongjian=shijianqidian+90;
    dajuzhen(i,shijianqidian:(shijianqidian+ou-1))=str2double(sprintf('%d%d', ZZZ, 10));
    dajuzhen(i,shijianqidian+ou:(shijianqidian+2*ou-1))=str2double(sprintf('%d%d',    ZZZ,
9));
    dajuzhen(i,shijianqidian+2*ou:(shijianqidian+3*ou-1))=str2double(sprintf('%d%d',  ZZZ,
8));
    dajuzhen(i,shijianqidian+3*ou:(shijianqidian+4*ou-1))=str2double(sprintf('%d%d',  ZZZ,
7));
    dajuzhen(i,shijianqidian+4*ou:(shijianqidian+5*ou-1))=str2double(sprintf('%d%d',  ZZZ,
6));
    dajuzhen(i,shijianqidian+5*ou:(shijianqidian+6*ou-1))=str2double(sprintf('%d%d',  ZZZ,
5));
    dajuzhen(i,shijianqidian+6*ou:(shijianqidian+7*ou-1))=str2double(sprintf('%d%d',  ZZZ,
4));
    dajuzhen(i,shijianqidian+7*ou:(shijianqidian+8*ou-1))=str2double(sprintf('%d%d',  ZZZ,
3));
    dajuzhen(i,shijianqidian+8*ou:(shijianqidian+9*ou-1))=str2double(sprintf('%d%d',  ZZZ,
2));
    dajuzhen(i,shijianqidian+9*ou:(shijianqidian+10*ou-1))=str2double(sprintf('%d%d',
ZZZ, 1));
```

```
    if i==1
        dajuzhen(i,(shijianzhongjian+qianzhui/2+1):zongtime)=3;
    end
    if i~=1
        dajuzhen(i,(shijianqidian-qianzhui/2):(shijianqidian-1))=1;
        dajuzhen(i,(shijianzhongjian):(shijianzhongjian+qianzhui/2))=2;
        dajuzhen(i,(shijianzhongjian+qianzhui/2):zongtime)=3;
    end
    zzk=ZZZ;
    shangyigeshijian=shijianqidian;
end
[zzzz,klkl]=size(dajuzhen);
for i=1:klkl
    if dajuzhen(end,i)==3;
        lk=i;
        break
    end
end
dajuzhen(:,lk:end)=[];
for i=1:2
    for j=1:(lk-1)
        if dajuzhen(i,j)==0;
            dajuzhen(i,j)=3;
        end
    end
end

[aaaa,bbbb]=size(d_fanhuidao);
zuizhongpingfen=100-(y1+y2+0.2*aaaa+0.1*(0.01*(lk-9*n-72)));
```