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	南京师范大学
参赛队号	22103190082
队员姓名	1.薛亦晖
	2.袁浩淼
	3.丁乙

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 **汽车制造涂装-总装缓存调序区调度优化研究**

摘　　要：

汽车制造涂装-总装缓存调序区（PBS）在汽车制造厂中可以帮助涂装车间的出车序列进行调整，以更好的满足总装车间需要的进车队列，同时提高生产效率。车间调度问题是对实际生产过程进行高度抽象化得到的一类最优化的数学问题，其本质为对不同的序列通过线性流进行合理的重排，使生产效益最大化。由于该问题无法直接求出最优解，且无法直接判断所求解是否为最优解，因此我们下面将其作为 NP-hard 问题来处理。

针对**问题 1**，我们需要根据题中所给约束和时间数据来建立 PBS 约束条件。我们利用 MATLAB 对 PBS 的操作程序创造性地进行**动态仿真模拟**，并建立**单目标的 0-1 整数规划模型**来进行描述，形成问题 1 的 PBS 优化调度模型。我们接着采用**改进的遗传算法**来找最优解。为了减少迭代次数和运算时间，我们需要先找到一个较好的初始种群。所以我们利用**启发式算法**寻找初始值，根据实际生产经验提出了 **4 种调度策略**，包括设置不同车型进入的车道序号、使用返回道的概率（0.05 和 0.001），以及利用**贪心算法**考虑出车序列是否优先满足优化目标 1 的情况。通过计算发现，设置不同驱动类型车身进入相应进车道得到的分数较高，即**混动型车身进入第 1、2、3、4、5、6 进车道，燃油型车身进入第 3、4 条进车道**。因此我们利用这种策略来约束遗传算法中的车道选择序列，并利用**混沌序列**找到染色体的交叉点，进行**单点交叉**，后来又加入**单点变异**以防算法早熟。附件 1 优化结果**总得分为 26.91，各优化目标得分分别为-70,82,100,103.12**，说明该结果不仅较为满足生产要求，生产时间也较短，实现了生产效益最大化。再利用附件 2 数据测试模型和算法的适应性，结果**总得分为 53.04，各优化目标得分分别为-9、82、100、104.41**，而此时全走 4 车道的优化得分仅为 35.1，提高较大，说明我们优化模型的有效性和鲁棒性均较高。

针对**问题 2**，仅是减少了问题 1 中的两条约束，求解的可行域更大。我们继续对问题 1 中启发式算法探索出的最优策略与遗传算法进行调整与改进，在问题 1 目标规划模型和仿真模拟程序的基础上，我们对调序策略**加上对不满足生产序列的燃油车型进入返回道的概率 0.95**，以及**返回道 1 停车位被接车横移机下一个处理的概率 0.5** 的约束。继续利用问题 1 中同时进行单点交叉和变异的思路，得到附件 1 的优化结果**总得分为 28.72，各优化目标得分分别为-63、82、98、97.21**，相比于问题 1 的优化结果提高约 2 分。再利用附件 2 数据测试模型和算法的适应性，结果**总得分为 57.30，各优化目标得分分别为 15、85、84、90.04**，相比于问题 1 的优化结果也提高较大，说明我们优化模型的有效性和鲁棒性较高。

本文模型的**创新点**在于（1）选取多种调度策略，比较得到较优良的初始化种群；（2）灵活使用混沌序列，构造科学合理的交叉点和变异点；（3）创新性地提出 PBS 仿真程序，精准模拟车身调度状态。（4）在算法中，巧妙利用矩阵算子$Position$和$Time$及矩阵的初等变换描述每个时刻车身位置及状态。

关键词：PBS 优化调度模型；PBS 仿真模拟；启发式算法；改进的遗传算法；整数规划

一、问题重述

1.1 问题背景

我国汽车制造业柔性化水平及生产规模与发达国家还存在较大差距，导致这种差距的原因除了设备的自动化程度和操作者的熟练程度等因素之外，制定的车间生产计划情况也是重要因素。制定合理有效的车间生产计划对于进一步提高汽车企业的生产效率、有效地降低产品成本具有重要意义。

汽车制造厂主要由焊装车间、涂装车间、总装车间构成，每个车间有不同的生产偏好，由于各车间的约束不同导致生产调度无法按照同一序列连续生产，特别是涂装车间与总装车间序列差异较大，这就需要在两个车间之间建立一个具有调序功能的缓存区，即 PBS（Painted Body Store，汽车制造涂装-总装缓存调序区），用来将涂装车间的出车序列调整到满足总装车间约束的进车序列。一天安排上线生产的车辆数介于 200-450 之间，对于每天要上线生产的车辆，车型、颜色等属性均有变化，目前车型大类有 2 种，颜色大约有 10 种，各个车间的产能不定，主要根据当天生产安排调整，从涂装车间喷漆之后出车，通过 PBS 调度调整后进入总装车间。PBS 安置有多条进车道和一条返回道，以及接送车横移机。每条进车道有 10 个停车位，为 FIFO 结构（先进先出）。

在考虑 PBS 调序时间更短的同时，要让进出车间的生产偏好尽可能地满足，从而优化 PBS 的调度，提高混流装配的生产效率。

1.2 问题描述

在该案例背景下，需要优化的目标主要有如下四个：（1）混动车型间隔 2 台非混动车型为优，权重系数 0.4；（2）四驱车型与两驱车型倾向 1:1 出车序列，权重系数 0.3；（3）返回道使用次数倾向于 0，权重系数 0.2；（4）倾向总调度时间越短越好，权重系数 0.1。

PBS 缓冲区的调度需要借助接车横移机和送车横移机来进行。接车横移机可将当前涂装-PBS 出车口队列第一个车身或者返回道 10 停车位的车身送入任意进车道的 10 停车位。送车横移机可将任意进车道 1 停车位的车送入返回道 1 停车位或 PBS-总装借车口。这两种横移机均可在一段时间内被设置为空闲状态。

但 PBS 缓冲区的调度需要满足如下的约束条件：

（1）送车横移机不能将返回道的车身送入 PBS-总装接车口。

（2）车身在进车道和返回道的移动方向为图中标注方向，不得改变。

（3）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。

（4）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。

（5）接车横移机和送车横移机在执行任何动作过程中，均不能被打断。

（6）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。

（7）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。

（8）如果任意进车道 1 停车位有车身，那么送车横移机不能设置为空闲状态。

（9）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。

（10）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。

（11）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。
（12）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。

1.3 问题提出

从上述背景和描述不难看出，PBS 调度优化问题不仅涉及到上游生产车间的生产计划，还影响着下游装配车间的计划进度。随着车型种类和车道等约束条件的增多，寻求最优解成了一个$NP-hard$问题。为此，本文采用基于实践的启发式算法寻求较优初始种群，并利用改进的遗传算法找寻不同约束条件下的最优解。主要包括以下两个问题：

问题 1：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，对 PBS 中每个时刻的车身状态变化进行模拟仿真，使得出车序列尽可能满足总装生产需求，即使得优化目标得分尽可能地高。

问题 2：在问题 1 地基础上，删去 PBS 约束说明中第 6、7 两条对接送横移机执行任务的优先级设置，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，使得出车序列尽可能满足总装生产需求，即使得优化目标得分尽可能地高。

二、模型假设与符号说明

2.1 模型假设

1. 每个车身在 PBS 的调度中，最多只经过一次返回通道；
2. 同一车道前后两个车身可以同时开始往下一个停车位移动；
3. 当一个车身向下一个停车位移动但还未到达时，车身区域代码不变；
4. 优化目标 1，2，3 的得分可以小于 0，目标 4 的得分可以大于 100；
5. 每次调度都只考虑理论最快完成时间的移动方式。

2.2 符号定义

符号	符号说明
N_1	表示不满足每两个混动车型之间的燃油车型为 2 个的情况数量
N_2	表示不满足混动车型和燃油车型数量为 1：1 的块数
N_3	表示返回道使用次数
x_i	表示第i个优化目标的得分
y	表示混动型车身总数量
z	表示燃油型车身总数量
ω_i	表示第 i个优化目标所占的权重
$t_1(i)$	表示接车横移机将车送入第i个车道或送车横移机将车送出第i个车道用的时间
$t_2(i)$	表示送车横移机将第i个进车道的需调序车身送进返回道所花的时间
$t_3(i)$	表示接车横移机将返回道车身运入第i个进车道所花的时间
h	表示车身从一个停车位向下一个停车位移动的时
M	表示一维染色体数组
N	表示车身序号
P_1	表示不满足出车序列要求的燃油型车身进入返回道的概率
P_2	表示把返回车道车身优先接入进车道的概率
(i,j)	表示第i条车道的第j个停车位$(1 \le i \le 7, 1 \le j \le 10)$
$P_m^{(t)}$	表示第m辆车在第t时刻的坐标
$pos_{ij}^{(t)}$	表示t时刻第i条车道第 $11-j$个停车位的车身序号
$time_{ij}^{(t)}$	表示第i条车道第 $11-j$个停车位上车身开始移动的时间

$time_{max}^{(t)}$	表示t时刻 1-6 车道停车位 10 上车身停留的最长时间
λ_i	0-1 变量，表示车型是否为燃油型
μ_i	0-1 变量，表示车型是否为四驱
$\alpha_{ij}^{(t)}$	0-1 变量，表示t时刻(i,j)上有无车身
$\beta_k^{(t)}$	0-1 变量，表示t时刻横移机是否空闲
$Come_{ij}^{(t)}$	0-1 变量，表示t时刻(i,j)上是否从无车身变为有车身
$Go_{ij}^{(t)}$	0-1 变量，表示t时刻(i,j)上是否从有车身变为无车身
$Position^{(t}$	表示t时刻调度区的车身情况
$Time^{(t)}$	表示t时刻车身从上一个车位开始移动经过的时间
Q_{in}	表示进入PBS调度区的车身
$Rand_{in}$	表示随机进入PBS调度区的车身
PBS_{in}	表示每个车身所在进车道
$Rand_{out}$	表示随机装载到送车横移机上的车身
PBS_{out}	表示每个车身从进车道装载到送车横移机上的顺序

三、问题 1 优化模型的建立与求解

3.1 问题分析

318 个包含两种车型、两种动力和两种驱动类型的车身按照给定顺序进入 PBS 调度缓冲区，使得出车序列的车型分配更加理想。我们需要通过 PBS 的约束说明和时间数据说明增加约束条件，建立目标规划模型，并用算法模拟 PBS 在每时刻的状态。

为了最大程度地保证出车序列中每两个混动型车身之间包含两个燃油型车身，以及出车序列的分块中的混动型车身个数与燃油型车身个数保持相等，我们需要对出车序列进行检查与评分；在调度的过程中我们要尽量少地使用返回道，总调度时间也需要尽量地短。基于对这些目标评分权重的分析，找到启发式优化算法，在此基础上利用改进后的遗传算法求解模型，得到优化后的目标得分，同时输出整个队列从第一个车身进入 PBS 到最后一个车身进入 PBS 总装区过程中个每一秒的位置。

3.2 目标规划模型的建立

（1）设置目标函数

我们给 318 辆车的动力和驱动类型进行赋值，分别设置为 0-1 变量。第i个车身的动力

$$\lambda_i=\begin{cases}0,\ 第i个车身为混动型,\\ 1,\ 第i个车身为燃油型.\end{cases}$$

第i个车身的驱动

$$\mu_i=\begin{cases}0,\ 第i个车身为两驱型,\\ 1,\ 第i个车身为四驱型.\end{cases}$$

出车序列的车身的动力类型和驱动类型均成为一个由 0 和 1 构成的序列。

根据优化目标 1，对上述描述车身动力类型的出车序列进行遍历检查，对不满足每两个混动车身之间的燃油车身为 2 个的情况进行计数，即出现下述三种情况将进行计数：

① $\lambda_i=\lambda_{i+1}=1;$

② $\lambda_i=\lambda_{i+2}=1,\ \lambda_{i+1}=0;$

③ $\lambda_i=1,\ \lambda_{i+1}=\lambda_{i+2}=\ \lambda_{i+3}=0;$

再从 100 分中扣除，得到目标 1 的得分。

$$x_1=100-N_1 \tag{1}$$

根据优化目标 2，对上述描述车身驱动类型的出车序列进行遍历检查，进行分块，记录每一块的车身个数以及两驱和四驱车型的个数，若不相等，则计入扣分，再从 100 分中扣除，得到目标 2 的得分。

$$x_2=100-N_2 \tag{2}$$

根据优化目标 3，对模拟 PBS 程序中返回道的使用次数进行计数，再从 100 分中扣除，得到目标 3 的得分：

$$x_3=100-N_3 \tag{3}$$

根据优化目标 4，计算最后一个车身从区域 3 到区域 4 的时刻T，放入时间惩罚值公式$0.01\times(T-9C-72)$中计算，其中$C=318$，得到：

$$x_4=70.66-0.01T \tag{4}$$

再对上述 4 个优化目标的得分公式（1）（2）（3）（4）进行加权，得到优化目标函数：

$$max\ f = \sum_{i=1}^{4} \omega_i x_i\,, \tag{5}$$

其中$\omega_i \in \{0.4, 0.3, 0.2, 0.1\}$。

（2）约束条件

首先我们用数字 1-6 分别表示进车道 1-6，用数字 7 表示返回道，则可定义如下 4 个 0-1 变量：

$$\alpha_{ij}^{(t)} = \begin{cases} 0, & t\text{时刻}(i,j)\text{上无车身}, \\ 1, & t\text{时刻}(i,j)\text{上有车身}. \end{cases}$$

$$\beta_k^{(t)} = \begin{cases} 0\,, & \text{区域}k\text{有任务}, \\ 1\,, & \text{区域}k\text{无任务}. \end{cases}$$

$$Come_{ij}^{(t)} = \begin{cases} 0, & \text{其他情况}, \\ 1, & t\text{秒时}(i,j)\text{上从无车身道变为有车身}. \end{cases}$$

$$Go_{ij}^{(t)} = \begin{cases} 0, & \text{其他情况}, \\ 1, & t\text{秒时}(i,j)\text{上从有车身道变为无车身}. \end{cases}$$

其中，$k = 1,2$ 分别表示接车横移机和送车横移机的区域代码。

下面根据 PBS 约束说明增加约束条件：

i. 车身在进车道和返回道的移动方向固定，因此在进车道上若有一个停车位上无车身，则 9 秒后它的下一个车位也没有车身，均为 0。所以若$\alpha_{ij}^{(t)} = 0$, 则$\alpha_{i,(j-1)}^{(t+9)} = 0$，即

$$\alpha_{ij}^{(t)} + \alpha_{i,(j-1)}^{(t+9)} = 0,\ \ \forall i \in [1,6],\ j \in [2,10],\ i,j,t \in \mathbb{Z}_+ \tag{6}$$

当车身在返回道上时则有，若$\alpha_{7j}^{(t)} = 0$, 则$\alpha_{7,(j+1)}^{(t+9)} = 0$，即

$$\alpha_{7j}^{(t)} + \alpha_{7,(j+1)}^{(t+9)} = 0,\ \ j \in [1,9], j,t \in \mathbb{Z}_+ \tag{7}$$

ii. 接车横移机上同一时刻分别最多有一个车身,因此当$Come_{i,10}^{(t)} = 1$ 时，对任意$s \neq i$，$Come_{s,10}^{(t)} = 0$，即

$$Come_{i,10}^{(t)} + Come_{s,10}^{(t)} \leq 1,\ \ \forall t = 1,2,\ldots T, \forall s \neq i, i,s = 1,2,\ldots,6 \tag{8}$$

送车横移机上同一时刻也最多只有一个车身，因此

$$Go_{i,1}^{(t)} + Go_{s,1}^{(t)} \leq 1,\ \ \forall t = 1,2,\ldots T, \forall s \neq i, i,s = 1,2,\ldots,6. \tag{9}$$

iii. 若任意进车道的 1 停车位有车身，送车横移机就要开始执行任务，因此当$\exists i \in [1,6], \alpha_{i1}^{(t)} = 1$ 时，$\beta_2^{(t)}$必须为 1，则可以写成如下约束条件：

$$\beta_2^{(t)} \geq \alpha_{i1}^{(t)}, \forall i \in [1,6], i,t \in \mathbb{Z}_+. \tag{10}$$

iv. 当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个停车位移动。则有如下约束条件：

$$1-\alpha_{ij}^{(t)} \le Come_{ij}^{t+9}, \forall i \in [1,7], j \in [1,10], i,j,t \in \mathbb{Z}_+. \quad (11)$$

v. 由于车身在进车道和返回道的不同停车位之间移动时不能被调度，因此当车身在进车道上时，若$\alpha_{ij}^{(t)}=1$，则$Come_{ij}^{(t+9)}=0$，即

$$\alpha_{ij}^{(t)} - Come_{ij}^{(t+9)} = 1, \ \forall i \in [1,6], \ j \in [2,10], \ i,j,t \in \mathbb{Z}_+ \quad (12)$$

当车身在返回道上时，若$\alpha_{7j}^{(t-9)}=1$，则$Come_{7j}^{(t)}=0$，即

$$\alpha_{7j}^{(t-9)} - Come_{7j}^{(t)} = 1, \ \forall j \in [1,9], \ j,t \in \mathbb{Z}_+. \quad (13)$$

综合公式$(5)-(13)$，我们得到目标规划模型如下：

$$max\ f = 0.4x_1 + 0.3x_2 + 0.2x_2 + 0.1x_4$$

$$s.t. \begin{cases} \alpha_{ij}^{(t)} + \alpha_{i,(j-1)}^{(t+9)} = 0, & \forall i \in [1,6], j \in [2,10], i,j,t \in \mathbb{Z}_+, \\ \alpha_{7j}^{(t)} + \alpha_{7,(j+1)}^{(t+9)} = 0, & \forall j \in [1,9], j,t \in \mathbb{Z}_+, \\ Come_{i,10}^{(t)} + Come_{s,10}^{(t)} \le 1, & \forall t = 1,2,\dots T, \forall s \ne i, i,s = 1,2,\dots,6, \\ Go_{i,1}^{(t)} + Go_{s,1}^{(t)} \le 1, & \forall t = 1,2,\dots T, \forall s \ne i, i,s = 1,2,\dots,6, \\ \beta_2^{(t)} \ge \alpha_{i1}^{(t)}, & \forall i \in [1,6], i,t \in \mathbb{Z}_+, \\ 1-\alpha_{ij}^{(t)} \le Come_{ij}^{t+9}, & \forall i \in [1,7], j \in [1,10], i,j,t \in \mathbb{Z}_+, \\ \alpha_{ij}^{(t)} - Come_{ij}^{(t+9)} = 1, & \forall i \in [1,6], j \in [2,10], i,j,t \in \mathbb{Z}_+ \\ \alpha_{7j}^{(t-9)} - Come_{7j}^{(t)} = 1, & \forall j \in [1,9], j,t \in \mathbb{Z}_+. \end{cases}$$

其余 PBS 约束说明中的条件可以通过下面的程序模拟来实现，再用目标规划模型中的约束条件对仿真模拟中的情况进行检验，从而满足所有的 PBS 约束条件。

（3）PBS 仿真程序描述

STEP 1：分类确定进车道序号与在横移机上的时间。

由于接车横移机将车身从涂装出车口运送到第i个进车道与送车横移机将车身从第i个进车道运送到总装接车口时间相同，我们先将车身在接送车横移机上的时间与其随机进入的进车道建立一一对应关系[1]：$i \longmapsto t_1(i), i \in \{1,2,3,4,5,6\}$，接送车横移机进行这一任务所用时间为$2t_1(i)$。若要用送车横移机将车身送进返回道，也可以将这一过程的时间与需调序车所在进车道建立一一对应关系：$i \longmapsto t_2(i), i \in \{1,2,3,4,5,6\}$，送车横移机在这一过程中经历的时间为$t_1(i)+t_2(i)+6$。接车横移机将车身从返回道运入进车道的时间与随机选择的进车道序号也建立一一对应关系：$i \longmapsto t_3(i), i \in \{1,2,3,4,5,6\}$，接车横移机在这一过程中经历的时间为$t_1(i)+t_3(i)+6$。上述对应关系均分别存入二维的数组中，以便后续模拟接送车状态时进行调取。

STEP 2：模拟接车横移机的状态

首先根据是否有车身序号被标记在区域 1 来判断接车横移机是否空闲[2]，根据接车优先级的约束条件，判断返回道 10 停车位是否有车，若接车横移机空闲则首先接返回道的车。若返回道 10 停车位没有车身，则将未进入 PBS 的车身按序号先后随机选择停车位 10 为空的进车道送入，在PBS_{in}矩阵上标记车身序号。接车横移机再返回初始位置，经过$t_1(i)$秒后将接车横移机设置为空闲，再开始任务的循环，生成一个 318 个数的序列来表示每个车身进入的进车道序号。此步骤可以满足接车横移机在任务中不能被打断，且同一时刻最多只有一个车身。STEP 2 算法流程图大致如图 3-1 所示。

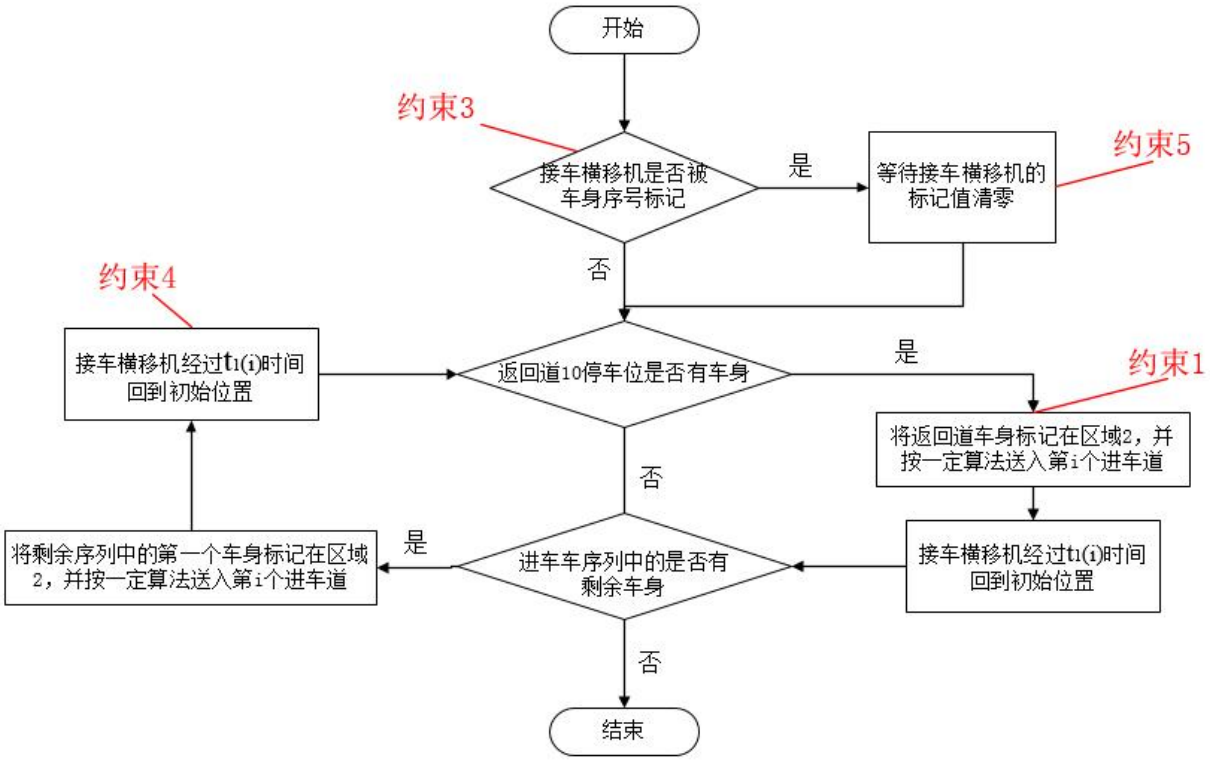

图 3-1 STEP 2 算法流程图

STEP 3: 模拟车身在进车道与返回道上的状态

对于每个时刻，我们用一个矩阵$Position^{(t)} = (pos_{ij}^{(t)})_{7\times10}$来存放处于此区域每个位置的车身的序号，即

$$\alpha_{i,j}^{(t)} = \frac{pos_{i,11-j}^{(t)}}{N}. \quad (14)$$

然后可以根据目标规划中的约束条件对车身位置进行检验，保证车身在进车道和返回道的移动方向固定，且移动过程中不会被调度。

在 6 条进车道上，若有车身所在停车位的下一个停车位为空，车身必须在 9 秒之后到达下一个车位，后面的车身也同时开始移动,即进行如下设定： 若$\alpha_{ij}^{(t-1)} = 1$且$\alpha_{ij}^{(t)} = 0$，则$time_{il}^{(t+h)} = h, h = 1,2,\ldots,8,\ time_{10-l}^{(t+9)} = 0, l = 1,2,\ldots,10-j$,且有$\delta_{i,j}^{(t+9)} = 1$。

而在返回道上，若有车身所在停车位的前一个停车位为空，车身必须在 9 秒之后到达前一个车位，后面的车也随之移动，则进行如下设定：$\alpha_{7j}^{(t-1)}=1$ 且 $\alpha_{7j}^{(t)}=0$，则$time_{il}^{(t+h)}=h, h=1,2,\ldots,8,\ time_{il}^{(t+9)}=0, l=12-j,\ldots,10,$ 且有$\delta_{ij}^{(t+9)}=1$。

在矩阵$Time^{(t)}=(time_{ij}^{(t)})_{7\times10}$中存放时间，$time_{ij}^{(t)}$与$pos_{ij}^{(t)}$一一对应，表示第$pos_{ij}^{(t)}$个车身从开始动经过的时间，到第 9 秒的时候，$pos_{ij}^{(t)}$发生变化.

由于我们这里设置的是空停车位后面的出同时开始移动，所以可以避免多个车身在一个停车位上的情况。由此显然，任意时刻每个进车道和返回道最大容纳 10 个车身。

STEP 3 算法流程图大致如图 3-2 所示。

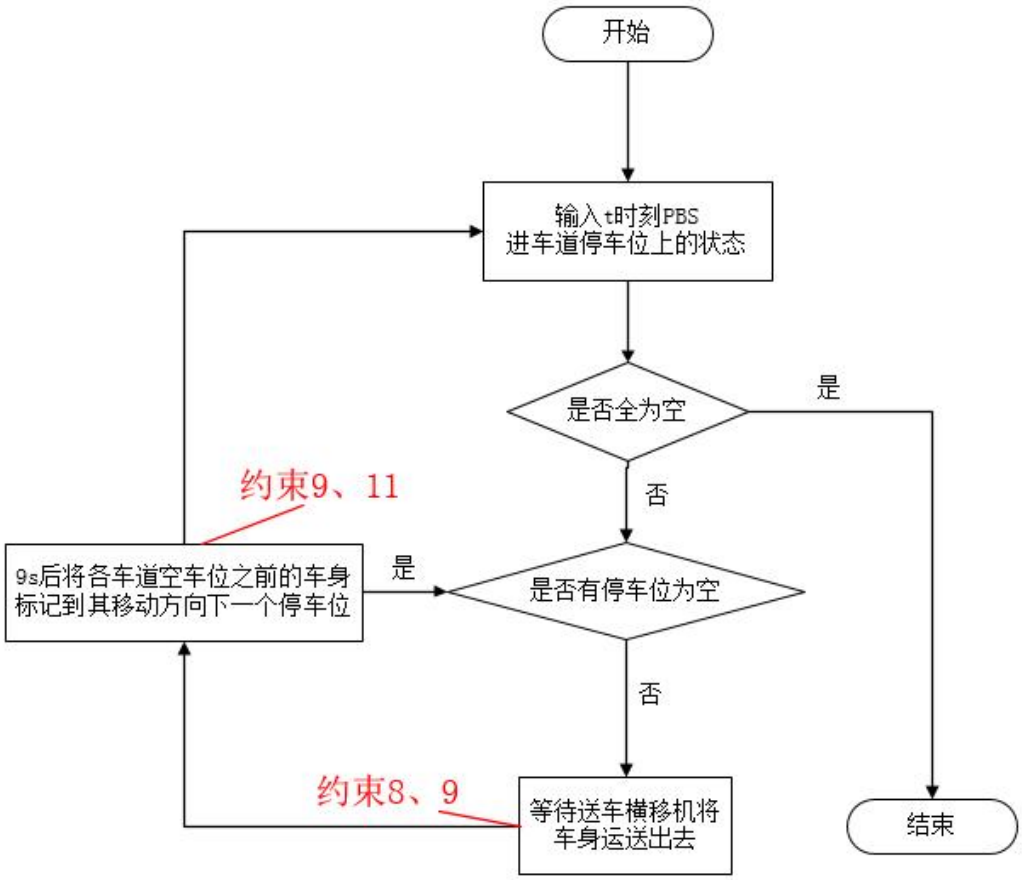

图 3-2 STEP 3 算法流程图

STEP 4: 模拟送车横移机的状态

首先根据是否有车身序号被标记在区域 2 来判断送车横移机是否空闲。考虑送车的优先级，等待时间最长的优先由送车横移机运走。我们通过比较$time_{ij}^{(t)}$，找到 6 条进车道停车位 10 上车身停留的最长时间$time_{max}^{(t)}$，再索引其车道序号i,将此车身序号标记在区域 2，此步骤保证了进车道停车位 10 有车身时，送车横移机不能空闲。

送车横移机按照我们设置的小概率将车身运进返回车道，再返回初始位置，其余等待$t_1(i)$后标记上送车横移机，送入 PBS 总装接车口后，接车横移机状态为空闲，再开始任务的循环[3]。此步骤可以满足送车横移机在任务中不能被打断，且同一时刻最多只有一个车身。由此我们生成一个 318 个数的序列来表示每个车身从 PBS 中出去的序列，算法结束。

STEP 4 算法流程图大致如图 3-3 所示。

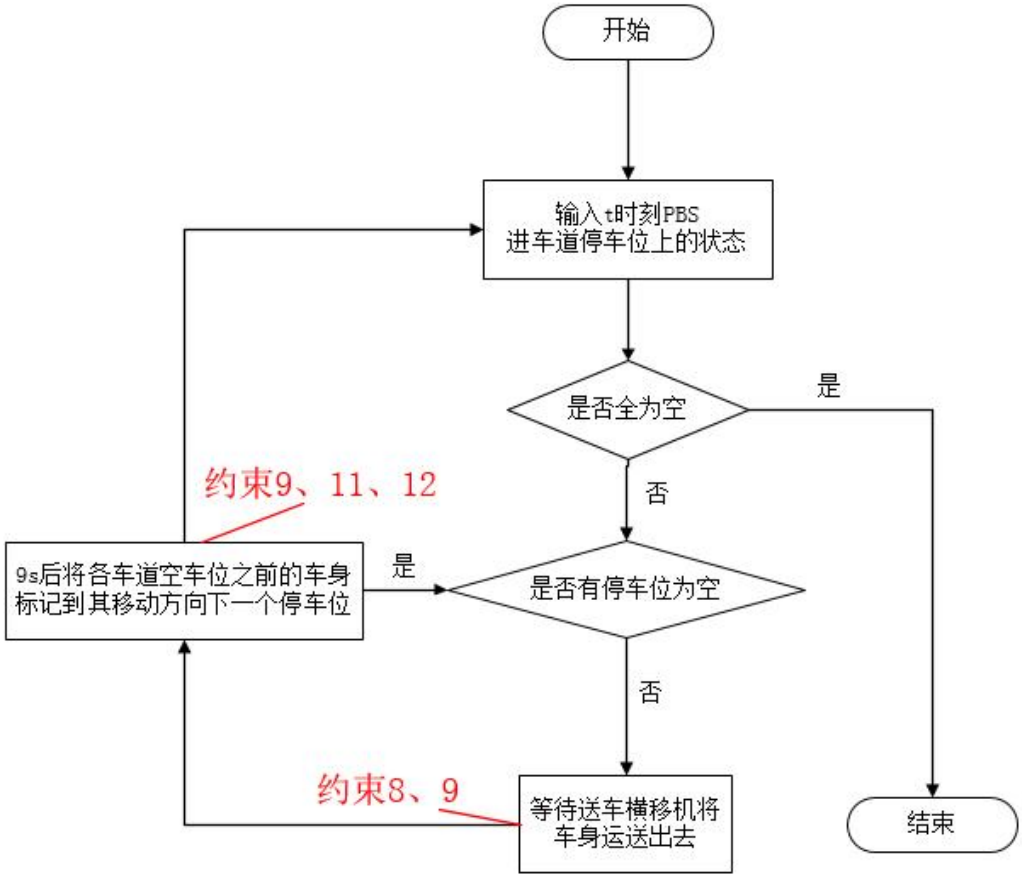

图 3-3 STEP 4 算法流程图

STEP 5: 生成出车序列及其车型序列

可以用矩阵相乘的形式表示车身序列的重新排序过程[4]，将进入 PBS 调度区的车身序列用 318×318 的对角矩阵表示为：

$$Q_{in}=\begin{pmatrix}1 & 0 & \cdots & 0\\ 0 & 2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & 318\end{pmatrix}$$

这个涂装出车序列按顺序进入进车道的道数用 6×318 的 0-1 矩阵表示，第i行第j列的数为 1 表示第j个车身进入了第i条车道，因此矩阵的每行和每列分别只会出现一个“1”。在初始情况下，这是一个随机生成的矩阵，再根据约束条件和优化目标进行 0 和 1 的重排。下面以第 1、2、3、4 个车身分别进入第 4、2、3、6 条车道的情况为例进行计算的说明：

$$Rand_{in}=\begin{pmatrix}0&0&0&0&\cdots\\0&1&0&0&\cdots\\0&0&1&0&\cdots\\1&0&0&0&\cdots\\0&0&0&0&\cdots\\0&0&0&1&\cdots\end{pmatrix}$$

将其转化为一个 318 个数的序列$Seq_{in}=\{4,2,3,6,\ldots\ldots\}$，表示矩阵每列中“1”所在行，即车身序列进入的车道序号。

通过上面矩阵对Q_{in}的左乘运算进行矩阵的行交换，得到每个车身的所在进车道的一个 6×318 的矩阵：

$$PBS_{in}=Rand_{in}\cdot Q_{in}=\begin{pmatrix}0&0&0&0&\cdots\\0&2&0&0&\cdots\\0&0&3&0&\cdots\\1&0&0&0&\cdots\\0&0&0&0&\cdots\\0&0&0&4&\cdots\end{pmatrix}$$

对于PBS_{in}矩阵，可以用$\alpha_{ij}^{(t)}$和$p_m^{(t)}$来表示每个位置的车身和每个车身在 t 时刻的位置坐标。 例如上面的矩阵中，选取某个时刻t，有$\alpha_{41}^{(t)}=1$，$\alpha_{63}^{(t)}=0$，$p_1^{(t)}=(4,1)$，$p_3^{(t)}=(3,3)$.

而在车身进入进车道之后，每个车道的车身队伍并不一定是连续的，可能会因为出入进车道的优先顺序发生缓冲，或导致两个车身之间有空停车位。此问题在仿真模拟程序中进行设置。

车身从进车道的停车位 1 装载到 PBS 送车横移机的序列也为一个 318×318 的 0-1 矩阵，第i行第j列的数为“1”表示第i个车身是第j个被装载到送车横移机，因此矩阵的每行和每列分别只会出现一个“1”。这也是一个随机生成的矩阵，再根据约束条件和优化目标进行矩阵列的重排，即将车身装载到送车横移机上的顺序进行重排。下面以在上述进车道序列下，第 1、2、3、4 个车身分别为第 2、4、3、1 个转载到送车横移机上的情况为例进行计算的说明：

$$Rand_{out}=\begin{pmatrix}0&0&0&1&\cdots&0\\1&0&0&0&\cdots&0\\0&0&1&0&\cdots&0\\0&1&0&0&\cdots&0\\\vdots&\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&0&0&\cdots&1\end{pmatrix}$$

通过上述矩阵的右乘对矩阵PBS_{in}的右乘运算进行矩阵的列交换，得到表示每个车身从所在进车道转移到送车横移机的顺序一个 6×318 的矩阵：

$$PBS_{out} = PBS_{in} \cdot Rand_{out} \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 2 & \cdots \\ 0 & 0 & 3 & 0 & \cdots \\ 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ 4 & 0 & 0 & 0 & \cdots \end{pmatrix}$$

将上述矩阵也转化为一个一维序列，在矩阵第j列的车身序号表示该车身是第j个进入PBS 总装区，生成一个有 318 个数的出车序列$Seq_{out} = \{4,1,3,2, ...\}$，再将这些序号与$\lambda_i, \mu_i$一一对应，实现两个 0-1 序列的重排，即出车序列对应的混动车型和燃油车型的序列，以及两驱车与四驱车的序列。从而对优化目标 1 和 2 计算得分。上述所有算法的程序见附录。

3.3 模型求解

首先我们对题目所给的附件 1 中的动力类型数据进行筛选，发现 318 个车身中的混动车身和燃油车身的比例为 2:1。而想要使得优化目标 1 的值为 100，混动车身和燃油车身的比例应为 1:2，因此至少会有 159 个混动车身前后两两之间非混动车身数不为 2，即至少扣 158 分。我们得到附件 1 数据针对优化目标 1 的最高得分为-58 分。

再对驱动类型进行筛选，得到两驱车身和四驱车身的数量分别为 289 和 29 个。不考虑其他约束条件和优化目标，若 29 个四驱车身的序号均不相邻，且间隔不同数量的两驱车身，优化目标 2 的得分最低，为 71 分；最高为 100 分。当因此优化目标 2 的分数均在 71-100 之间，加权之后的变动范围仅为 8.7 分。

在附件 1 中，若所有车都依次进入第 4 条车道，则容易求出最后的总得分为 13.1 分。

PBS优化调度问题属于 NP-hard 问题，遗传算法在求解 NP-hard 问题中具有很强的适用性，遗传算法（GA）相比其他算法具有鲁棒性强，搜索能力强[5]，但针对PBS优化调度问题，搜索车身任务过程中容易过早收敛，从而可能导致结果为局部最优解。

对此，我们提出了改进的算法：（1）提出了针对PBS优化调度问题的编码方式，解决了缓冲区位置与不同车身之间多对多的关系。（2）提出父代子代融合操作，以防丢失优质全局解及陷入局部最优解。（3）利用贪心策略，比较了不同的随机方式得到的初值的好坏，并分别分析了其算法的优劣，选用了较优的一种算法迭代得到初始种群[6]。（4）改进了针对该编码方式的适应性交叉和变异概率，有效的避免了遗传算法局部最优解的产生。（5）比较了几个不同的交叉及变异的结果，给出了更优的解。基于这些思想，构造了求解问题的算法。

对于求解PBS优化调度问题，根据问题一所要求的全部约束条件（如接车横移机在有任务时不能被打断），对每个车身在每个时刻的路径进行规划，确定每个车身的路径图。首先，构建一维数组，利用随即方法得到一系列初始数组，利用贪心策略分别构建多种初始值算法，经过适当次迭代后，选取其中结果较好的作为初始化种群。其次，通过交叉和变异操作，对初始化种群进行改造，分别比较了几种不同的交叉变异操作，并且利用适应性函数对每个结果进行评价。然后，检查结果优劣，即结果需要满足全部约束条件，并尽可能获得更好的适应性函数评价。最后，输出最优解结果，包括总时间及每一车身在每一时刻的

位置矩阵。我们改进后的优化算法如图 3-4 所示。

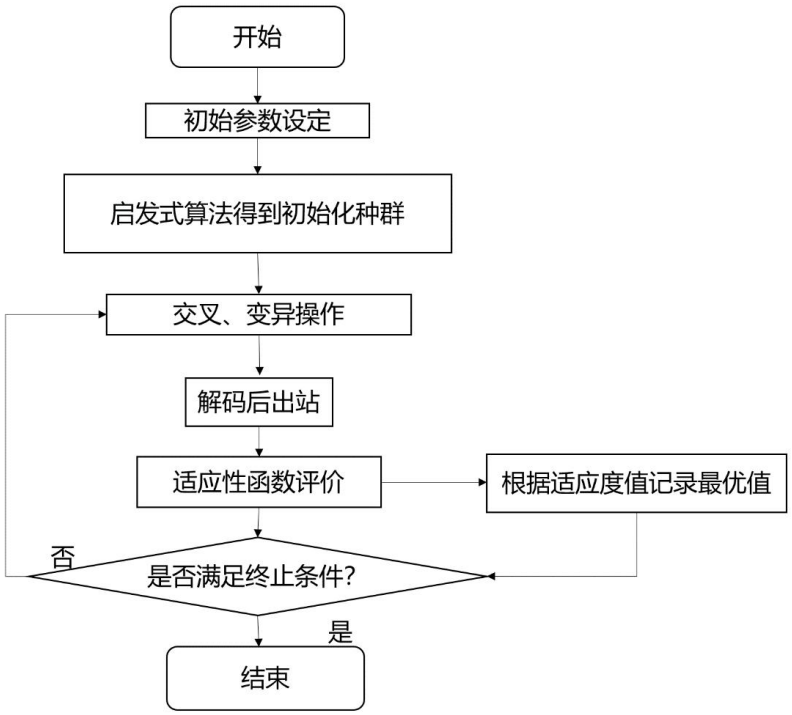

图 3-4　遗传算法流程图

① 启发式算法得到初始种群

首先，我们利用仿真程序 STEP 5 随机生成一个 1×318 的数组，并根据优化目标的权重，增加不同的约束条件来构建相应的启发式算法。

若假设所有车身都按照顺序依次从第 4 条车道进入。此时，可以明确计算出各优化目标的得分。记为r. 后续所有得分都与r进行比较，若得分小于r，则该算法和结果直接不予考虑。

调度策略 1: 对进车序列增加约束条件：每一辆车身进入车道是随机的。对出车序列增加约束条件：每一辆车返回的概率$P = 0.05$.利用随机数组迭代 10000 次。循环 3 次，并对结果取平均值，记为r_1.

调度策略 2: 对进车序列增加约束条件：每一辆车身进行车道是随机的。对出车序列增加约束条件：每一辆车返回的概率$P = 0.001$. 利用随机数组迭代 10000 次。循环 3 次，并对结果取平均值，记为r_2.

调度策略 3: 利用贪心策略，只对出车序列增加约束条件：若同时有多辆车身到达停车位 1，则优先输出一辆混动车型和两辆燃油车型；若余下的车型中无法构成一辆混动车型和两辆燃油车型，则优先输出混动车型，其次输出燃油车型。利用随机数组迭代 10000 次。循环 3 次，并对结果取平均值，记为r_3.

调度策略 4: 由于附件 2 中的比例有所变化，所以我们在优化算法时要考虑到不同的车型比例。遍历导入数据，记混动车型总数量为y,燃动车型总数量为z，比较$\frac{y}{z}$与$\frac{1}{2}$的大小.

若$\frac{y}{z} > \frac{1}{2}$，则对进车序列增加约束条件：若当前进车是混动车型，则随机进入第 1，2，3，4，5，6 条车道；若当前进车是燃油车型，则随机从第 3，4 条车道中选取一条车道进

入。利用随机数组迭代 10000 次。循环 3 次，并对结果取平均值，记为r_4.

若$\frac{y}{z} \leq \frac{1}{2}$，则对进车序列增加约束条件：若当前进车是混动车型，则随机进入第 3，4 条车道；若当前进车是燃油车型，则随机进入第 1，2，3，4，5，6 条车道。利用随机数组迭代 10000 次。循环 3 次，并对结果取平均值，记为r_4.

图 3-5 中的流程图展示了我们的启发式调度策略 4 和 PBS 仿真模拟之间的关系。

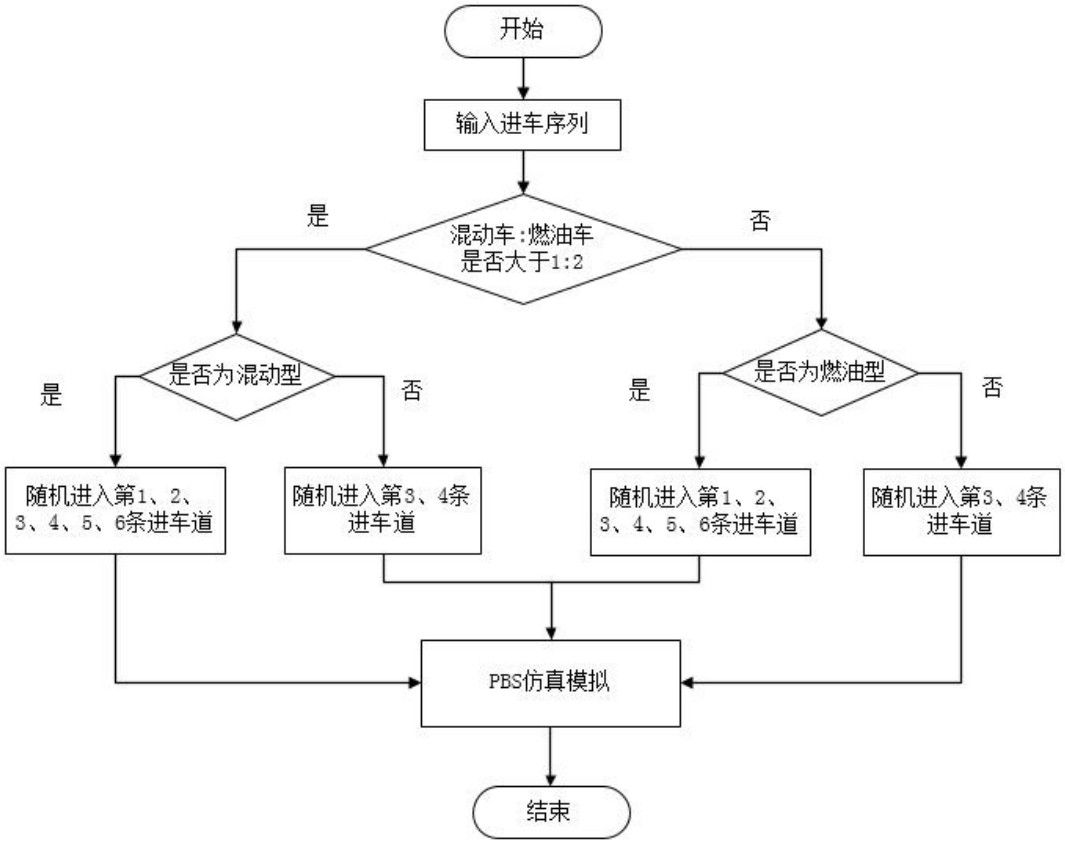

图 3-5 调度策略 4 流程图

② 启发式算法的结果比较

将上述得到的结果进行比较，结果如表 3-1 所示。

表 3-1 各算法附件 1 得分表

	总得分	优化目标 1 得分	优化目标 2 得分	优化目标 3 得分	优化目标 4 得分	执行附件 1 时间（秒）
调度策略 1	15.13	-87	82	86	81.37	1046
调度策略 2	15.90	-88	80	90	85	1055
调度策略 3	16.28	-85	74	100	80.75	1062
调度策略 4	17.68	-83	76	100	80.48	1082

由表 3-1 可知，调度策略 1 和调度策略 2 的约束条件和执行时间比较接近。其原因在

于这两个调度策略都是基于一定概率下的随机调度，无论是进车还是出车都服从等概率模型。等概率模型的随机性较强，因此这两者的优化结果相对较差。其中，调度策略 2 由于初始设定大大降低了车身返回的概率，因此目标 3 和目标 4 的得分有了一定地提升，从而总体分数略微超过了调度策略 1。但是显然，返回概率的进一步降低对结果的进一步优化贡献将会十分有限。由此，返回道使用次数降低有利于结果的优化，但其对总优化分数的贡献力度会随着使用次数的趋近于零而不断减弱。

对于调度策略 3 来说，其约束条件有所增多，但执行时间增加不明显，最后的总得分增加也不够明显，相对调度策略 2 没有明显优势。我们发现，贪心策略的介入在优先满足目标 1 的同时相对于调度策略 2 来说提升并不明显。究其原因，是调度策略 2 的随机性和整体程序运行的偶然性导致的。后续可以通过多次运行程序来进行检验。

对于调度策略 4 来说，其约束条件的数量比较适中，执行时间有所增加，但得到的优化总得分增加相对明显。该算法充分考虑了不同数据中混动车型和燃动车型的比例，在不同比例时给出了不同的车道分配方案。在附件 1 数据下，最终得分相对前三个算法分别大约提升了 17%，11%和 9%，但运行时间仅分别多出了 3%，2.6%和 1.9%。因此，调度策略 4 的效果和效率都显著好于前三种算法。

因此我们选取调度策略 4 进行初始值的选取和迭代，利用随机方法得到一个初始数组，利用调度策略 4 循环10^4次，得到问题一的初始种群。我们将初始种群数设置为 50.

③ 对初始种群进行交叉变异

在遗传算法中，种群的个体是由基因组成的染色体来表示的。一条染色体就对应了问题的一个解，对染色体的编码过程就是染色体和问题解的对应过程，而对染色体基因的优化过程就是对问题解的优化过程。其染色体上的基因序列即为每个车身进入的进车道序号。

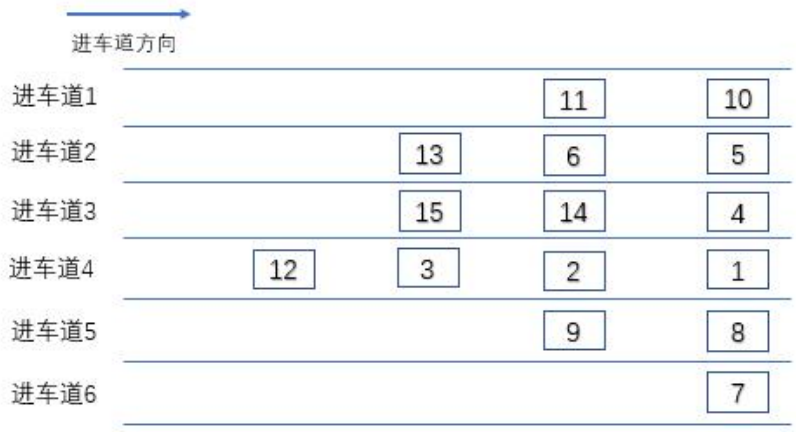

图 3-6 PBS 缓冲区车辆调度示例图

在图 3-6 中，选取了过程的第t时刻，此时，假设共有序列中前 15 个车身在缓冲区中，分别对应车道 1 至车道 6 中的某些车道。这个问题对应的染色体编码示意图如图 3-7 所示。

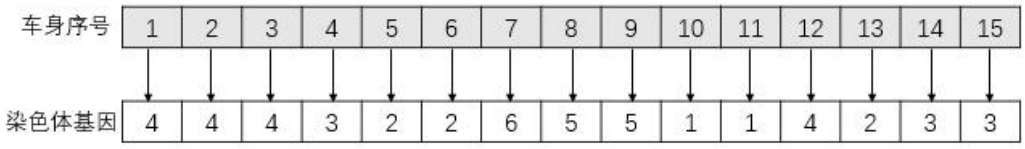

图 3-7 染色体编码示意图

为了方便记录和编程，可将上述染色体记为一个一维向量

$$M = \{4,4,4,3,2,2,6,5,5,1,1,4,2,3,3\}$$

有了一维向量以后，就可以利用交叉和变异的方法对其进行处理。本文主要用到单点交叉和单点变异的方法。

单点交叉是一种简单交叉方式，是指在编码串中只随机设置一个交叉点，然后在该点相互交换两个被配对的个体的部分基因[7]。例如，图 3-8 直观地描述了一个单点交叉的过程。在该示例中，3 和 6 被随机选中进行交叉操作。于是，交换 3 和 6 的位置，得到一组新的染色体。这种操作的好处是：若邻接的基因之间的关系比较好的话，就能在形成新的个体的同时以最小的可能性破坏个体形状。

问题 1 求解过程中的交叉操作直接采用了改进型的交叉遗传。

首先，设置初始种群数为 50，迭代次数为 1000，变异概率为 0.1。其次，利用混沌序列确定交叉点的位置。取$(0,1)$区间上的随机数作为一个初始值，然后利用$x(n+1) = 4x(n)[1-x(n)]$迭代一次产生一个$(0,1)$区间上的混沌值。将其保存并作为产生下一个混沌值的初始迭代值，最后把这个值乘以 316，再加上 2，并取整即可。这种做法的好处是，对初始解的改动非常小，有效避免了遗传算法在组合优化运用中产生的寻优抖振问题，并且可以一定程度上提高收敛精度。

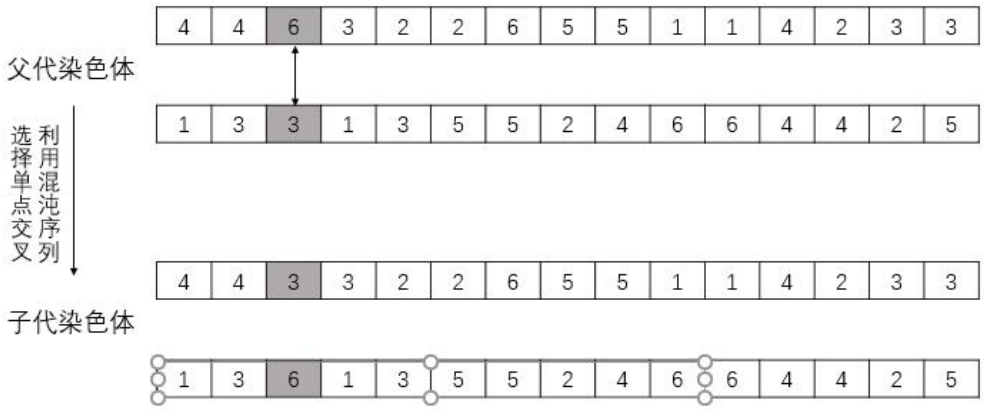

图 3-8 单点交叉示意图

单点变异是以变异概率P随机指定染色体的某一基因进行变异运算。也就是将染色体编码中的某些基因值用该基因的等位基因来替换，进而形成一个新的染色体[8]。如图 3-9 是一个单点变异的示例。在图 3-9 的示例中，4 被选中，给 4 随机赋一个值，比如 7，于是形成一个新的个体。该操作的目的主要是提高算法的局部搜索能力，并维持种群的多样性[9]。

接着，为了跳出局部最优解，寻找全局最优，我们进一步采用变异操作。变异算子的设计方法如下：首先根据设定的变异概率 0.1，随机地选取两个 2 到 317 之间的整数，对

这两个数对应位置的基因进行变异。具体操作为利用混沌序列把这两个位置的基因换成两个新的基因，从而得到新的染色体。得到新的染色体后形成子代染色体种群，将子代染色体种群和父代染色体种群融合到一起。这种做法的好处是，能一定程度上缓解单一的单点交叉导致的算法的早熟问题。随着迭代次数的增多，种群中就留下了适应度高的个体，它们就集中在最优解附近[10]。

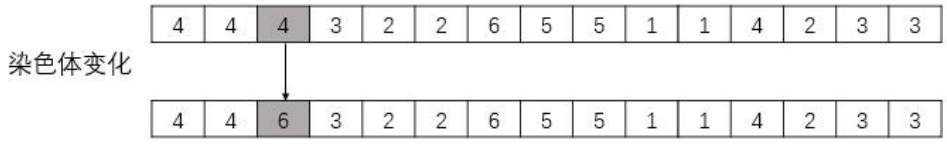

图 3-9 单点变异示意图

④ 改进后的遗传算法结果

我们将上述改进过的遗传算法的计算结果呈现于表 3-2，具体每个车身每个时刻的位置信息见 result 11.xlsx.

表 3-2 附件 1 优化结果

	总得分	优化目标 1	优化目标 2	优化目标 3	优化目标 4
单个交叉	22.53	-79	80	100	**101.26**
单个交叉+单个变异	26.91	-70	82	100	**103.12**

由表 3-2 可以看出，改进后的遗传算法运用于附件 1 得到的结果相比于所有车身全走 4 车道情形和调度策略 4 的初始解来说都有明显的提升。注意到，运用该算法后，优化目标 4 的分数均超过了 100 分，即此时所用的时间比题中所给的理论最短时间还要短。经检验，这种情况是存在的（举一个简单的例子：三个车身均走 4 车道，花费 99 秒；但若其依次进入第 4，第 3，第 4 车道，则只需花费 90 秒）。这说明优化后的结果生产效率较高，利于工业实际生产。

在仅运用单个交叉的情形中，其总得分要明显低于将单个交叉和单个变异结合起来用的情形，并且前者的收敛速度要高于后者。这是因为仅运用单点交叉会造成算法的早熟，而后者在加入变异以后，种群的多样性得到了一定的维持，因此收敛速度会降低，并且结果更好。

3.4 模型适应性分析

我们将附件 2 中的数据放入上述目标规划模型算法中进行测试，从而检验算法的适应性，得到结果见表 3-3，具体每个车身每个时刻的位置信息见 result 12.xlsx.

表 3-3 附件 2 运算结果

	总得分	优化目标 1	优化目标 2	优化目标 3	优化目标 4
单个交叉	48.24	-17	83	100	**101.41**
单个交叉＋单个变异	53.04	-9	82	100	**104.41**

由表 3-3，附件 2 的运算结果相较于其初始值也有一定的提高，说明该模型和优化算法的鲁棒性较高。同样地，在加入变异操作后，种群的多样性被更好地维持了，得到的结果也要好得多。

四、问题 2 优化模型的建立与求解

4.1 问题分析

问题 2 删去了 PBS 约束条件中对接送车横移机任务优先级的设置，从而增大了模型求解的可行域。显然问题 1 的解包含于问题 2，因此我们从问题 1 的模型求解结果出发，进一步对调度模型进行优化，对调度策略以及算法进行改进。

由于不要求在进车道 1 停车位停留时间最长的车身被优先处理，它们可以在此处任意停留，接送横移机先处理其他符合我们算法中约束条件的车身。根据优化目标的权重设置，连续的两个混动车型是否间隔两台非混动车型对得分的影响较大，因此，我们可以进一步利用贪心策略，使出车序列优先满足上述条件，不满足则返回。且返回道 10 停车位的车身不一定被优先处理，所以可以一定程度上缓解车身拥堵等待的问题。

问题 2 的解决思路部分与问题 1 重复，如图 4-1 所示，与问题 1 解决过程重复的步骤下面我们将不再详述。

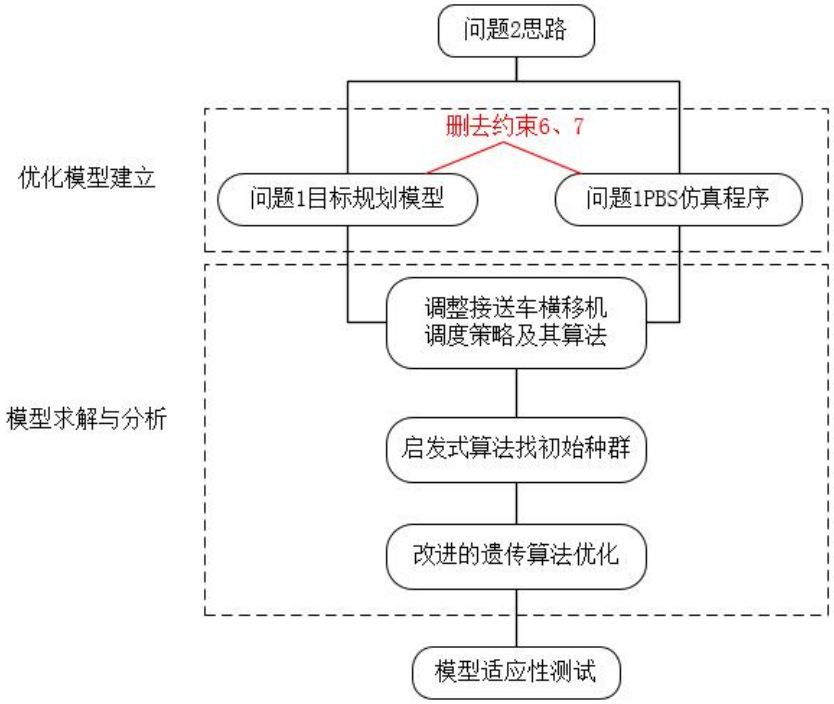

图 4-1 问题 2 求解思路

4.2 问题求解

下述过程都在问题 1 的模型和 PBS 仿真程序上进行操作。

① 改进调度策略的启发式算法

不考虑约束 7，即不给进车道 10 停车位车身停留时长设置优先级，我们尝试进行如下

调度策略的算法：

改进策略 1 为了尽可能地避免进车道拥堵的情况，我们先将所有停车位都已有车身的进车道的 10 停车位的车身优先送出。若有多个满足所有停车位已满的车道，则按照车身序号的先后处理。若车身为混动型，则直接运送进 PBS 总装区。若车身为燃油型，且不满足 2 个燃油型前有 1 个混动型的的排序，直接送入返回道。从返回道出来的车身仍然被随机送入第 3 或第 4 条车道。

表 4-1 改进策略 1 运行结果

总得分	优化目标 1	优化目标 2	优化目标 3	优化目标 4
22.75	-73	82	91	91.51

但是我们意识到若燃油型车身在进车序列中占比较大，会陷入一直循环的状态，模型非常不稳定，因此我们应当继续加上返回道使用约束条件。

由于出车序列的得分占比高，而返回道使用次数以及时间的得分占比较小，所以我们进行权衡之后，认为应当让不满足序列要求的燃油型车身以较高概率进入返回道，且经过尝试我们发现 0.95 左右的概率的值较高。因此进一步又设置了不满足燃油和混动间隔排序的燃油型车身有 0.95 的概率被送入返回道。运行结果如表 4-2 所示。

表 4-2 改进策略 2（置$P_1 = 0.95$）运行结果

总得分	优化目标 1	优化目标 2	优化目标 3	优化目标 4
21.76	-74	80	91	91.60

由上表可知，结果与不考虑返回概率，即令$P_1 = 1$ 的数据结果相差甚微，且模型更为稳定，因此我们后续仍然依次概率进行策略的进一步调整。

在不考虑约束 7 的基础上再删去约束 6，即接车横移机不需要优先处理返回道 1 停车位上的车身。因此，我们再在上述改进策略的基础上，对于接车横移机增加调度策略：

改进策略 2 由于返回道的车身均为燃油型，且燃油型车身仅放入第 3 和第 4 进车道，若设置优先级，会发生堵车等待的情况。但是由于我们需要将返回道的燃油车型放入进车道与剩下的混动型进行间隔排序，避免混动型已经全部运送出 PBS，导致对目标优化作用比较大的燃油车型冗余，尤其是燃油车型数量较少的进车序列，更需要关注这个问题。所以我们仍然要尽量让返回道 10 停车位的车身尽早被接车横移机处理。

因此我们给返回道 10 停车位的优先处理设置一个概率 0.5，即若返回道 10 停车位有车，它有 0.5 的概率被接车横移机在下一个任务中处理。

表 4-3 改进策略 2 运行结果

	总得分	优化目标 1	优化目标 2	优化目标 3	优化目标 4
$P_1 = 0.95, P_2 = 0.5$	23.50	-70	80	92	90.97

② 改进的遗传算法

通过问题 1 对遗传算法中交叉和变异的尝试，我们发现采用利用混沌序列确定单个基因，并进行交叉，同时也要进行单点变异，得到的优化结果最佳。因此对于问题 2 的算法得到的染色体序列，我们仅运行单点交叉和单点变异同时进行的算法，得到附件 1 中数据

优化的得分如表 4-4 所示。

表 4-4 问题 2 附件 1 数据运行结果

总得分	优化目标 1	优化目标 2	优化目标 3	优化目标 4
28.72	-63	82	98	97.21

我们将问题二的结果与问题一和初始值进行了比较，结果如图 4-2 所示。

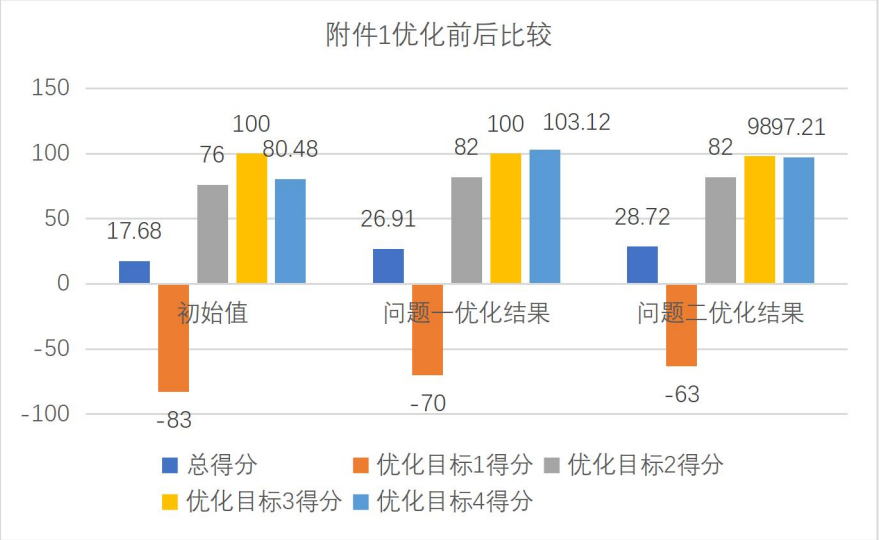

图 4-2 附件 1 优化前后比较

由图 4-2 可以看出，附件 1 优化结果较为明显。问题二相比初始值总分大幅增长，其原因主要是优化目标 1 和优化目标 4 分数的大幅提高。这是由于问题二的主要优化方向就是这两者。而相比于问题一，问题二的优化目标 1 分数有一定增长，但优化目标 3 和 4 的分数均有所降低。这样设置优化方向的原因是优化目标 1 的权重占比最高。

4.3 模型适应性分析

我们再把附件 2 的数据放入算法中运行，以测试模型和策略的适应性，其优化结果的得分如表 4-5 所示。

表 4-5 问题 2 附件 2 数据运行结果

总得分	优化目标 1	优化目标 2	优化目标 3	优化目标 4
57.30	15	85	84	90.04

问题 2 中附件 2 优化结果与问题 1 和初始值比较如图所示。

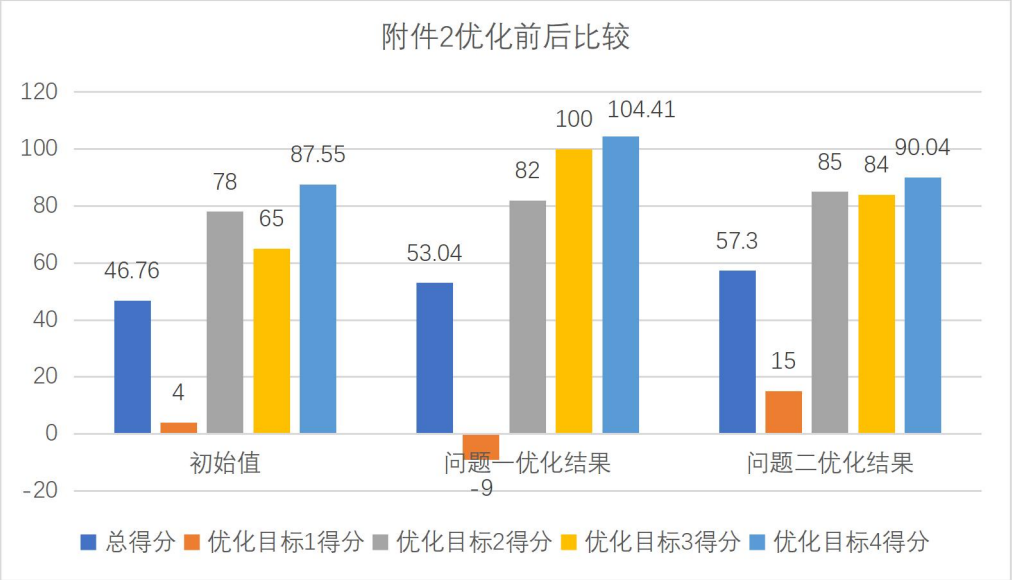

图 4-3 附件 2 优化前后比较

由图 4-3 所知，附件 2 的优化结果中，同样问题二优于问题一，且均大幅优于初始值。其中，问题二的优化目标 1 得分大幅超过问题一，但优化目标 4 的得分有一定下降。这是鉴于优化目标 1 的权重占比最高来考虑的。此外，问题二进一步根据权重优化目标 2，降低目标 3，与问题一结果相比，有较为明显的提高。说明我们减少约束条件后，改进调度策略的有效性较好。

五、模型的评价与推广

5.1 模型的优点

我们基于一定的实际情况与直观经验，形成启发式算法，将 NP-hard 问题转化为在有限时间内得到一个较优的可行解的问题，并利用改进的遗传算法得到了相对比较好的解。具体来说，

（1）调度策略多样，初始化种群选取优良

我们基于现实和实践经验，分别受等概率模型、贪心策略和启发式策略启发，创造性地提出了四种不同的调度策略。针对这四种调度策略，我们分别添加一定的约束条件，用随机数组和大量次数的迭代分别展示了其算法优劣，包括结果优劣及算法复杂度，并据此选出了后续算法中的主要策略。

（2）灵活使用混沌序列，科学选取交叉变异位置

我们基于混沌序列的遍历性、随机性和非周期性，利用其产生科学的随机数，确定父代染色体中的交叉变异位置，有效避免了遗传算法在优化问题中产生的寻优抖振，提高了算法的收敛精度。

（3）PBS 仿真程序精准模拟调度状态

我们基于约束条件和算法，利用 PBS 仿真程序精准模拟了 PBS 线性缓冲区的优化过程。在模拟过程中，我们发现优化目标 4 的得分可以超过 100，即存在比全走 4 车道花费时间更短的情形。因此，我们提出的仿真程序描述该问题是比较精准科学的。

5.2 模型的缺点

启发式算法得到的结果并不能确定是最优解，有可能结果会比较差。我们在启发式算法中采用的贪心策略也并不能完全考虑进出车道这两个过程同时作用的影响。而改进后的遗传算法中单点交叉的操作强度较弱，容易使算法早熟，后续可以再采用强度较大的多点变异，在一定程度上解决这个问题。

5.3 模型推广

PBS 调度区车身应当同时考虑进和出两个过程，这两个过程相互影响，所以我们的启发式算法不能简单地进行贪心搜索，也就是只考虑一边的情况进行优化。后续应当要重点研究将两个过程结合考虑得到的调度策略。

本文提出的调度算法是一种启发式的算法，但是由于实际生产扰动因素的影响，它的鲁棒性受到了很大的影响，它必须根据调度方案来安排特定的车辆，而由于干扰因素的影响，不符合调度方案，所以系统不能识别出这些车辆，从而会使车辆在 PBS 调序区等待，影响缓冲区的整体运行。在此基础上，加入一种基于实时运算的启发式算法，可以针对各种干扰类型，作出相应的决策。比如，如果有品质不达标的定单要下线，就会被直接从生产计划中删除；对于紧急插入的定单，将会将此定单添加到原始的制造计划中，并提供其所要插入的地点。采用这种启发式方法，可以对干扰因子进行快速反应，提高整个算法的鲁棒性。

参考文献

[1]陈正茂. 基于排序缓冲区的多车间关联排序研究[D]. 华中科技大学, 2008.
[2]胡婷婷. PBS 缓冲区车身路由调度策略的研究与应用[D].合肥工业大学,2017.
[3]陈广阳. 汽车生产线缓冲区设计及排序问题研究[D].华中科技大学,2007.
[4]沈振宇. 面向汽车混流生产线的排产与调度问题研究[D].重庆大学,2019.
[5]杜利珍,叶涛,宣自风,王宇豪.改进遗传算法求解织造车间并行批调度问题[J].武汉纺织大学学报,2022,35(04):8-12.
[6]易春磊,柴良明,王世友,贺柱.基于改进型混沌遗传算法的农村配电网无功补偿研究[J].电气自动化,2022,44(04):40-43.
[7]殷新凯,茅健,周玉凤,陈晓平.基于混合混沌序列与遗传算法的排爆机器人路径规划[J].计算机时代,2019(07):5-8.
[8]蒋珉,王廷平,严洪森.含有限缓冲区的多生产线协调生产计划的研究.东南大学学报(自然科学版),2004，34(15):622-637.
[9]Golberg J,et.al. Genetic Algorithm for the Traveling Salesman Problem. In Proc. of 1 st Int. Conf. on Genetic Algorithms and Their Applications, Lawrence Erlbaum Associates, 1985,160-180.
[10]Zhipeng Tian,Xinyu Shao,Haiping Zhu,Hui Yin,Fei He. Small-World Optimization Algorithm and Its Application in a Sequencing Problem of Painted Body Storage in a Car Company[J]. Mathematical Problems in Engineering,2015,2015(Pt.5).

附录 MATLAB 代码

核心代码及说明:

问题 1

初始解

```
clc, clear;


data = xlsread('附件 1.xlsx');
in_t = [18;12;6;0;12;18];
rng('shuffle');
lane_2 = [3,4];


for epoch=1:10000
    c_in = zeros(1, length(data));              %入车序列


    left = length(data);
    t = 0;
    cnt = 1;
    pos = zeros(8, 10);                 %记录 70 车位以及横移机上车辆的序号
    pos_t = zeros(7, 10);               %记录车辆在该车位上停留的时间


    wait_in_t = 0;                      %横移机回到中间的时刻
    reach_in_t = 0;                     %车辆从横移机到车位的时刻
    up_out_t = 10000;                   %车辆从车位到横移机的时刻
    wait_out_t = 10000;                 %横移机回到中间时刻
    lane = 0;
    res = [];                           %记录每个时刻每辆车所处的位置，由多个 pos 拼接而成
    c_out = [];                         %出车序列
    in_1 = 1;
    in_2 = 2;                           %混动-燃油-燃油计数


    while(left~=0)


        %进车
```

```
if cnt~= length(data)+1
    if pos(8,1) == 0
        if data(cnt,2) == 1
            lane = randperm(6,1);  %混动随机进入一道
        end
        if data(cnt,2) == 2
            lane = lane_2(randperm(length(lane_2),1));   %燃油进入 3  4 中一道
        end
        c_in(cnt) = lane;
        if in_t(lane) > 0
            pos(8,1) = cnt;
            reach_in_t = t + in_t(lane) / 2;
            wait_in_t = t + in_t(lane);
        else                        %第 4 道
            if pos(lane, 1) == 0    %10 号位没车
                pos(lane, 1) = cnt;
                cnt = cnt + 1;
                continue
            end
        end
    end
end


t = t + 3;                          %时间取最大公约数 3


%车在车道中移动
for i=1:6
    for j=10:-1:1
        if j == 10
            if pos(i, j) ~= 0
                pos_t(i, j) = pos_t(i, j) + 3;
            end
        else
            if pos(i, j) ~= 0 && ~isempty(find(pos(i,j+1:10) == 0)) %前方有
空位

                pos_t(i, j) = pos_t(i, j) + 3;
            end
            if pos_t(i, j) >= 9                  %移动
                pos(i, j + 1) = pos(i, j);
                pos(i, j) = 0;
                pos_t(i, j) = 0;
```

```
            end
        end
    end
end


if t == reach_in_t
    if pos(lane, 1) == 0
        pos(lane, 1) = pos(8,1);
    else
        reach_in_t = reach_in_t + 3;
        wait_in_t = wait_in_t + 3;
    end
end


if t == wait_in_t
    pos(8,1) = 0;
    cnt = cnt + 1;
end


if t == up_out_t
    pos(row,col) = 0;
    pos_t(row,col) = 0;
end


if t == wait_out_t
    c_out = [c_out, pos(8,2)];
    pos(8,2) = 0;
    left = left - 1;
end


%出车
if pos(8,2) == 0
    car1 = [];          %将待出车辆分为混动和燃油两类
    car2 = [];          %记录待出车辆所在车道
    car1_t = [];
    car2_t = [];        %记录待出状态的时间，判断优先级
```

```
car_1 = [];
car_2 = [];
col = 10;
for i=1:6
    if pos(i,10)~=0
        if data(pos(i,10),2) == 1
            car1 = [car1 i];
            car1_t = [car1_t pos_t(i,10)];
        else
            car2 = [car2 i];
            car2_t = [car2_t pos_t(i,10)];
        end
    end
end


%优先满足混动-燃油-燃油的序列


if isempty(car1) && isempty(car2)
    res = [res;pos];
    continue
end


if ~isempty(car1) && isempty(car2)
    [car_1, indcar1]=sort(car1_t,'descend');
    index1 = find(car1_t == car_1(1));
    row = car1(index1(randperm(length(index1),1)));
    in_1 = 0;
    if in_2 ~= 2
        in_2 =2;
    end
end


if isempty(car1) && ~isempty(car2)
    [car_2, indcar2]=sort(car2_t,'descend');
    index2 = find(car2_t == car_2(1));
    row = car2(index2(randperm(length(index2),1)));
    if in_1 == 0
        in_2 = in_2 - 1;
        if in_2 == 0
```

```
            in_1 = 1;
            in_2 = 2;
        end
    end
end


if ~isempty(car1) && ~isempty(car2)
    [car_1, indcar1]=sort(car1_t,'descend');
    [car_2, indcar2]=sort(car2_t,'descend');
    if car_1(1) > car_2(1)
        index1 = find(car1_t == car_1(1));
        row = car1(index1(randperm(length(index1),1)));
        in_1 = 0;
        if in_2 ~= 2
            in_2 =2;
        end
    elseif car_1(1) < car_2(1)
        index2 = find(car2_t == car_2(1));
        row = car2(index2(randperm(length(index2),1)));
        if in_1 == 0
            in_2 = in_2 - 1;
            if in_2 == 0
                in_1 = 1;
                in_2 = 2;
            end
        end
    else
        if in_1 == 1
            index1 = find(car1_t == car_1(1));
            row = car1(index1(randperm(length(index1),1)));
            in_1 = 0;
        else
            index2 = find(car2_t == car_2(1));
            row = car2(index2(randperm(length(index2),1)));
            in_2 = in_2 - 1;
            if in_2 == 0
                in_1 = 1;
                in_2 = 2;
            end
        end
    end
end
```

```
if row == 4
    c_out = [c_out,pos(row,col)];
    pos(row,col) = 0;
    pos_t(row,col) = 0;
    left = left - 1;
    if find(car1==4)
        car1_t(find(car1==4))=[];
        car1(find(car1==4))=[];
    else
        car2_t(find(car2==4))=[];
        car2(find(car2==4))=[];
    end


    if isempty(car1) && isempty(car2)
        res = [res;pos];
        continue
    end


    if ~isempty(car1) && isempty(car2)
        [car_1, indcar1]=sort(car1_t,'descend');
        index1 = find(car1_t == car_1(1));
        row = car1(index1(randperm(length(index1),1)));
        in_1 = 0;
        if in_2 ~= 2
            in_2 =2;
        end
    end


    if isempty(car1) && ~isempty(car2)
        [car_2, indcar2]=sort(car2_t,'descend');
        index2 = find(car2_t == car_2(1));
        row = car2(index2(randperm(length(index2),1)));
        if in_1 == 0
            in_2 = in_2 - 1;
            if in_2 == 0
                in_1 = 1;
                in_2 = 2;
            end
```

```
        end
    end


    if ~isempty(car1) && ~isempty(car2)
        [car_1, indcar1]=sort(car1_t,'descend');
        [car_2, indcar2]=sort(car2_t,'descend');
        if car_1(1) > car_2(1)
            index1 = find(car1_t == car_1(1));
            row = car1(index1(randperm(length(index1),1)));
            in_1 = 0;
            if in_2 ~= 2
                in_2 = 2;
            end
        elseif car_1(1) < car_2(1)
            index2 = find(car2_t == car_2(1));
            row = car2(index2(randperm(length(index2),1)));
            if in_1 == 0
                in_2 = in_2 - 1;
                if in_2 == 0
                    in_1 = 1;
                    in_2 = 2;
                end
            end
        else
            if in_1 == 1
                index1 = find(car1_t == car_1(1));
                row = car1(index1(randperm(length(index1),1)));
                in_1 = 0;
            else
                index2 = find(car2_t == car_2(1));
                row = car2(index2(randperm(length(index2),1)));
                in_2 = in_2 - 1;
                if in_2 == 0
                    in_1 = 1;
                    in_2 = 2;
                end
            end
        end
    end
end
if row ~= 4
    pos(8,2) = pos(row, col);
```

```
            end


            up_out_t = t + in_t(row) / 2;
            wait_out_t = t + in_t(row);
        end
        res = [res;pos];
    end


    %计算得分
    x1 = 100;
    x2 = 100;
    data_out = data(c_out,:);
    ind1 = find(data_out(:,2)==1);


    for i=2:length(ind1)
        if ind1(i) - ind1(i-1) ~= 3
            x1 = x1 - 1;
        end
    end


    ind2 = [];
    for i=1:length(data_out) - 1
        if data_out(1, 3) == 1
            if data_out(i, 3) == 2 && data_out(i+1,3) == 1
                ind2 = [ind2, i];
            end
        else
            if data_out(i, 3) == 1 && data_out(i+1, 3) == 2
                ind2 = [ind2, i];
            end
        end
    end


    ind2 = [0 ind2 length(data_out)];
    for i=1:length(ind2)-1
        if length(find(data_out(ind2(i)+1:ind2(i+1),3)==1)) ~=
length(find(data_out(ind2(i)+1:ind2(i+1),3)==2))
            x2 = x2 - 1;
```

```
        end
    end


    x3 = 100;
    x4 = 100 - 0.01*(t - 9 * 318 - 72);
    x = 0.4*x1+0.3*x2+0.2*x3+0.1*x4;


end
```

问题二

初始解

```
clc, clear;


data = xlsread('附件 1.xlsx');
in_t = [18;12;6;0;12;18];
out_t =  [9 12 3;6 9 3;3 6 3;0 3 3;6 3 3;9 6 3];      %车进返回道所需时间，一共三个阶段
rng('shuffle');
lane_2 = [3 4];
maxx = 0;


for epoch=1:10000
    c_in = zeros(1, length(data));


    left = length(data);
    t = 0;
    cnt = 1;
    pos = zeros(8, 10);
    pos_t = zeros(7, 10);


    up_in_t = 0;                                    %车从返回道到横移机的时刻
    wait_in_t = 0;                                  %横移机回中位的时刻
    reach_in_t = 0;                                 %车从横移机到车道的时刻
    up_out_t = 10000;                               %车从车道到横移机的时刻
```

```
wait_out_t = 10000;                                  %横移机回中位时刻
reach_out_t = 10000;                                 %车从横移机到返回道的时刻
lane = 0;
res = [];
c_out = [];
re_cnt = 0;
re_0 = 0;
re_1 = 0;                                            %使用接车和送车横移机送入返回道
in_1 = 1;
in_2 = 2;
rate = 0.95;                                         %返回概率，针对燃油车


while(left~=0)


    %进车
    r=rand;
    k = rand;
    %概率取返回道中的车进入
    if k < 0.5
        if pos(7, 1) ~= 0
            if pos(8,1) == 0
                re_0 = 1;
                lane = lane_2(randperm(length(lane_2),1));   %走 3 4 道
                pos(8,1) = pos(7, 1);
                up_in_t = t + out_t(lane, 3);
                reach_in_t = up_in_t + out_t(lane, 2);
                wait_in_t = reach_in_t + out_t(lane, 1);
            end
        end
    end


    if cnt~= length(data) + 1
        if pos(8,1) == 0
            lane = randperm(6,1);                    %所有车随机走六道中的一道
            c_in(cnt) = lane;
            if in_t(lane) > 0
                pos(8,1) = cnt;
                reach_in_t = t + in_t(lane) / 2;
                wait_in_t = t + in_t(lane);
            else
```

```
            if pos(lane, 1) == 0
                pos(lane, 1) = cnt;
                cnt = cnt + 1;
                continue
            end
        end
    end
end


t = t + 3;


%移动
for i=1:6
    for j=10:-1:1
        if j == 10
            if pos(i, j) ~= 0
                pos_t(i, j) = pos_t(i, j) + 3;
            end
        else
            if pos(i, j) ~= 0 && ~isempty(find(pos(i,j+1:10) == 0))
                pos_t(i, j) = pos_t(i, j) + 3;
            end
            if pos_t(i, j) >= 9
                pos(i, j + 1) = pos(i, j);
                pos(i, j) = 0;
                pos_t(i, j) = 0;
            end
        end
    end
end


%返回道移动
i = 7;
for j=2:10
    if pos(i, j)~=0 && ~isempty(find(pos(i,1:j) == 0))
        pos_t(i, j) = pos_t(i, j) + 3;
    end
    if pos_t(i, j) >=9
        pos(i, j - 1) = pos(i, j);
        pos(i, j) = 0;
```

```
        pos_t(i, j) = 0;
    end
end



if t == up_in_t && re_0 == 1
    pos(7, 1) = 0;
    pos_t(7, 1) = 0;
end


if t == reach_in_t
    if pos(lane, 1) == 0
        pos(lane, 1) = pos(8,1);
    else
        reach_in_t = reach_in_t + 3;
        wait_in_t = wait_in_t + 3;
    end
end
if t == wait_in_t
    pos(8,1) = 0;
    if re_0 == 0
        cnt = cnt + 1;
    else
        re_0 = 0;
    end
end


if t == up_out_t
    pos(row,col) = 0;
    pos_t(row,col) = 0;
end


if t == reach_out_t && re_1 == 1
    re_cnt = re_cnt + 1;
    if pos(7, col) == 0
        pos(7,col) = pos(8,2);
    else
        reach_out_t = reach_out_t + 3;
```

```
        wait_out_t = wait_out_t + 3;
    end
end


if t == wait_out_t
    if re_1 == 0
        c_out = [c_out, pos(8,2)];
        pos(8,2) = 0;
        left = left - 1;
    else
        pos(8,2) = 0;
        re_1 = 0;
    end


end


%出车
if pos(8,2) == 0
    car1 = [];              %将车分为燃油混动两类
    car2 = [];              %记录待出车辆所在车道
    car1_cnt = [];
    car2_cnt = [];          %记录该车道总共有多少辆车,较拥挤的道优先出车
    col = 10;
    for i = 1:6
        if pos(i,col)~=0
            if data(pos(i,col),2) == 1
                car1 = [car1 i];
                car1_cnt = [car1_cnt length(find(pos(i,:)~=0))];
            else
                car2 = [car2 i];
                car2_cnt = [car2_cnt length(find(pos(i,:)~=0))];
            end
        end
    end


    %优先满足混动-燃油-燃油序列


    if isempty(car1) && isempty(car2)
```

```
    res = [res;pos];
    continue
end


if ~isempty(car1) && isempty(car2)
    [car1n,car1ind] = sort(car1_cnt,'descend');
    row = car1(car1ind(1));
    in_1 = 0;
    if in_2 ~= 2
        in_2 = 2;
    end
end


if isempty(car1) && ~isempty(car2)
    [car2n,car2ind] = sort(car2_cnt,'descend');
    row = car2(car2ind(1));
    if in_1  == 1
        if r < rate          %不满足混动-燃油-燃油条件，燃油车概率返回
            re_1 = 1;
            if row == 4
                pos(8,2) = pos(row,col);
                pos(row,col) = 0;
                pos_t(row,col) = 0;
            end
        else
            in_2 = 2;
        end
    else
        in_2 = in_2 - 1;
        if in_2 == 0
            in_1 = 1;
            in_2 = 2;
        end
    end
end


if ~isempty(car1) && ~isempty(car2)
    if in_1 == 1
        [car1n,car1ind] = sort(car1_cnt,'descend');
        row = car1(car1ind(1));
```

```
        in_1 = 0;
    else
        [car2n,car2ind] = sort(car2_cnt,'descend');
        row = car2(car2ind(1));
        in_2 = in_2 - 1;
        if in_2 == 0
            in_1 = 1;
            in_2 = 2;
        end
    end
end


if row == 4 && re_1 == 0
    c_out = [c_out,pos(row,col)];
    pos(row,col) = 0;
    pos_t(row,col) = 0;
    left = left - 1;
    if find(car1==4)
        car1_cnt(find(car1==4))=[];
        car1(find(car1==4))=[];
    else
        car2_cnt(find(car2==4))=[];
        car2(find(car2==4))=[];
    end


    if isempty(car1) && isempty(car2)
        res = [res;pos];
        continue
    end


    if ~isempty(car1) && isempty(car2)
        [car1n,car1ind] = sort(car1_cnt,'descend');
        row = car1(car1ind(1));
        in_1 = 0;
        if in_2 ~= 2
            in_2 = 2;
        end
    end
```

```
    if isempty(car1) && ~isempty(car2)
        [car2n,car2ind] = sort(car2_cnt,'descend');
        row = car2(car2ind(1));
        if in_1  == 1
            if r < rate   %不满足混动-燃油-燃油条件，燃油车概率返回
                re_1 = 1;
                if row == 4
                    pos(8,2) = pos(row,col);
                    pos(row,col) = 0;
                    pos_t(row,col) = 0;
                end
            else
                in_2 = 2;
            end
        else
            in_2 = in_2 - 1;
            if in_2 == 0
                in_1 = 1;
                in_2 = 2;
            end
        end
    end


    if ~isempty(car1) && ~isempty(car2)
        if in_1 == 1
            [car1n,car1ind] = sort(car1_cnt,'descend');
            row = car1(car1ind(1));
            in_1 = 0;
        else
            [car2n,car2ind] = sort(car2_cnt,'descend');
            row = car2(car2ind(1));
            in_2 = in_2 - 1;
            if in_2 == 0
                in_1 = 1;
                in_2 = 2;
            end
        end
    end
end


if row~=4
```

```
            pos(8,2) = pos(row, col);
        end
        if re_1 == 0
            up_out_t = t + in_t(row) / 2;
            wait_out_t = t + in_t(row);
        else
            up_out_t = t + out_t(row, 1);
            reach_out_t = up_out_t + out_t(row, 2);
            wait_out_t = reach_out_t + out_t(row, 3);
        end


    end
    res = [res;pos];
end


x1 = 100;
x2 = 100;
data_out = data(c_out,:);
ind1 = find(data_out(:,2)==1);


for i=2:length(ind1)
    if ind1(i) - ind1(i-1) ~= 3
        x1 = x1 - 1;
    end
end


ind2 = [];
for i=1:length(data_out) - 1
    if data_out(1, 3) == 1
        if data_out(i, 3) == 2 && data_out(i+1,3) == 1
            ind2 = [ind2, i];
        end
    else
        if data_out(i, 3) == 1 && data_out(i+1, 3) == 2
            ind2 = [ind2, i];
        end
    end
end
```

```
    ind2 = [0 ind2 length(data_out)];
    for i=1:length(ind2)-1
        if length(find(data_out(ind2(i)+1:ind2(i+1),3)==1)) ~=
length(find(data_out(ind2(i)+1:ind2(i+1),3)==2))
            x2 = x2 - 1;
        end
    end


    x3 = 100 - re_cnt;
    x4 = 100 - 0.01*(t - 9 * 318 - 72);
    x = 0.4*x1+0.3*x2+0.2*x3+0.1*x4;

end
```

遗传

```
clc,clear;


data = xlsread('附件 1.xlsx');
inin = load('yichuan11.txt');                          %初始解入车序列，  50 * 318
inin = inin';
in = [];
in(1,:) = inin(1:length(data));
for i = 2:50
    in(i,:) = inin(length(data)*(i-1)+1:length(data)*i);
end
out = [];
in_t = [18;12;6;0;12;18];
out_t =  [9 12 3;6 9 3;3 6 3;0 3 3;6 3 3;9 6 3];
rng('shuffle');
w=50;
g=500;


for k=1:g
    A=in;
    for i=1:2:w
        ch1(1)=rand;
        for j=2:50
            ch1(j)=4*ch1(j-1)*(1-ch1(j-1));        %混沌序列
        end
```

```
    c=2+floor(316*ch1);
    %单点交叉
    temp=A(i,c);
    A(i,c)=A(i+1,c);
    A(i+1,c)=temp;
end


%变异
by=[];
while ~length(by)
    by=find(rand(1,w)<0.1);
end
num1=length(by); B=in(by,:);
for t=1:2*num1
    ch2(t)=randperm(6, 1);
end
for j=1:num1
    bw=sort(2+floor(316*rand(1,2)));
    B(j,bw)=ch2([j,j+1]);
end


G=[in;A;B];
num2=size(G,1); score=zeros(1,num2);
score_x = [];
score_res = [];
score_out = [];




for ee=1:num2
    c_in = G(ee,:); %入车序列

    %计算得分

    score(ee) = x;
    score_x(ee,:) = [x1,x2,x3,x4];
    score_out(ee,:) = c_out;
    long_res = [long_res;res];
end
[slong,ind1]=sort(score,'descend');
```

```
    in=G(ind1(1:w),:);


end
```

```
maxx = score(ind1(1)                    %输出得分
max_x = score_x(ind1(1),:)              %输出四项的得分
max_out = score_out(ind1(1),:)          %输出出车序列
```

得到附件结果

```
%得到每辆车每个时刻的状态
sum_t = 0;
tt = 100 * (100-score_x(ind1(1),4)) + 9 * 318 + 72;
for i=1:ind1(1)-1
    sum_t = sum_t + 100*(100-score_x(i,4)) + 9 * 318 + 72;
end
sum_s = sum_t / 3 * 8 + 1;
sum_e = sum_s + tt / 3 * 8;
max_res = long_res(sum_s:sum_e,:)


kk = tt / 3;
carpos = zeros(length(data), round(tt));  %每辆车每个时刻的状态
pos = [];


for i = 1:kk
    pos = max_res(8 * (i-1)+1:8 * i,:);
    t = 3 * i;
    for r=1:7
        for c=1:10
            if pos(r,c) ~= 0
                if c ~= 1
                    carpos(pos(r,c),t) = r * 10 + 11 - c;
                    if t > 3
                        carpos(pos(r,c),t-2:t-1) = carpos(pos(r,c),t-3);
                    end
```

```
            else
                carpos(pos(r,c),t) = r * 100 + 11 - c;
                if t > 3
                    carpos(pos(r,c),t-2:t-1) = carpos(pos(r,c),t-3);
                end
            end
        end
    end
    end
    if pos(8, 1) ~= 0
        if carpos(pos(8,1),t)==0
            carpos(pos(8,1),t-2:t) = 1;
        end
    end
    if pos(8, 2) ~= 0
        if carpos(pos(8,2),t)==0
            carpos(pos(8,2),t) = 2;
            carpos(pos(8,2),t-2:t-1) = carpos(pos(8,2),t-3);
        end
    end
end


for i=1:length(data)
    for j=1:tt-3
        if carpos(i,j)==2 && carpos(i,j+1) == 0
            carpos(i,j+1:j+2) = 2;
            carpos(i,j+3:end) = 3;
        end
        if carpos(i,j) == 42 && carpos(i,j+1) == 0
            carpos(i,j+1:j+2) = 42;
            carpos(i,j+3:end) = 3;
        end
        if carpos(i,j)==41 && carpos(i,j+1) == 0
            carpos(i,j+1:j+2) = 41;
            carpos(i,j+3:end) = 3;
        end
    end
end


for i=1:length(data)
    for j=4:tt
```

```
        if carpos(i,j) == 0 && carpos(i,j+1) == 1
            carpos(i,j) = 1;
        end
        if carpos(i,j) == 310
            carpos(i,j-3:j-1) = 1;
            break
        end
        if carpos(i,j) == 410 && carpos(i,j-1) == 0
            carpos(i,j-3:j-1)=410;
            break
        end
        if carpos(i,j) == 42 && carpos(i,j+1) == 0
            carpos(i,j+1:j+2) = 42;
            carpos(i,j+3:end) = 3;
        end
        if carpos(i,j) == 710
            carpos(i,j+1:j+2) = 710;
            break
        end
    end
end


ss = zeros(length(data),1);
ss(1,1) = 1;
carpos = [ss carpos];
```