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	电子科技大学
参赛队号	22106140003
队员姓名	1. 谭岩
	2. 厉俊
	3. 张睿智

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问题

摘　　要：

汽车制造厂由于涂装车间与总装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，这就需要在两个车间之间建立涂装-总装缓存调序区（Painted Body Store, PBS），利用多条进车道与返回道将涂装车间的出车序列调整到能够尽量满足总装车间约束的进车序列，因此，PBS 缓存区的调度优化成为亟待解决的问题。

本文建立了 PBS 缓存区进出车调度优化模型，分别实现了缓存区的进车道入口处与出口处的车辆调度。在进车道入口处，设立启发式规则，根据各可用车道的拥塞程度以及当前待调度车辆被调度至各不同进车道所需消耗的时间，选择对应的最优进车道；在进车道出口处，基于贪心准则，判定是否应将该车辆出车并送入总装接车口，或是送入返回道重新进行排序调度。此外，为避免陷入循环调度，根据当前时刻的调度时间开销，对返回道的允许使用次数赋予惩罚项，从而在一定程度上对性能及时间开销进行均衡。

对于问题一，在进车道入口处，以减少车辆抵达进车道右侧出口的时间为目标设立启发式规则，统计当前时刻各进车道的现有车辆数，以及将待入车的车辆送入各进车道的所需时间，对当前车辆抵达进车道出口的送车时间进行估计，并以预期送车时间最短的车道作为当前的目标进车道，通过接车横移机将待入车车辆移至该目标进车道；在进车道出口处，基于现有出车序列，分别计算将当前待出车车辆加入出车序列与不加入出车序列两种情况下的优化目标 1、2 加权和，利用贪心算法，决定应将当前待出车车辆送入 PBS 总装接车口还是送回返车道重新进行调度。为在优化目标 1、2 与返回道使用次数及时间开销之间取得一定平衡，根据当前的时间开销，调整当前时刻返回道剩余允许使用次数。

对于问题二，在进车道入口处，为进一步提高调度效率，以车辆的期望调度时间为依据选择总等待与接送时间之和最短的车辆作为待入车的调度车辆，并根据问题一中最短送车时间为原则选择目标进车道；在进车道出口处，与之类似，以各车辆的最短运输时间（Minimum Transportation Time, MTT）为依据选择对应车道的车辆作为当前待出车的调度车辆，同时基于问题一中所述的贪心原则进行接车决策，即当前待调度车辆应送入 PBS 总

装接车口出车，或是送回返车道位置处重新调度。此外，由于该所述机制倾向于更少的使用返车道，为进一步取得性能-效率均衡，则相对于问题一，将施加一项更高的返车道允许使用次数惩罚值。

对于上述调度机制，在 MATLAB 软件平台下进行实验设计。经过求解得，问题一的附件 1 与附件 2 经过调度后输出出车序列的得分分别为 **9.4520** 与 **33.0810**，问题二附件 1 与附件 2 调度后序列得分分别为 **20.8910** 和 **43.7220**，仿真结果验证了本文所提缓存区进、出车机制的有效性。

关键字：缓存区进出车调度机制　启发式规则　贪心法　性能-效率均衡

目录

图录

表录

1. 问题重述

1.1 引言

汽车制造厂主要由焊装车间、涂装车间、总装车间构成，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。

1.2 问题重述

根据涂装车间的出车序列通过 PBS 调度调整得到总装车间的进车序列。PBS 具体信息如图1所示。PBS 由涂装-PBS 出车口、接车横移机、进车道 6 条（每条进车道有 10 个停

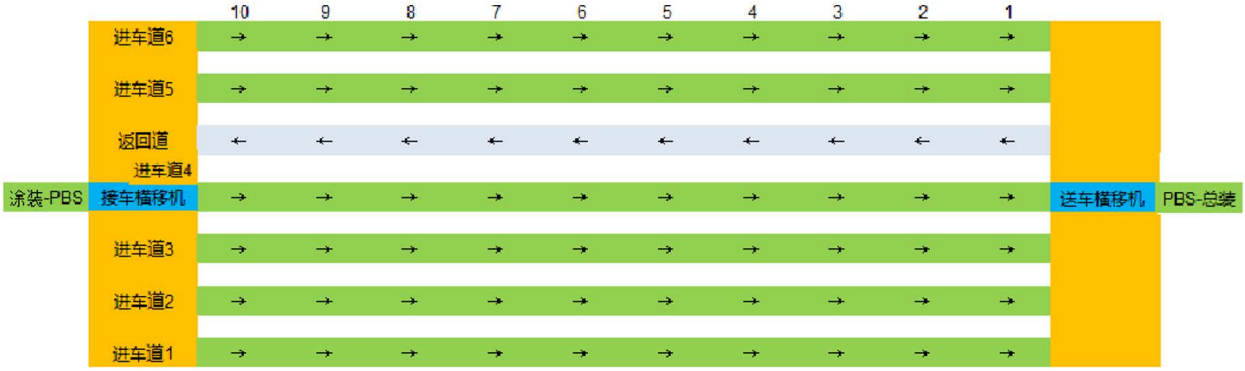


图 1　涂装-总装缓存调序区（PBS）示意图

车位、FIFO 结构）、返回道 1 条（有 10 个停车位）、送车横移机及 PBS-总装接车口等 7 个区域组成。各车道距离等分，每车道宽度 2 米，两相邻车道间间隔 1 米。横移机运动时的速度保持一致。

- **优化目标：**

  1. 混动车型间隔 2 台非混动车型为优，权重系数 0.4，初始分数 100 分。
  2. 四驱车型与两驱车型倾向 1:1 出车序列，权重系数 0.3，初始分数 100 分。
  3. 返回道使用次数倾向于 0，权重系数 0.2，初始分数 100 分。
  4. 倾向总调度时间越短越好，权重系数 0.1，初始分数 100 分。

建模完成以下优化问题：

- **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，使得总装进车序列尽可能满足总装生产需求。给出优化调度方案分别应用于附件 1 和附件 2 数据的得分结果，并将使用附件 1 和附件 2 两套数据的调度输出结果按照附件 4 格式分别存入 result11.xlsx 和 result12.xlsx。

- **问题 2:** 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。给出将优化调度方案分别应用于附件 1 和附件 2 数据的得分结果，并将使用附件 1 和附件 2 两套数据的调度输出结果按照附件 4 格式分别存入 result21.xlsx 和 result22.xlsx。
- **PBS 相关时间数据：**
  1. 进车过程：对于 1-6 进车道，消耗时间分别为 [18, 12, 6, 0, 12, 18] 秒。
  2. 出车过程：对于 1-6 进车道，消耗时间分别为 [18, 12, 6, 0, 12, 18] 秒。
  3. 车进返回道过程：对于 1-6 进车道，消耗时间分别为 [24, 18, 12, 6, 12, 18] 秒。
  4. 车出返回道过程：于 1-6 进车道，消耗时间分别为 [24, 18, 12, 6, 12, 18] 秒。
  5. 车移动过程：任意车身在进车道/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。

2. 模型的假设

- **PBS 时间数据假设：**
  1. 忽略横移机卸载（装载）车身到车道 1（10）的时间.
  2. 忽略车身在进车缓冲区（出车缓冲区）处装载（卸载）的时间。
- **PBS 约束假设：**
  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。

3. 符号说明

符号	意义
N	车身总数（$N = 318$）
N_m	混合动力车身总数
N_n	分块出车序列总数
M_α	车道数 ($M_\alpha = 7$)，前 6 个为进车道，第 7 个为返回道
M_β	停车位数 ($M_\beta = 10$)
$a_{i,j}$	涂装车间的出车序列矩阵 $a_{i,j}, i = 1, \ldots, N, j = 1, 2, 3$
$b_{i,j}$	总装车间的进车序列矩阵 $b_{i,j}, i = 1, \ldots, N, j = 1, 2, 3$
$x_{i,j,k}$	第 i 辆车是否在第 j 车道第 k 个停车位上停留
c_j^N	每个车道的当前汽车数量 $c_j^N, j = 1, \ldots, M_\alpha - 1$
$t_{f,j}^{in}$	第 j 个车道进车过程所需时间 $t_{f,j}^{in}(j = 1, \ldots, M_\alpha - 1)$
$t_{f,j}^{out}$	第 j 个车道出车过程所需时间 $t_{f,j}^{out}(j = 1, \ldots, M_\alpha - 1)$
$t_{b,j}^{in}$	第 j 个车道车进返回道过程所需时间 $t_{b,j}^{in}(j = 1, \ldots, M_\alpha - 1)$
$t_{b,j}^{out}$	第 j 个车道车出返回道过程所需时间 $t_{b,j}^{out}(j = 1, \ldots, M_\alpha - 1)$
t_i^w	第 i 辆车身在 PBS 中的等待时间 t_i^w
$t_{j,n}^e$	第 n 秒车道 j 最前面的车身到车道口的时间 $t_{j,n}^e$
t_m	从某一停车位移动至后一停车位消耗时间 ($t_m = 9s$)
t_s	第一辆车离开涂装车间进入 PBS 的当前时间 t_s
t_e	最后一辆车离开 PBS 进入总装车间的当前时间 t_e
χ_{il}	χ_{il} 为整数决策变量，表示是否将车辆 i 送至进车道 j，$i = 1, \ldots, N, j \in L$
c_i	出车序列中第 i 辆混合车型的位置索引序列
$\delta_{1,i}$	第 i 与第 $i + 1$ 辆混动车型是否间隔 2 台非混动车型指标 $q_{1,i}, i = 1, N_m - 1$
$\delta_{2,i}$	混动车型是否间隔 2 台非混动车型指标 $q_{1,i}, i = 1, N_m - 1$
d_i	四驱车型与两驱车型分块出车序列 ($i = 1, \ldots$)
$N_{b,i}$	车身 $a_{i,j}$ 使用返回道的次数 $N_{b,i}, i = \ldots, N$
T_i	车身 $a_{i,j}$ 在 PBS 的总调度时间 $T_i, i = 1, \ldots, N$
w_i	第 i 个优化目标权重系数 $w_i, i = 1, \ldots 4$
Q_i	第 i 个优化目标得分值 $p_i, i = 1, \ldots 4$

4. 问题一的分析与求解

4.1 问题一的分析

涂装车间缓冲区排序主要涉及确定两个策略，一个是车辆进入 PBS 缓存区时如何选择需放置的进车道（进车过程），另一个就是缓存区移除车辆时应选择直接送入接车口出车还是放置于返回道从而尽可能保持队列车型的规律性。总装车间在安排生产时主要是在保证能按时交货的前提下保证生产负荷平衡和物流消耗平准化等，需要考虑的是车的动力类别和驱动方式，因此，出车序列中相邻车辆的动力和驱动方式需要满足一定程度的规律性，从而使后续工艺环节的涂装刷头不必要频繁更换，即：

1. 混动车型间隔 2 台非混动车型为优。
2. 四驱车型与两驱车型倾向 1:1 出车序列。

返回道的主要作用在于若当前等待出车车辆违背于上述规律时，可以考虑将该车辆送入返回道，并在送车侧重新进行调度排序。值得注意的是，在实际中，由于工厂车辆订单的不确定性，上述规律只能根据订单中不同种类车辆的数量尽可能设计相对合适的出车序列，而很难保证存在完全满足该规律的调度方法。

总体来看，需考虑的是将车尽快送入出车道，并由出车侧根据当前已有出车序列、待出车序列来决定是否将车送入返回道从而重新调度。因此，在进车侧应侧重考虑减小每一进车车辆经过进车道到出车侧的时间，而出车侧的调度应综合考虑出车序列的排列规律、返回道使用次数及缓冲区排序的总消耗时间。

4.1.1 进车过程分析

对于 PBS 缓冲区进车调度过程，其进车规则相对简单，即若返回道 10 车位有车则优先调度该位置车辆，否则按序调度进入涂装 PBS 出车口的车辆，并将待调度车辆放入合适进车道。如前文所述，在进车侧需考虑尽快将每一车辆送至出车道从而保证出车调度效率，设 $i \in \{m, \ldots, N\}$ 为待调度车辆序列，包括等待进入涂装 PBS 出车口的车辆、返回道的车辆，$\{1, \ldots, m-1\}$ 为已出车及已进入进车道的车辆，则可构建进车过程中车辆的最短送车时间问题为

$$
\begin{aligned}
P1): \ & \min_{\chi_{i,l}} t_i \\
& \text{s.t.} \begin{cases} \sum_{l \in L} \chi_{i,l} = 1, \forall i \in \{m, \ldots, N\} & (a) \\ c_j^N \leq 10, \forall j \in L & (b) \end{cases}
\end{aligned} \tag{1}
$$

其中 $L = \{1, \ldots, 6\}$ 为进车道集合，t_i 为每个待调度车辆的最短送车时间，χ_{il} 为整数决策变量，表示是否将车辆 i 送至进车道 j，c_j^N 为每个车道的当前汽车数量。约束 (a) 表示每个车辆最多只能被送入一个进车道，约束 (b) 限制每个进车道中当前时刻的最大车辆数。

送车时间分为两部分，一部分为接车横移机将车辆送入返回道所消耗的时间，另一部分为车辆在进车道入口到出车道口的等待时间。在理想状态下，车辆自进入进车道到离开，消耗时间均为 81 秒，然而在实际中，车辆到达进车道出口位置仍需等待送车横移机的调度，这一等待时间的估计值可为 $\text{mean}(t_{f,j}^{out})$。因此，对于每一进车车辆，可根据当前 6 个进车道中车的数量及送入当前返回道所消耗的时间来作为评估进车道选择依据。即

$$t_{i,j} = c_j^N * \text{mean}(t_{f,j}^{out}) + t_{f,j}^{in}, \quad \text{if} \quad y_{il} = 1, \tag{2}$$

其中 $t_{i,j}$ 为当前车辆 i 送至进车道 j 的估计送车时间，且有 $t_i = \min\limits_j t_{i,j}$。对于进车侧的待

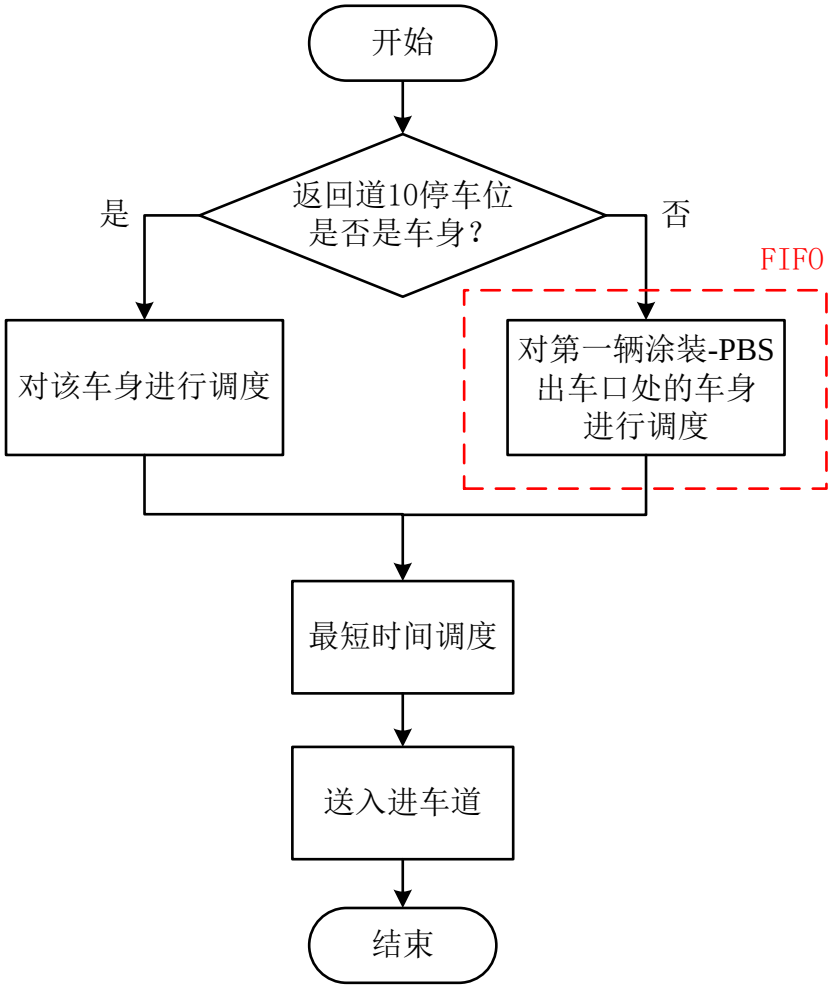


图 2　进车流程图

调度车辆，可根据上式计算将其调度至每一进车道的估计送车时间，并选择计算结果最小的进车道将其送入。综上所述，涂装车间缓冲区进车过程的启发式规则如下：

1) 待送车横移机完成上一任务回到初始位置，若返回道 10 车位有车则将该车设置为当前待调度车辆；

2) 若返回道 10 车位不存在，则将 PBS 出车口的第一辆车设置为当前待调度车辆；
3) 根据各进车道的车辆数，结合接车横移机送至各进车道的消耗时间，计算该车送至每个进车道的估计送车时间 $t_{i,j}$；
4) 选择估计送车时间最小的进车道，将待调度车辆送入该进车道。

基于最短送车时间的进车调度流程如图2所示

4.1.2 出车过程分析

不同于进车过程，出车侧需要保证最终送入 PBS 总装区的出车序列在动力形式和驱动数上保持一定的规律性，因此不能单纯考虑调度时间。返车道可以将影响规律性的到达送车口的车辆送回进车道入口，从而辅助出车序列的排序。因此，通过合理利用返回道，能够使整体车型序列尽量满足实际生产需求。

对于此种情况，易想到的思路是可以将问题中所给出的四个优化目标的加权和作为优化目标函数，以每个接车口的接车决策及每个送车口的送车决策作为优化变量，构建一个以最大化目标函数值为目标的优化问题进行求解，文献 [1, 2, 3, 4, 5, 9] 中所述方法均是基于上述思想。然而，这种方法并未考虑返回道的使用以及接送车的时间消耗，而是集中于理想的无时间消耗的 PBS 缓冲区排序模型，这会在实际中造成性能的损耗。此外，对于该 NP-hard 问题，智能算法（遗传算法、模拟退火等）几乎是唯一效果较好的求解方式。然而，随着车辆数量的增加，该优化问题中所涉及的优化变量规模也将大幅增长，这会造成求解的困难。在第6.章中将展示无时间消耗、无返回道的 PBS 简化模型基于遗传算法的接送车决策求解结果。

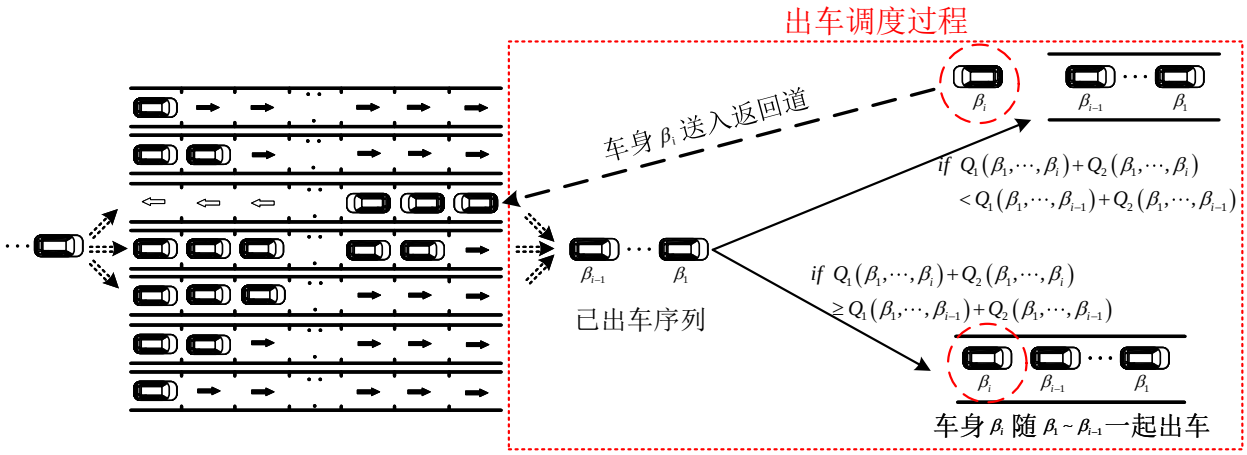


图 3　动态出车过程流程图

基于上述分析，为控制问题规模，可以将每一次出车过程视作为一个独立的决策过程。即，给定已送入 PBS-总装接的序列 $\{\beta_1, \beta_2, \ldots, \beta_{i-1}\}$，以及当前最先到达送车口处的待出车车辆 β_i、返回道使用次数及当前车辆数等条件，决定将 β_i 送入 PBS-总装接车口构成新序列 $\{\beta_1, \beta_2, \ldots, \beta_{i-1}, \beta_i\}$，还是将 β_i 送回返回道，出车序列维持原序列 $\{\beta_1, \beta_2, \ldots, \beta_{i-1}\}$。

即每一次出车决策都尽可能使当前出车判断原则应基于新序列与原序列的规律性序列满足最优的规律性，其动态出车过程如图3所示。

4.2 PBS 优化调度模型

对于每一次出车决策过程，定义涂装车间的出车序列矩阵 $\{a_{i,j}\}$，其中 $\{a_{i,1}\}$ 表示出车顺序，$\{a_{i,2}\}$ 表示车身是否为混动车型，$\{a_{i,3}\}$ 表示车身是为二驱还是四驱，

$$a_{i,1}=i,\quad a_{i,2}=\begin{cases}1, & \text{if 混动}\\ 0, & \text{if 非混动}\end{cases}\qquad a_{i,3}=\begin{cases}2, & \text{if 二驱}\\ 4, & \text{if 四驱}\end{cases}\tag{3}$$

总装车间出车序列矩阵 $\{b_{i,j}\}$ 类似定义。

已知涂装车间的出车序列 $\{a_{i,1}\}_{i=1}^{N}$, 考虑 N 个车身需要在汽车制造涂装-总装缓存调序区（PBS，Painted Body Store）的 M_α 条车道调度调整的问题，调整后得到总装车间的进车序列 $\{b_{i,1}\}_{i=1}^{N}$，PBS 调度示意图如图4所示。假设车身在车道内移动时不可被调度，横移机在执行任何动作时均不能被打断。每条车道上有 M_β 个停车位，每个停车位至多能停一辆车身，车身在停车位之间移动需要恒定的时间 t_m，且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。

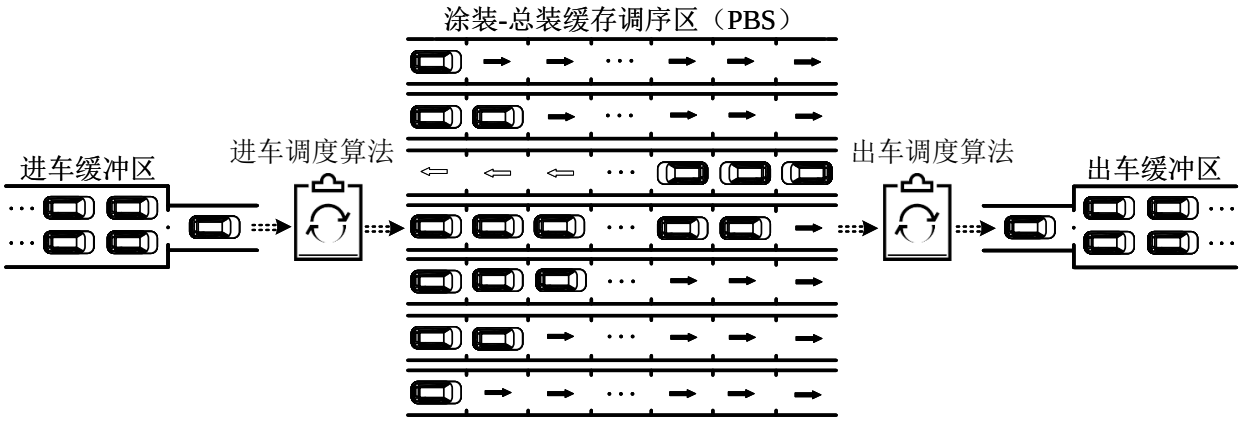


图 4　PBS 调度示意图

当缓冲区进车序列 $\{a_{i,1}\}_{i=1}^{N}$ 的第 i 辆车身 $a_{i,1}$ 放置到接车横移机时，我们遵循先到先服务的 FIFO（First In First Out）原则，将车身 $a_{i,1}$ 通过进车调度算法移到第 j 条进车道，花费时间 $t_{f,j}^{in}, j=1,\ldots,M_\alpha-1$，其中 $1,\ldots,M_\alpha-1$ 和 M_α 车道分别为进车道和返回道，进车道和返回道上停车位 k 的移动方向分别为 $M_\beta,\ldots,1$ 和 $1,\ldots,M_\beta$。进入进车道 j 的第 k 个停车位后，若前方出现空位车身经过时间 t_m 后自动移动到第 $k-1$ 个车位，直至到达第 j 车道的 1 停车位。这时遵循最小等待时间的 LWT（Least Waiting Time）原则，进行出车调度算法进行判断，若判断出车，则送车横移机花费 $t_{f,j}^{out}$ 时间将车身 $a_{i,1}$ 从车道 j 移至 PBS 总装车间出车，得到出车序列 $b_{i,1}, i=1\ldots,N$，若判断返回，则送车横移机将车道 j

上 1 停车位的车身运送至返回道（车道 M_α）1 停车位，花费时间 $t_{b,j}^{in}$，之后车辆在返回道（车道 M_α）按车道移动规则返回，每次移动车位花费时间 t_m。特别地是，当返回道的车辆到达 10 停车位时，比进车序列 $\{a_{i,1}\}$ 的车身优先处理。

4.2.1 基于 FIFO 和 RRF 原则的启发式进车调度算法

车身 $a_{i,1}$ 从涂装车间出车序列 $\{a_{i,1}\}_{i=1}^{N}$ 调整后的道总装车间的进车序列 $\{b_{i,1}\}_{i=1}^{N}$ 花费的总时间为 T_i，车身 $a_{i,1}$ 在 PBS 中等待的总时间为 t_i^w。若车身 $a_{i,1}$ 不进行返回，则车身 $a_{i,1}$ 的调度总时间 T_i 为

$$T_i = t_{f,j}^{in} + 9t_m + t_{f,j}^{out} + t_i^w, \tag{4}$$

若车身 $a_{i,1}$ 进行返回，则需要额外花费在返回道道内的移动时间 $9t_m$，进车时间 $t_{f,j}^{in}$ 和出车时间 $t_{f,j}^{out}$，假设车身 $a_{i,1}$ 进入返回道的次数为 $N_{b,i}$，则车身 $a_{i,1}$ 的调度总时间 T_i 为

$$T_i = t_{f,j}^{in} + 9t_m + t_{f,j}^{out} + t_i^w + N_{b,i}(t_{f,j}^{in} + 9t_m + t_{f,j}^{out}). \tag{5}$$

由于每个车道只有 M_β 个停车位，且每个停车位至多只能容纳 1 个车身，定义车道 j 停车位 k 上是否有车身的指标

$$x_{i,j,k} = \begin{cases} 1, & \text{占用} \\ 0, & \text{空闲} \end{cases}, i = 1, \ldots, N, j = 1, \ldots, M_\alpha - 1, k = 1, \ldots, M_\beta, \tag{6}$$

每个车道 j 上需要满足停车位数量约束

$$\sum_{k}^{M_\beta} x_{i,j,k} \leq M_\beta, \quad \forall i = 1, \ldots, N, \forall j = 1, \ldots, M_\alpha - 1.$$

对于 PBS 约束条件 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，定义车身由横移机卸载到各进车道 10 停车位操作对应的决策变量 $V_{i',j}$

$$V_{i',j} = \begin{cases} 1, & \text{if 将第 } i' \text{ 辆车身移至车道} j \\ 0, & \text{Otherswise} \end{cases}. \tag{7}$$

我们遵循 FIFO 原则对进车序列进行调度，将车身 $a_{i,1}$ 通过进车调度算法移到第 j 条进车道，并且要满足 PBS 约束条件 6，即返回道优先 (RRF，Return Road First) 原则，若满足 PBS 约束条件 6 则进行如下操作

$$V_{i',j} = \begin{cases} 1, & \text{if } x_{i,7,10} = 1, \forall i = 1, \ldots, N, j \in L \\ 0, & \text{if } x_{i,7,10} = 0, \forall i = 1, \ldots, N, j \in L \end{cases}. \tag{8}$$

若不满足 PBS 约束条件 6，则按照 $t_{f,j}^{in}$ 和车道空闲状态 $x_{i,j,k}$, $j \in L$, $k = 1$ 按照 FIFO 原则和 RRF 原则进行入车过程调度，见4.2.1节 Algorithm 1。

Algorithm 1: 问题一入车过程

Input: 进车道当前车位状态 $x_{i,j,k}$，进车序列 a_{i1}，进车道进车过程所需时间 $t_{f,j}^{in}$，返回道出车过程所需时间 $t_{b,j}^{out}$，进车道当前车身数量 c_j^N，进车序列 a_{i1}

Output: 进车道车位状态 $x_{i,j,k}$，进车道当前车身数量 c_j^N，进车序列 a_{i1}

Step A：检查横移车是否完成上一送车任务；
if < 返回道出口位置非空 ($x_{i,7,10} = 1$)> **then**
    Step B：选择返回车道作为下一时刻出车
else if < 返回道出口位置为空 ($x_{i,7,10} = 0$)> **then**
    Step B：选择进车序列 a_{i1} 第一辆车作为下一时刻出车
end if
Step C：根据进车道车数 c_j^n, 进车道进车过程所需时间 $t_{f,j}^{in}$，返回道出车过程所需时间 $t_{b,j}^{out}$，基于式（2），计算对应目标进车道 j^{tar}
while < 当前选定进车道 j^{tat} 有车 ($x_{i,j^{tar},10} = 1$)> **do**
    Step D：选择估计送车时间第二小的进车道作为目标车道 j^{tar}，以此类推，直至满足目标进车道 j^{tar} 的 10 停车位无车
end while
Step E：基于 $t_{f,j}^{in}$, $t_{b,j}^{out}$ 更新 $x_{i,j,k}$, c_j^N, $a_{i,j}$

4.2.2 基于 LWT 原则的出车调度算法

对于 PBS 约束条件 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，设最先到达 1 停车位的车身所在车道为 j^{fir}，定义出车或返回决策变量 P_j

$$P_j = \begin{cases} 1, & \text{将车道 } j \text{ 的车身出车} \\ 0, & \text{将车道 } j \text{ 的车身送回返回道} \end{cases} . \tag{9}$$

1、优化目标评价指标

为了使得车身调度尽可能高效，定义相应的优化目标得分 $Q_i, i = 1, \ldots, 4$。对于优化目标 Q_1，需要检查出车序列，找到所有混动车身，从出车序列头部按照先后顺序开始算，计算每连续两辆混动车身之间的非混动车身数，不等于 2 则扣 1 分。设出车序列 $\{b_{i,1}\}_{i=1}^{N}$ 中混动车型数量为 N_m，混动车型在总装车间入车序列中的顺序 c_l 为，

$$c_l = \begin{cases} i, & \text{if } b_{i,2} = 1 \\ \text{空}, & \text{if } b_{i,2} = 0 \end{cases}, \tag{10}$$

接着定义混动车型是否间隔 2 台非混动车型指标 $\delta_{1,i},\ i = 1, \ldots, N_m - 1$,

$$\delta_{1,i} = \begin{cases} 1, & \text{if } c_{i+1} - c_i - 1 = 2 \\ 0, & \text{Otherswise} \end{cases}, i = 1, \ldots, N_m - 1, \tag{11}$$

则优化目标 Q_1 得分

$$Q_1 = \sum_{i=1}^{N_m - 1} \delta_{1,i}, \quad i = 1, \ldots, N_m - 1.$$

对于优化目标 Q_2，需要对总装车间入车序列 $\{b_{i,1}\}$ 进行分块，共分 N_n 块，得到分块序列 $\{d_i\}_{i=1}^{N_n}$

$$d_j = \begin{cases} b_{i,3}, & \text{if } b_{i,3} \neq b_{i+1,3} \\ \text{空}, & \text{Otherswise} \end{cases}, i = 1, \ldots, N_m - 1, \tag{12}$$

判断每一分快中的四驱车型与两驱车型之比是否满足 1:1，若不满足，则扣 1 分。定义混动车型是否间隔 2 台非混动车型指标 $\delta_{1,i},\ i = 1, \ldots, N_n - 1$,

$$\delta_{2,i} = \begin{cases} 1, & \text{if } d_i\text{中四驱车型与两驱车型之比为 1:1} \\ 0, & \text{Otherswise} \end{cases}, i = 1, \ldots, N_n - 1, \tag{13}$$

则优化目标 Q_2 得分为

$$Q_2 = \sum_{i=1}^{N_n} \delta_{2,i}, \quad i = 1, \ldots, N_n.$$

对于优化目标 Q_3，涂装车间出车序列 $\{a_{i,1}\}_{i=1}^{N}$ 在出车调度算法的计算下进行 $N_{b,i}$ 次返回道操作，则优化目标 Q_2 得分为

$$Q_3 = \sum_{i=1}^{N} N_{b,i}, \quad i = 1, \ldots, N.$$

对于优化目标 Q_4，涂装车间出车序列 $\{a_{i,1}\}_{i=1}^{N}$ 长度为 N，设涂装车间出车序列 $\{a_{i,1}\}$ 第一辆车 a_1 进入 PBS 的当前时间为 t_s，最后一辆车 a_N 离开 PBS 进入总装车间的当前时

间为 t_e，则总完成时间 T 为

$$T = t_e - t_s,$$

其理论最快完成时间为 $9N + 72$(全部走进车道 4，出车序列与入车序列相同)，其时间惩罚值设置为

$$Q_4 = 0.01 \times (T–9N - 8t_m) \tag{14}$$

其中优化目标 $\{Q_i\}_{i=1}^4$ 对应的权重系数 $\{w_i\}_{i=1}^N$ 分别为 $w_i = 0.4, 0.3, 0.2, 0.1, i = 1, \ldots, 4$，且各权值系数和 $\sum_{i=1}^4 = 1$。

Algorithm 2: 问题一出车过程

Input: 已出车序列 $b_{i,j}$，车道状态 $x_{i,j,k}$，进车道出车过程所需时间 $t_{f,j}^{out}$，返回道进车过程所需时间 $t_{b,j}^{in}$，进车道当前车身数量 c_j^N

Output: 进车道车位状态 $x_{i,j,k}$，进车道当前车身数量 c_j^N

Step A：检查送车横移机是否完成上一接车任务
if < 仅有一个进车道 1 车位有车 > **then**
    Step B：则将其作为接车车辆
else if < 多个进车道 1 车位有车 > **then**
    Step B：则按照 LWT 原则以最先到达 1 车位的车作为接车车辆
else
    Step B：接车横移机设置为空闲状态
end if
Step C：基于一出车序列 $b_{i,1}$，调用子程序计算当前 Q_1, Q_2 加权得分值 f_1
Step D：假设当前接车车辆已出车，调用子程序计算当前 Q_1, Q_2 加权得分值 f_2
if <$f_1 > f_2$> **then**
    Step E：将当前接车辆送入返回道
else
    Step E：将当前接车辆加入出车序列
end if
Step F：基于 $t_{f,j}^{out}$, $t_{b,j}^{in}$ 更新 $x_{i,j,k}$, c_j^N

2、基于 MILP 的出车模型

问题一的 PBS 优化调度模型可以表述为如下的混合整数线性规划 (MILP，Mixed Integer Linear Programming) 模型：

其中，目标函数为

$$(P3): \min_{V_{i',j},P_j} \sum_{i=1}^{4} w_i * (100 - Q_i),$$

约束条件为

$$\text{s.t.} \begin{cases} \sum_{k=1}^{M_\beta} x_{i,j,k} \le M_\beta, & \forall i = 1,\ldots,N, \forall j = 1,\ldots,M_\alpha \\ T_i = t_{f,j}^{in} + 9t_m + t_{f,j}^{out} + t_i^w & \\ \quad +N_{b,i}(t_{f,j}^{in} + 9t_m + t_{f,j}^{out}), & i = 1,\ldots,N \\ V_{i',j} = 1, & \text{if } x_{i,7,10} = 1, \forall i = 1,\ldots,N, j \in L \\ P_j, & \text{if } x_{i,j,1} = 1, \forall j \in L \\ T = t_e - t_s & \\ Q_1 = \sum_{i=1}^{N_m - 1} \delta_{1,i}, & i = 1,\ldots,N_m - 1 \\ Q_2 = \sum_{i=1}^{N_n} \delta_{2,i} & i = 1,\ldots,N_n \\ Q_3 = \sum_{i=1}^{N} N_{b,i} & \\ Q_4 = 0.01 \times (T - 9N - 8t_m) & \\ \sum_{i=1}^{4} w_i = 1, & i = 1,\ldots,4 \\ w_i = 0.4, 0.3, 0.2, 0.1, & i = 1,\ldots,4 \\ t_{f,j}^{in} = 18, 12, 6, 0, 12, 18, & j = 1,\ldots,M_\alpha - 1 \\ t_{f,j}^{out} = 18, 12, 6, 0, 12, 18, & j = 1,\ldots,M_\alpha - 1 \\ t_{b,j}^{in} = 24, 18, 12, 6, 12, 18, & j = 1,\ldots,M_\alpha - 1 \\ t_{b,j}^{out} = 24, 18, 12, 6, 12, 18, & j = 1,\ldots,M_\alpha - 1 \\ t_m = 9 & \\ x_{i,j,k} \in \{0,1\} & \forall i = 1,\ldots,N, \forall j = 1,\ldots,M_\alpha, k = 1,\ldots,M_\beta \\ \delta_{1,i} \in \{0,1\} & \forall i = 1,\ldots,N_m - 1 \\ \delta_{2,i} \in \{0,1\} & \forall i = 1,\ldots,N_n \end{cases} \tag{15}$$

4.2.3 基于贪心原理的启发式算法

上述所述优化模型为一混合整数非凸优化模型，且问题规模随着车辆数量的增加而增大，即使采用可以趋近全局最优解的智能算法也需要较长的求解时间。针对于此，本文提出了一种基于贪心法的求解方法，即，从问题的某一个初始状态出发，在每一个阶段都根据贪心策略来做出当前最优的决策，逐步逼近给定的目标，尽可能快地求得更好的解。贪心法可以通过逐步的局部最优来达到最终的全局最优。

贪心法本身具有以下特点：

1. 每个阶段贪心法都做出对眼前来讲是最有利的选择，不考虑该选择对将来是否有不良影响；在本设计中，对应的就是每次出车决策仅考虑使得当前评价指标达到最优 (如图 3 所示)，不考虑该决策是否会影响未来出车序列的评价值。
2. 每个阶段的决策一旦做出，就不可更改、不允许回溯；在本设计中，对应的是待出车车辆一旦选择送入 PBS 总装口或选择送入返回车道，都将是确定规则，而不能出现接车横移机停止或选择接送其他车道车辆。
3. 贪心选择性质所做的是一个非线性的子问题处理流程，即一个子问题并不依赖于另一个子问题，但是子问题间有严格的顺序性；对应于本设计，每一次出车决策并不依附于后续的出车序列，但性能的评价需要基于现有已出车序列。

综上所述，基于贪心法的出车序列排序问题包括以下步骤：

- **分解：**将原问题分解为若干个相互独立的阶段，每个阶段对应对于当前最先到达进车道 1 车位的车辆的出车决策，即将其送入 PBS 总装口或选择送入返回车道；
- **解决：**对于每个阶段，依据图3和4.2.2节中 Algorithm 2 所示的贪心策略进行贪心选择，求出局部的最优决策；
- **合并：**待所有车辆完成出车过程，将各个阶段的解合并为原问题的一个可行解，即出车序列及每个时刻车辆所处位置（同附件 4 形式）。

一个问题能够分解成各个子问题来解决，通过各个子问题的最优解能递推到原问题的最优解。那么原问题的最优解一定包含各个子问题的最优解，这是能够采用贪心法来求解问题的关键。本文所提出的基于贪心法的最优出车序列决策算法流程见4.2.2节中 Algorithm 2。

4.3 模型的求解

接下来将对题目所给附件 1、附件 2 利用上述机制对其进行缓存区进出车调度求解。如表1、2所示，附件 1 中混动与燃油车辆数比例约为 2：1，而附件 2 中这一比值为 1：1，因此，相较于附件 1，附件 2 的 Q1 优化目标值更难提升。

表 1　附件 1 附件 2 动力类型分布

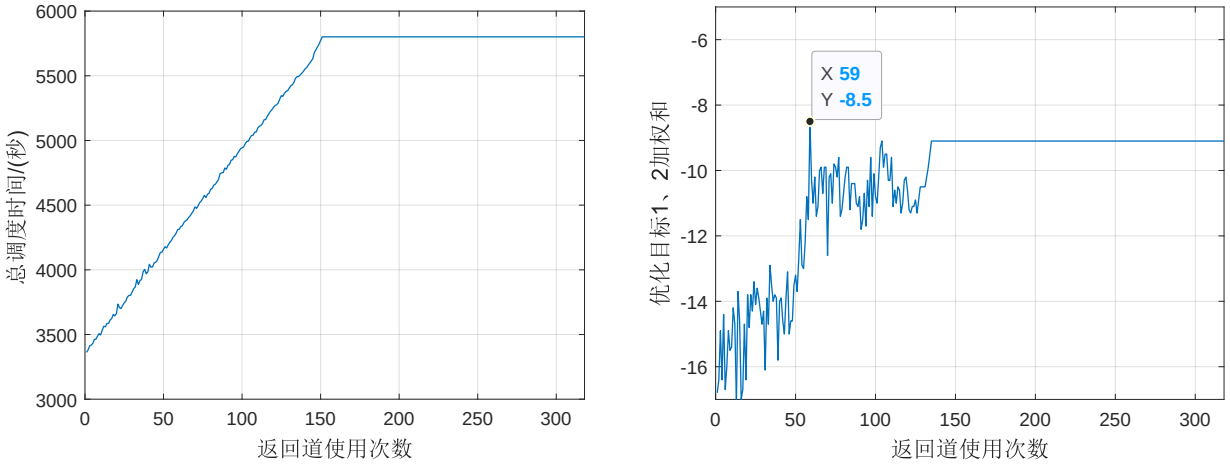
车型	混动	燃油	比例
附件 1	212	105	**2.01904762：1**
附件 2	159	159	**1:1**

基于 4.2.1 节中所述基于启发式规则的进车调度和 4.2.4 中所述基于贪心法的出车调度算法，利用 MATLAB 软件对问题一的附件一、二所给数据进行求解。其中，利用一个 6*90 的矩阵表示每一时刻（秒）的 6 个进车道车位存储车辆状态，其中 1-9 列代表车位 10，

表 2　附件 1 附件 2 驱动类型分布

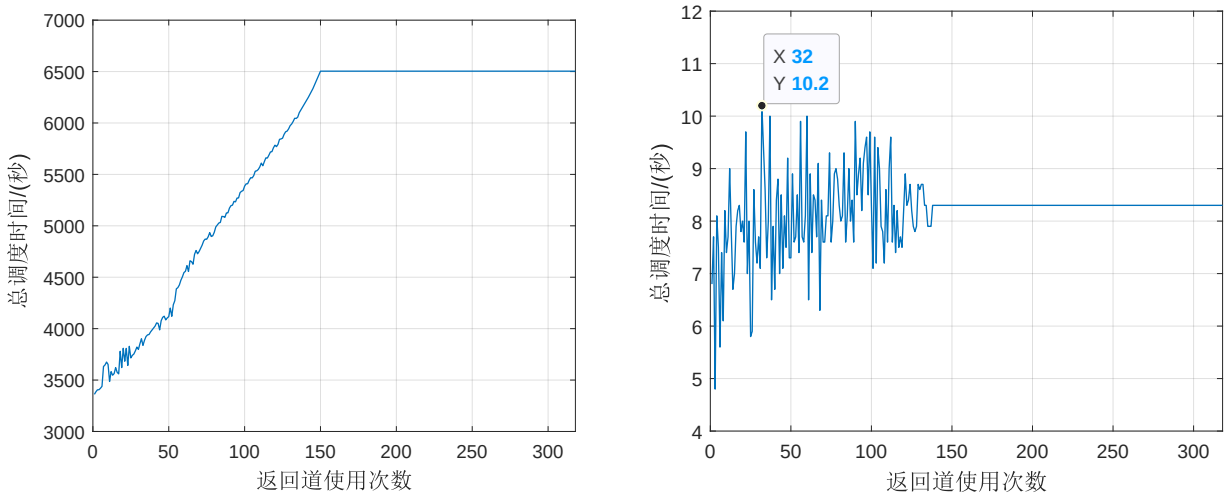
车型	两驱	四驱	比例
附件 1	288	29	**9.9310345:1**
附件 2	289	29	**9.9310345:1**

10-18 代表车位 9，以此类推。当某一车位的 9 列索引中有车辆时，后一车位的车辆不可移入。例如，若车道矩阵的第 3 行第 19 列为 10，则三车道的 8 车位为附件中的车辆 10 占有。同理，设置一个 1*90 的矩阵表示每一时刻（秒）的返回道存储车辆状态，其元素意义与进车道矩阵相同、车辆移动方向相反。



(a) 总调度时间　　(b) 优化目标 1、2 加权值

图 5　问题一附件 1 总调度时间 (a) 和优化目标 1、2 加权值 (b) 随返回道使用次数的变化



(a) 总调度时间　　(b) 优化目标 1、2 加权值

图 6　问题一附件 2 总调度时间 (a) 和优化目标 1、2 加权值 (b) 随返回道使用次数的变化

返回道的使用次数会影响 Q_1、Q_2 的评价结果，理论上返回道使用次数越多，则进行的调序次数越多，于是会有更多子阶段达到局部最优值。然而实际中，不可能允许无限次使用返回道，这会导致时间的进一步增大甚至陷入轮回，也因此，题目中设置了 Q_3、Q_4 评价对返回道使用次数进行限制。为了达到 Q_1/Q_2 与 Q_3/Q_4 之间的平衡，本文设置不同的返回道使用次数上限作为值可变的自变量，在不同的使用上限次数下分别进行仿真，本文设置上线次数范围为 0-318 次。根据所给出的进车、出车时间项 $(t_{f,j}^{in}, t_{f,j}^{out}, t_{b,j}^{in}, t_{b,j}^{out})$，对进入和出车两侧进行仿真，具体代码见论文附录。

表 3　问题一 PBS 缓冲区出车序列结果

	PBS 缓冲区输出序列
附件 1	1 3 4 6 7 8 11 14 13 18 12 15 16 2 20 21 22 5 24 27 29 26 9 28 31 33 32 30 34 36 35 37 25 43 41 10 17 40 46 19 51 52 50 23 45 39 53 56 57 42 58 60 44 47 62 61 65 66 64 54 68 67 70 72 59 75 55 73 78 63 76 81 82 83 80 85 86 38 84 89 90 91 92 94 88 96 97 98 71 100 87 95 103 104 77 102 106 108 79 105 110 112 48 49 109 69 114 115 116 117 119 118 120 107 122 123 93 124 126 127 111 130 128 113 133 131 99 134 101 135 136 74 137 138 121 139 140 125 141 143 142 129 144 145 132 146 147 148 149 150 151 153 152 154 155 156 157 159 158 160 162 161 163 165 166 164 167 168 169 171 170 172 173 174 175 176 177 179 178 180 181 182 183 185 184 186 187 188 189 191 192 190 193 195 194 196 198 197 199 200 201 202 204 205 203 206 208 207 209 211 210 212 213 214 215 217 218 216 219 221 220 222 224 223 225 226 227 228 230 231 229 232 234 233 235 237 236 238 239 240 241 243 244 242 245 247 246 248 250 249 251 252 253 254 256 257 255 258 260 259 261 263 262 264 265 266 267 269 270 268 271 273 272 274 276 275 277 278 279 280 282 283 281 284 286 285 287 289 288 290 291 292 293 295 296 294 297 299 298 300 302 301 303 304 305 306 308 309 307 310 312 311 313 315 314 316 317 318
附件 2	1 3 4 5 6 7 8 13 9 12 16 17 11 20 19 21 24 22 23 26 27 29 25 32 31 35 37 15 18 14 42 41 38 28 43 10 46 45 2 30 53 50 51 52 54 55 56 57 34 60 59 58 48 61 63 64 47 39 66 67 69 70 71 72 73 68 74 49 75 79 80 76 78 82 83 85 86 40 88 90 89 36 93 92 91 44 77 62 95 97 98 65 100 81 96 94 103 101 99 33 102 84 105 107 108 109 112 110 106 87 117 113 111 115 114 104 119 121 122 125 120 116 126 129 123 124 130 127 128 118 132 134 135 138 136 133 139 142 140 137 143 144 141 131 146 148 149 152 150 147 153 156 154 151 157 158 155 145 160 162 163 166 164 161 167 170 168 165 171 172 169 159 174 176 177 180 178 175 181 184 182 179 185 186 183 173 188 190 191 194 192 189 195 198 196 193 199 200 197 187 202 204 205 208 206 203 209 212 210 207 213 214 211 201 216 218 219 222 220 217 223 226 224 221 227 228 225 215 230 232 233 236 234 231 237 240 238 235 241 242 239 229 244 246 247 250 248 245 251 254 252 249 255 256 253 243 258 260 261 264 262 259 265 268 266 263 269 270 267 257 272 274 275 278 276 273 279 282 280 277 283 284 281 271 286 288 289 292 290 287 293 296 294 291 297 298 295 285 300 302 303 306 304 301 307 310 308 305 311 312 309 299 314 316 317 318 315 313

问题一附件 1 及附件 2 在不同最大返回道使用次数限制下的仿真结果如图5、6所示。其中的图 (a) 表示随着返回道使用次数限制下的总调度时间变化，不难看出，若返回道使用次数增加，则总的调度时间也会随之增长。此外，在 150 以后，尽管后续允许使用返回道次数的上限进一步增大，但总的调度时间趋于平稳，亦即达到了时间-返回道使用次数的最大平稳值。此外，图 (b) 中展示了 Q_1、Q_2 加权性能随返回道最大使用次数的变化，附件 1 的 59、附件 2 的 32 为对应的峰值点，且此时而二者对应的返回道使用次数及总调度时间均处于较低水平，因此可以将此时的仿真结果作为 PBS 缓冲区调度结果进行输出。

附件 1 的 59、附件 2 的 32 对应的出车序列仿真结果如表3所示。图中下方数字序列为车辆未经过 PBS 缓冲区的数据，上方车身序号为经过 PBS 缓冲区排序后的出车序列，可以看出，通过使用返回道及进车道，车辆经过了重排序，更符合后续 PBS 总装所要求的规律

4.4 问题一的结果

问题一附件 1、2 经过上述方法进行调度排序后的各项题目所要求优化目标得分结果（已乘系数）对比如表4所示。题目所要求的每一时刻的车辆所处位置将于附件中给出。如表4所示，附件 1 由于混动与柴油动力的占比约为 1：2，而附件 2 则约为 1：1，因此，相比于附件 1，附件 2 的求解结果显著增加了 Q_1 的数值，并且附件 2 所使用返回车道的次数也相对减小，因此最终的总评分也有了大幅提升。此外，对比附件 1、2 的 Q_3、Q_4 评价结果，减少返回车道的使用次数也会相应节省一定的缓冲区排序时间消耗。

表 4　问题一调度输出结果

得分	Q_1	Q_2	Q_3	Q_4	**总得分结果**
附件 1	-24.4	15.9	8.4	8.652	**9.4520**
附件 2	-8.4	18.6	13.8	9.081	**33.0810**

5. 问题二的分析与求解

5.1 问题二模型的分析与构建

相比于问题一，问题二不再要求在进车道入口处优先处理返车道出口车辆，同时不必在进车道出口侧严格遵循先到先卸载原则。这样便为车辆出车顺序的调整赋予了更高的自由度。同时，由于问题一中要求优先处理到达返车道出口车辆，相比于直接处理缓存区进入车辆会消耗更多的时间，进车道出口位置的接车决策同理。因此，在问题二中，若优先处理当前期望调度时间最短的车辆，同样会提高总的调度效率。

5.1.1 无 PBS 约束 6 的进车过程分析

如果去除 PBS 约束说明中第 6 条约束，则返回道 10 停车位有车时，不一定优先处理，需要和涂装车间的出车序列一起进行调度。对于缓存区的进车过程，为进一步提高调度效率，以车辆的期望调度时间作为评判依据，选择总等待与调度时间之和最短的车辆作为目标车辆。例如，若返车道 10 车位有车，但其移送时间大于移送位于进车序列首位置车辆的时间，则选择不移送返车道 10 车位车辆；再如，若返车道的 9 车位有车且在 2s 后移至 10 车位，而此时，返车道移送车辆的下一目标车道为 3 车道，而进车序列的首位置车辆的下一移送目标车道为 6 车道，则等待返回车到达并移入 3 车道所需时间为 14s，小于将进车序列首位置车辆移至 6 车道所需的 18s，因此，在当前时刻接车横移机可选择等待 2s 后将进入返车道 10 车位的车辆送入 3 车道。上述流程如图7问题二流程图所示。

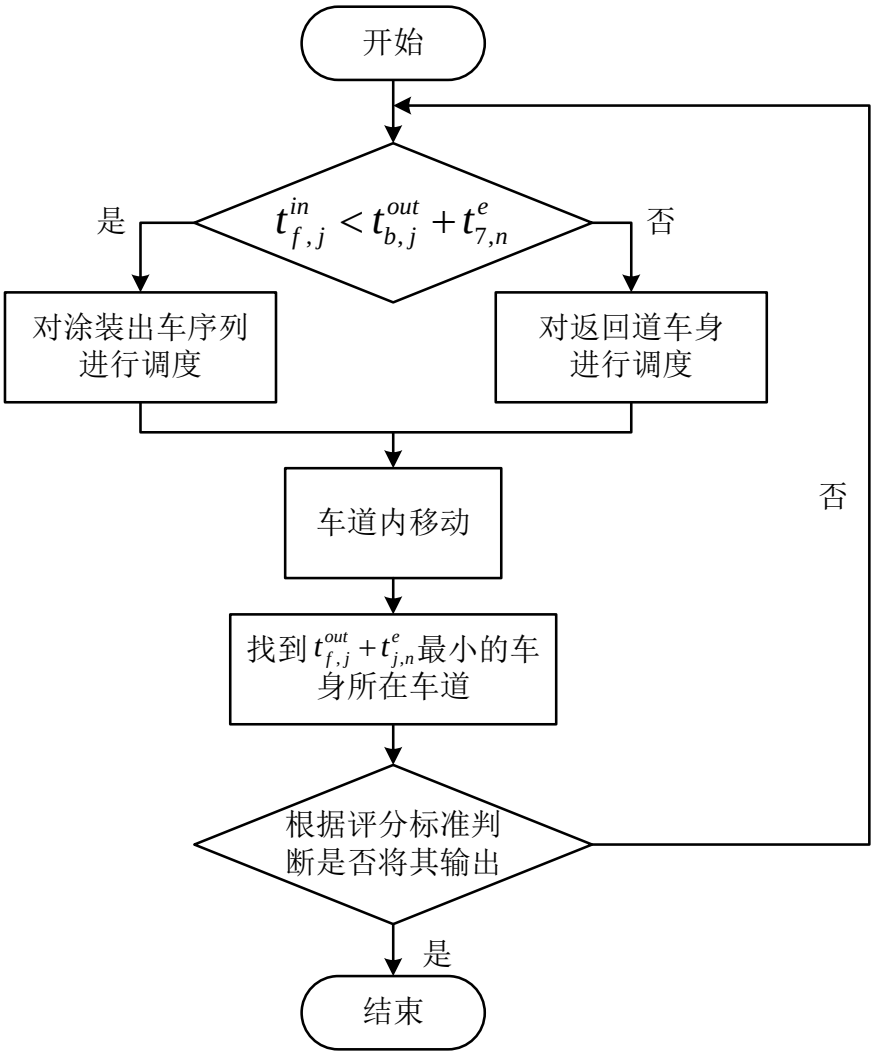


图 7　问题二流程图

具体地，设第 n 秒车道 j 最前面的车身到车道口的时间为 $t_{j,n}^{e}$，定义第 n 秒将车身由

横移机卸载到各进车道 10 停车位操作对应的决策变量 $V_{i',j,n}$

$$V_{i',j,n} = \begin{cases} 1, & \text{if } t^{in}_{f,j} \leq t^{out}_{b,j} + t^{e}_{7,n}, \forall i = 1, \ldots, N \\ 0, & \text{if } t^{in}_{f,j} > t^{out}_{b,j} + t^{e}_{7,n}, \forall i = 1, \ldots, N \end{cases}, \tag{16}$$

因此，在进车道入口位置，不断遵循上述最小运输时间的 MTT（Minimum Transportation Time）原则，将对应的最短等待与调度时间之和最短的车辆送入相应的进车道，车道的选择仍遵循问题一中的基于预计最短送车时间的启发式规则。问题二 PBS 缓存区进车调度整体算法如5 . 1 . 1节 Algorithm 3 所示。

Algorithm 3: 问题二入车过程

Input: 进车序列 $a_{i,j}$，车道状态 $x_{i,j,k}$，进车道进车过程所需时间 $t^{in}_{f,j}$，返回道出车过程所需时间 $t^{out}_{b,j}$，进车道当前车身数量 c^N_j

Output: 更新后的进车道车位状态 $x_{i,j,k}$，进车序列 $a_{i,j}$，进车道当前车身数量 c^N_j

Step A：检查送车横移机是否完成上一接车任务
if < 返回道出口位置非空 ($x_{i,7,10} = 1$)> **then**
    Step B：比较出车序列调度至车道 j 的时间 $t^{in}_{f,j}$，返回道第一辆车到 10 车位时间 $t^{e}_{7,n}$ 与到车道 j 的时间 $t^{out}_{b,j}$ 之和
    if <$t^{out}_{f,j} \leq t^{in}_{b,j} + t^{e}_{7,n}$ > **then**
        Step C：将进车序列 $a_{i,1}$ 第一辆车 $a_{1,1}$ 作为下一时刻出车
    else if <$t^{out}_{f,j} > t^{in}_{b,j} + t^{e}_{7,n}$> **then**
        Step D：将车道 j 最前面的车作为下一时刻出车
    end if
else if < 返回道出口位置为空 ($x_{i,7,10} = 0$)> **then**
    Step E：选择进车序列 $a_{i,1}$ 第一辆车 $a_{1,1}$ 作为下一时刻出车
end if
Step F：根据进车道车数 c^n_j, 进车道进车过程所需时间 $t^{in}_{f,j}$，返回道出车过程所需时间 $t^{out}_{b,j}$，基于式（2），计算对应目标进车道 j^{tar}
while < 当前选定进车道 j^{tat} 有车 ($x_{i,j^{tar},10} = 1$)> **do**
    Step G：选择估计送车时间第二小的进车道作为目标车道 j^{tar}，以此类推，直至满足目标进车道 j^{tar} 的 10 停车位无车
end while
Step H：基于 $t^{in}_{f,j}$, $t^{out}_{b,j}$ 更新 $x_{i,j,k}$, c^N_j, $a_{i,j}$

5.1.2 无 PBS 约束 7 的出车过程分析

若忽略原 PBS 约束说明中的约束 7，则进车道的 1 车位有车时，并不一定根据其最先到达时间优先处理。例如，若进车道 1 的 10 车位存在一辆车，同时另一辆车处于进车道 4 的车位 9 并且还有 5s 移动至进车道 4 出口处，则送车横移机对进车道 1 的车进行调度需要消耗 18 秒时间，而等待进车道 4 的车辆到达并进行调度仅需消耗 5s 时间。

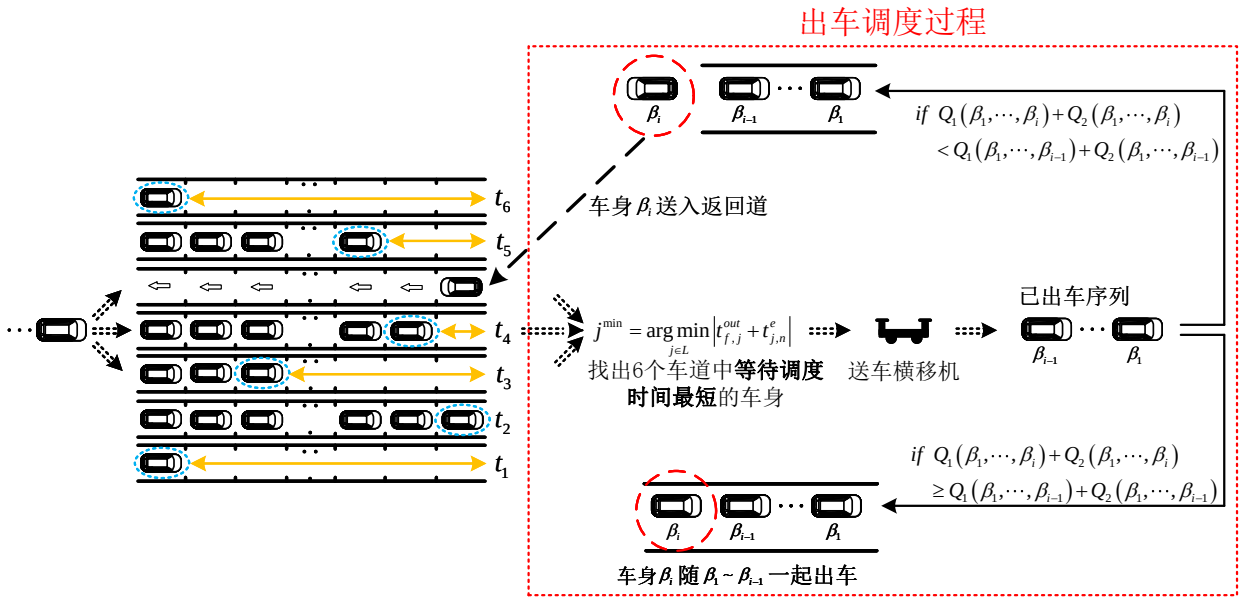


图 8　问题二动态出车过程流程图

因此，对于 PBS 缓存区进车道出口位置的出车调度，同样需要遵循 MTT 准则，即对 $t^{out}_{f,j}+t^{e}_{j}$ 最小的车身所在车道 $j^{min}, j\in L$ 的车辆进行调度，即

$$j^{min}=\arg\min_{j\in L}\left|t^{out}_{f,j}+t^{e}_{j,n}\right| \tag{17}$$

基于此，定义出车或返回决策变量

$$P_j=\begin{cases}1, & \text{将车道 } j \text{ 的车身出车}\\ 0, & \text{将车道 } j \text{ 的车身送回返回道}\end{cases},\text{if } j=j^{min}. \tag{18}$$

上述 PBS 缓存区出车调度整体算法如5.1.2节 Algorithm 4 所示。

Algorithm 4: 问题二出车过程

Input: 已出车序列 $b_{i,j}$，车道状态 $x_{i,j,k}$，进车道出车过程所需时间 $t_{f,j}^{out}$，返回道进车过程所需时间 $t_{b,j}^{in}$，进车道当前车身数量 c_j^N

Output: 进车道车位状态 $x_{i,j,k}$，进车道当前车身数量 c_j^N

Step A：检查送车横移机是否完成上一接车任务

if < 至少有一个进车道非空 ($\sum_{k=1}^{M_\beta} > 0$)> **then**

    Step B：计算所有进车道第一辆车到 1 停车位时间 $t_{7,n}^e$ 与到车道 j 时间 $t_{f,j}^{out}$ 之和，选择 $j^{min} = \arg\min_{j \in L} |t_{f,j}^{out} + t_{j,n}^e|$ 作为目标车道，并将对应车辆作为当前接待车辆，横移车等待至该车辆转运至进车道出口

else if < 所有进车道都为空 ($\sum_{k=1}^{M_\beta}$=0)> **then**

    Step B：接车横移机设置为空闲状态

end if

Step C：基于一出车序列 $b_{i,1}$，调用子程序计算当前 Q_1, Q_2 加权得分值 f_1

Step D：假设当前接车车辆已出车，调用子程序计算当前 Q_1, Q_2 加权得分值 f_2

if <$f_1 > f_2$> **then**

    Step E：将当前接车辆送入返回道

else

    Step E：将当前接车辆加入出车序列

end if

Step F：基于 $t_{f,j}^{out}$, $t_{b,j}^{in}$ 更新 $x_{i,j,k}$, c_j^N

5.1.3 问题二的 PBS 缓冲区进出车调度模型

基于上述分析，可以构建问题二的优化模型如下所示：

其中，目标函数为

$$(P4): \min_{V_{i',j}, P_j} \sum_{i=1}^{4} w_i * (100 - Q_i),$$

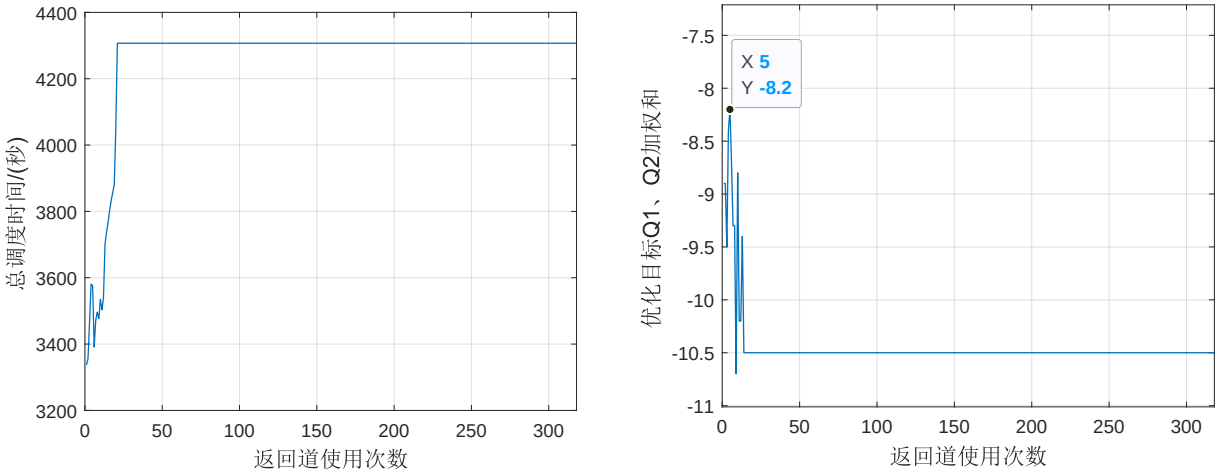
约束条件为

$$
\text{s.t.}\begin{cases}
\sum_{k=1}^{M_\beta} x_{i,j,k} \le M_\beta, & \forall i=1,\dots,N, \forall j=1,\dots,M_\alpha \\
T_i = t_{f,j}^{in} + 9t_m + t_{f,j}^{out} + t_i^w & \\
\quad +N_{b,i}(t_{f,j}^{in} + 9t_m + t_{f,j}^{out}), & i=1,\dots,N \\
V_{i,j}=1, & \text{if } t_{f,j}^{in} \le t_{b,j}^{out} + t_{7,n}^{e}, \forall i=1,\dots,N, j \in L \\
V_{i',j}=1, & \text{if } t_{f,j}^{in} > t_{b,j}^{out} + t_{7,n}^{e}, \forall i'=1,\dots,N, j \in L \\
P_j, & \text{find } \arg\min_{j\in L} |t_{f,j}^{out} + t_j^e| \\
T = t_e - t_s & \\
Q_1 = \sum_{i=1}^{N_m-1} \delta_{1,i}, & i=1,\dots,N_m-1 \\
Q_2 = \sum_{i=1}^{N_n} \delta_{2,i} & i=1,\dots,N_n \\
Q_3 = \sum_{i=1}^{N} N_{b,i} & \\
Q_4 = 0.01 \times (T - 9N - 8t_m) & \\
\sum_{i=1}^{4} w_i = 1, & i=1,\dots,4 \\
w_i = 0.4, 0.3, 0.2, 0.1, & i=1,\dots,4 \\
t_{f,j}^{in} = 18,12,6,0,12,18, & j=1,\dots,M_\alpha-1 \\
t_{f,j}^{out} = 18,12,6,0,12,18, & j=1,\dots,M_\alpha-1 \\
t_{b,j}^{in} = 24,18,12,6,12,18, & j=1,\dots,M_\alpha-1 \\
t_{b,j}^{out} = 24,18,12,6,12,18, & j=1,\dots,M_\alpha-1 \\
t_m = 9 & \\
x_{i,j,k} \in \{0,1\} & \forall i=1,\dots,N, \forall j=1,\dots,M_\alpha, k=1,\dots,M_\beta \\
\delta_{1,i} \in \{0,1\} & \forall i=1,\dots,N_m-1 \\
\delta_{2,i} \in \{0,1\} & \forall i=1,\dots,N_n
\end{cases} \tag{19}
$$

5.2 模型的求解

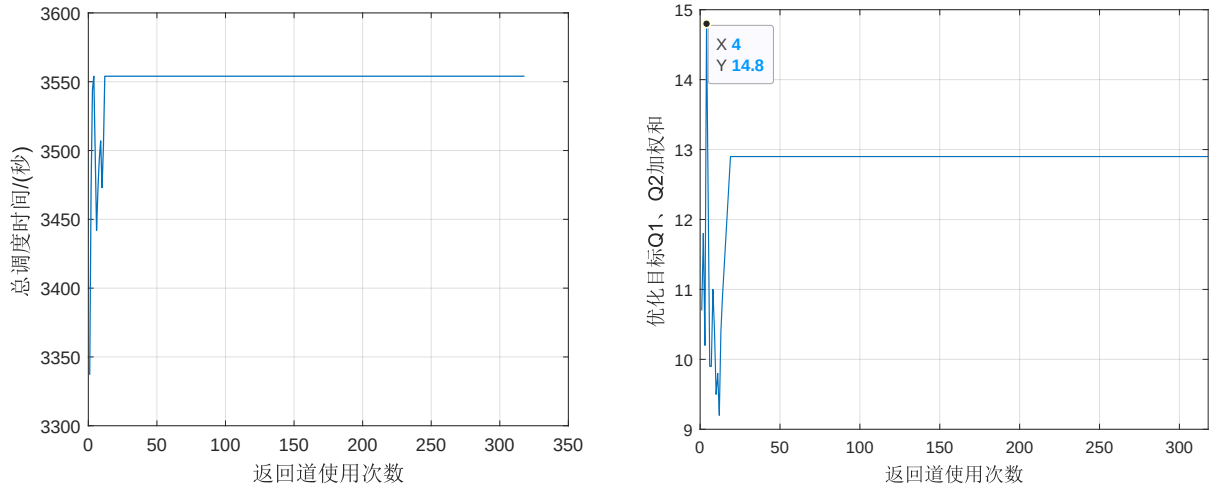
基于 5.1.1 与 5.1.2 所述基于改进启发式规则的缓存区进车调度和基于贪心法的出车调度算法，同样利用 MATLAB 软件对问题二的附件一、二所给数据进行求解，且仿真参数仍与问题一保持一致。

仿真输出的部分最终出车序列结果如表**??**所示，题目所要求的每一时刻的车辆所处位置将于附件中给出。Q_1、Q_2 加权性能随返回道使用次数的变化如图9、图10所示。根据图8与图9的 (a) 可知，对比问题一中总调度时间随返回道使用次数的变化趋势的峰值，问题二所设计的进出车机制能够大幅降低总调度时间。此外，根据图9与图10的 (b) 项，该机制对出车序列的整体优化性能分别在返回道使用次数为 5、4 处取得峰值，即，该机制倾向于更少的使用返回道，从而节约效率、更快地将出车排序任务交给进车道出口处。此外，返回道使用次数的减少以及总调度时间的降低也会进一步提高系统的加权评价指标数值，



(a) 总调度时间　　(b) 优化目标 1、2 加权值

图 9　问题二附件 1 总调度时间 (a) 和优化目标 1、2 加权值 (b) 随返回道使用次数的变化



(a) 总调度时间　　(b) 优化目标 1、2 加权值

图 10　问题二附件 2 总调度时间 (a) 和优化目标 1、2 加权值 (b) 随返回道使用次数的变化

提升总体评价得分。

5.3 问题二的结果

问题二中，附件 1、2 经过上述方法进行调度排序后的各项题目所要求优化目标得分结果（已乘系数）对比如表6所示。题目所要求的每一时刻的车辆所处位置将与问题一一起于附件中给出。如表6所示，与问题一类似，由于附件 1 中汽车的混动与柴油动力占比约为 1：2、附件 2 约为 1：1，因此相比于附件 1，附件 2 的求解结果同样显著增加了 Q_1 的数值，同时附件 2 所使用返回车道的次数也相对减小，因此最终的总评分也有了大幅提升。对比附件 1、2 的 Q_3、Q_4 评价结果，同样可以得出减少返回车道的使用次数也会相应节省部分时间消耗的结论。

表 5　问题二 PBS 缓冲区出车序列结果

	PBS 缓冲区输出序列
附件 1	1 3 4 6 7 8 13 9 12 16 17 11 18 20 19 15 21 22 24 23 25 26 28 29 27 2 30 32 31 5 34 35 36 10 33 37 39 40 38 41 43 42 14 45 46 48 44 47 49 50 52 53 51 55 57 54 58 60 59 56 62 63 64 66 67 65 61 70 68 69 71 73 75 72 74 76 77 79 80 78 82 83 81 84 85 87 89 86 88 90 91 93 94 92 96 97 95 98 99 101 103 100 102 104 105 107 108 106 110 111 109 112 113 115 117 114 116 118 119 121 122 120 124 125 123 126 127 129 131 128 130 132 133 135 136 134 138 139 137 140 141 143 145 142 144 146 147 149 150 148 152 153 151 154 155 157 159 156 158 160 161 163 164 162 166 167 165 168 169 171 173 170 172 174 175 177 178 176 180 181 179 182 183 185 187 184 186 188 189 191 192 190 194 195 193 196 197 199 201 198 200 202 203 205 206 204 208 209 207 210 211 213 215 212 214 216 217 219 220 218 222 223 221 224 225 227 229 226 228 230 231 233 234 232 236 237 235 238 239 241 243 240 242 244 245 247 248 246 250 251 249 252 253 255 257 254 256 258 259 261 262 260 264 265 263 266 267 269 271 268 270 272 273 275 276 274 278 279 277 280 281 283 285 282 284 286 287 289 290 288 292 293 291 294 295 297 299 296 298 300 301 303 304 302 306 307 305 308 309 311 313 310 312 314 315 317 318 316
附件 2	1 3 4 5 6 7 8 13 9 12 16 17 11 18 20 19 15 21 22 24 23 25 26 28 29 27 2 30 32 31 10 34 35 36 14 33 37 39 40 38 41 44 46 43 42 47 50 48 45 51 52 53 49 54 56 57 58 55 60 61 62 63 64 65 66 67 59 68 71 70 69 72 74 73 76 77 78 75 79 80 82 81 84 83 85 86 88 87 89 91 92 90 94 95 93 96 98 97 99 100 102 101 103 105 106 104 108 109 107 110 112 111 113 114 116 115 117 119 120 118 122 123 121 124 126 125 127 128 130 129 131 133 134 132 136 137 135 138 140 139 141 142 144 143 145 147 148 146 150 151 149 152 154 153 155 156 158 157 159 161 162 160 164 165 163 166 168 167 169 170 172 171 173 175 176 174 178 179 177 180 182 181 183 184 186 185 187 189 190 188 192 193 191 194 196 195 197 198 200 199 201 203 204 202 206 207 205 208 210 209 211 212 214 213 215 217 218 216 220 221 219 222 224 223 225 226 228 227 229 231 232 230 234 235 233 236 238 237 239 240 242 241 243 245 246 244 248 249 247 250 252 251 253 254 256 255 257 259 260 258 262 263 261 264 266 265 267 268 270 269 271 273 274 272 276 277 275 278 280 279 281 282 284 283 285 287 288 286 290 291 289 292 294 293 295 296 298 297 299 301 302 300 304 305 303 306 308 307 309 310 312 311 313 315 316 314 318 317

对比表4与表6，问题一与问题二的 Q_1、Q_2 数值较为接近但问题二整体得分有了显著提升，该提升主要来自于返回道使用次数的减少。尽管减少了返回道的使用次数，但由于进车道入口及出口处不再有严格的优先调度条件，使得调度机制拥有了更高的自由度，同时一定程度上节省了总调度时间，从而在更少使用返回道的情况下仍能够维持 Q_1、Q_2 的优化效果。

表 6　问题二调度输出结果

得分	Q_1	Q_2	Q_3	Q_4	**总得分结果**
附件 1	-25	16.8	19.2	9.491	**20.8910**
附件 2	-9.2	24	19	9.522	**43.7220**

6. 模型的对比与分析

综合本文叙述，本文集中于对于缓存区进出车调度，根据每个时刻当前的参数状态，来不断更新缓存区进车与出车决策变量，同时根据决策结果对其进行更新。整体思路及算法步骤如技术路线图11所示

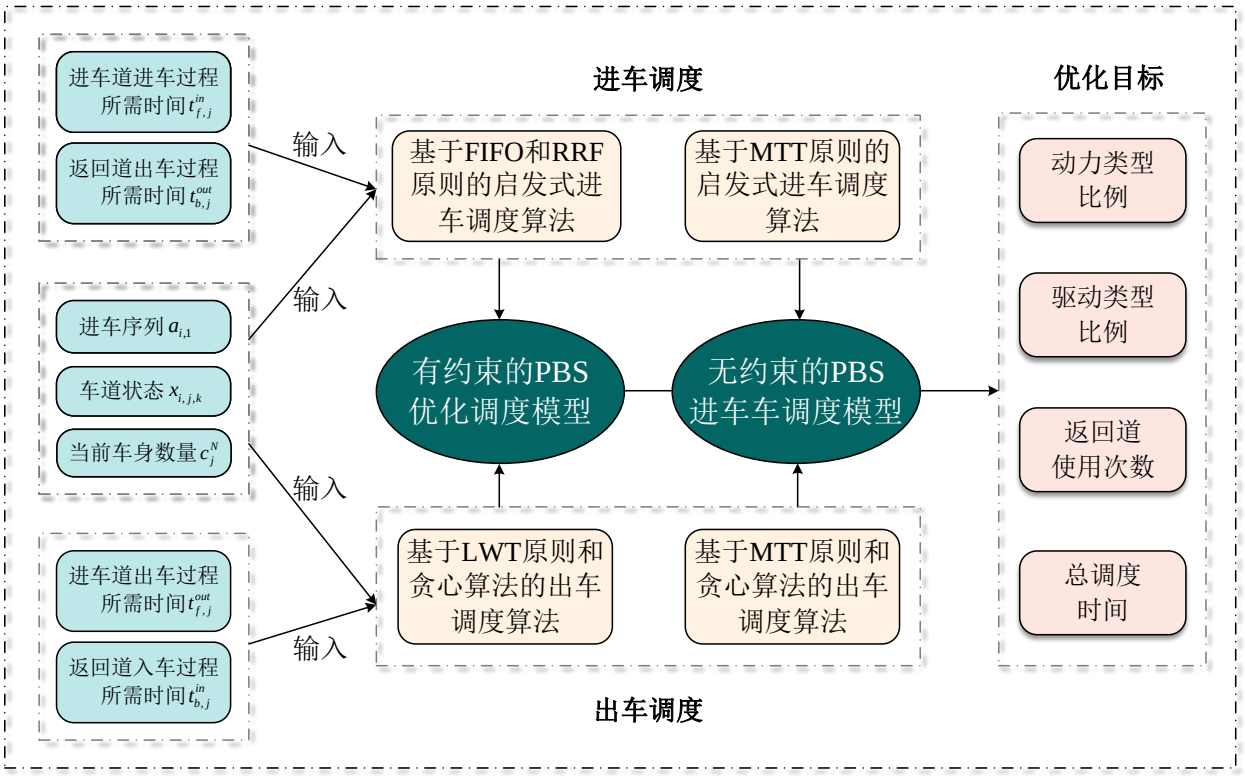


图 11　技术路线图

针对于上述缓冲区车辆重排序问题，文献 [4] 提出了一种泛化模型，可以根据不同的限定条件和参数来将模型具体化。同时，文献 [3] 设计了一种基于遗传算法对此类混合整数规划问题进行求解的方法。基于此，为了从多方面验证本文所提 PBS 缓冲区调度机制的有效性，本文在问题二的模型基础上，进一步忽视进车道所带来的影响，即在优化问题 $P4$) 中忽略变量 $V_{i',j}$ 及其所涉及约束，随后利用遗传算法对该简化问题进行求解。其中适应度函数设置为 $P4$) 的目标函数，采用二进制编码方式，设置种群大小为 200，进化代数为 1000。此外，对于决策变量，设置新的 $\chi_{i,j}$ 代表入车决策变量，代表是否将车辆 i 放置于进车道 j，设置 ϖ 代表是否将车辆 i 出车于出车序列位置 j。因此，除 $P4$) 原有约束外，

额外有如下非线性约束：

$$\begin{cases} \sum_{l=1}^{M_\alpha-1} \chi_{i,l} = 1, & \forall i = 1,\ldots,N \\ \sum_{i=1}^{N} \chi_{i,l} \le 10, & \forall l = 1,\ldots,M_\alpha - 1 \\ \sum_{i=1}^{N} \varpi_{i,j} = 1, & \forall i = 1,\ldots,N \\ \sum_{j=1}^{N} \varpi_{i,j} = 1, & \forall j = 1,\ldots,N \\ \sum_{i=1}^{N} \varpi_{i,j} \cdot i - \sum_{i=1}^{N} \varpi_{n,j} \cdot i & \\ \quad \le C \cdot (2 - \chi_{i,l} - \chi_{n,l}), & \forall i = 1,\ldots,N; \forall n = 1,\ldots,N; \forall l = 1,\ldots,M_\alpha - 1 \end{cases} \tag{20}$$

其中约束 1 表示每个车辆只能放置于 1 个进车道，约束 2 表示每个进车道最多只能同时存在 10 个车辆；约束 3、约束 4 表明每个车辆只能在最终出车序列中占有一个位置，以及最终出车序列的每个位置只能有一个车辆；约束 5 表示若两有 $i < j$ 关系的车辆分配至了同一进车道，则最终输出出车序列统一保证 i 在 j 之后。综上，变量 $\chi_{i,j}$ 为一 $(M_\alpha - 1) \times N$ 大小的 0-1 矩阵、$\{\varpi\}$ 为一 $N \times N$ 大小的 0-1 矩阵。随着车辆数量的增加，问题规模是呈平方级增长的。

利用 MATLAB 中的遗传算法工具箱，写入适应度函数及非线性约束。当设置 N=5 时，可以在多项式时间内求得最优解，其遗传算法的适应度的迭代过程如图12所示

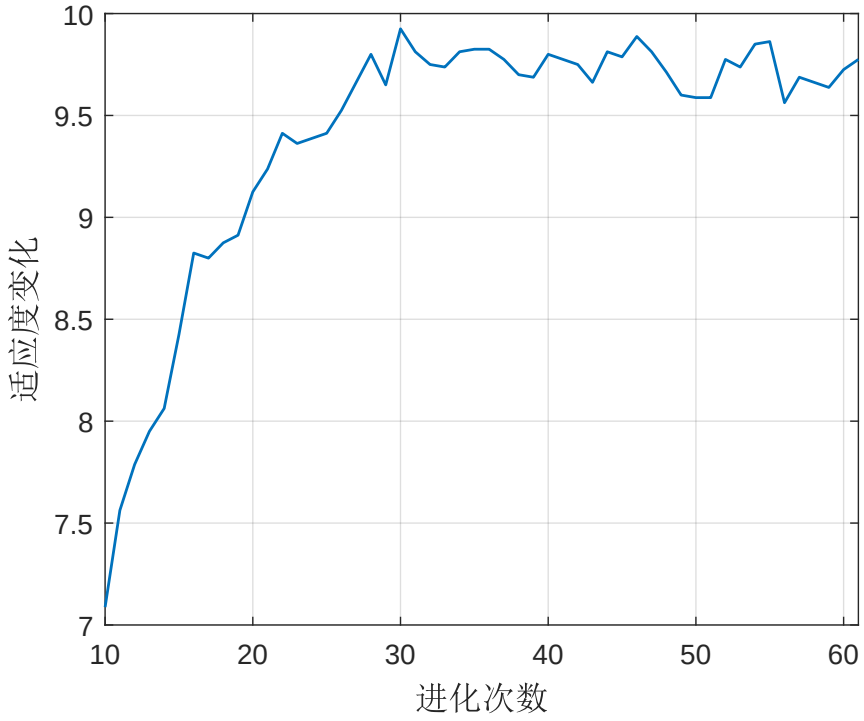


图 12　遗传算法适应度迭代结果图

然而，当变量数进一步增加，当 $N = 20$ 时，遗传算法在迭代过程中出现了求解结果不满足约束条件的情况；当 N 进一步增加至 $N = 100$ 时，不仅约束不再满足，遗传算法

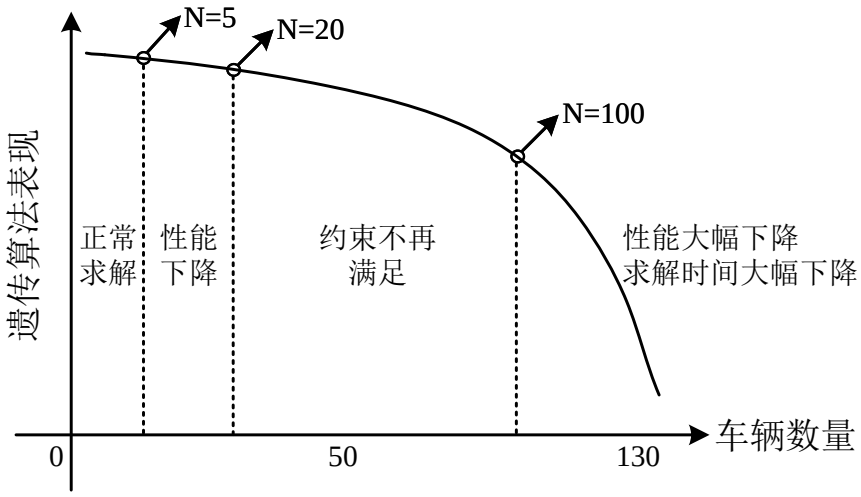


图 13　遗传算法表现随车辆数增加的结果图

所求解的适应度结果甚至不如原始序列。此外，当 $N = 100$ 时，遗传算法求解时间达到了小时量级。遗传算法随车辆数 N 增长的表现变化如图13所示

与之相对，本文所提 PBS 缓冲区进出车调度机制的平均调度时间统计结果见表7所示，不难看出，尽管启发式规则及贪心法取局部最优的方式不可避免的会损失部分性能，然而在求解效率上具有更好的表现。

表 7　本文所提缓冲区进出车调度时间统计

	PBS 缓存区进车调度总计算时间	PBS 缓存区出车调度总计算时间	寻找最优使用返回道次数上限所用时间 (采用二分法)
问题二，$N = 318$	**0.623s**	**1.202s**	**36.053s**

7. 参考文献

[1] Zohali H, Reformulation, linearization, and a hybrid iterated local search algorithm for economic lot-sizing and sequencing in hybrid flow shop problems. Computers & Operations Research, 2019, 104:127-138.

[2] Shen Z, Tang Q, Huang T. Dynamic Production Scheduling Modeling and Multi-objective Optimization for Automobile Mixed-Model Production. Communications in Computer and Information Science, 2018, 924: 25-33.

[3] Jalilvand-Nejad A, Fattahi P. A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem. Journal of Intelligent Manufacturing, 2015, 26(6):1085-1098.

[4] Boysen N, Zenker M. A decomposition approach for the car resequencing problem with selectivity banks. Computers & Operations Research, 2013, 40(1):98-108.

[5] Yavuz, Mesut. Iterated beam search for the combined car sequencing and level scheduling problem. International Journal of Production Research, 2013, 51(12):3698-3718.

[6] Soleimani H, Kannan G. A hybrid particle swarm optimization and genetic algorithm for closed-loop supply chain network design in large-scale networks[J]. Applied Mathematical Modelling, 2015, 39(14): 3990-4012.

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

[8] Akrami B, Karimi B, Hosseini S M M. Two metaheuristic methods for the common cycle economic lot sizing and scheduling in flexible flow shops with limited intermediate buffers: The finite horizon case[J]. Applied Mathematics and computation, 2006, 183(1): 634-645.

[9] 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.

[10] 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.

[11] 杜旭浩. 基于智能优化算法的汽车混流装配线排序问题研究 [D]. 燕山大学, 2013.

附录 A 问题一 PBS 优化调度模型程序代码

1.1 PBS 调度主程序-main.m

```
result=zeros(319,4290);
result(2:319,1)=1:318;
result(1,2:4290)=0:4288;
N_of = 0;
lane_in1 = 0;
lane_in2 = 0;
car_in_now1 = -1;
car_in_now2 = -1;
lan_fig = -ones(6,9*10,2);
t_in1 = [18, 12, 6, 0, 12, 18];
t_in2 = [24, 18, 12, 6, 12, 18];
t_out1 = [18, 12, 6, 0, 12, 18];
t_out2 = [24, 18, 12, 6, 12, 18];
t_out = mean(t_out1);
lane_in = 0;
lane_out = 0;

load type
load color
color = color';
type = type';
x_in = 1:318;%输入车
car_seq_in = x_in;
end_flg = 0;
lane = zeros(1,6);
ret_fig = -ones(1,9*10,2);
N_fin = 0;
car_in_now = -1;
flgi = 0;
flgo = 0;
L = 0;
inflg = 0;
x_out = [];%输出车
N = 20000;
epoch = 0;
feq = 58;
```

```
while epoch < N
    if (numel(car_seq_in) == 0 && sum(ret_fig(1,1:90,1)) == -90 &&
        sum(sum(lan_fig(:,1:90,1))) == -540) || numel(x_out)>=318
        break;
    end
    epoch = epoch+1;
    ret_fig = ret_renew_final(ret_fig);
    lan_fig = renew_final(lan_fig,flgo);
    %% 进车过程
    if flgi == 0
        % N_fin=1则说明上次从返回道拉车，基于上次的lane_in进行更新
        %（lan_fig都要更新，ret_fig仅在N_fin=1更新）
        if N_fin == 1
            if lane_in1 ~= 0
                lan_fig(lane_in1,1,1) = car_in_now1;
                lan_fig(lane_in1,1,2) = 0;
                ret_fig(1,1,1) = -1;
            end
        else
            if lane_in2 ~= 0
                if end_flg ~= 1
                    lan_fig(lane_in2,1,1) = car_in_now2;
                end
                if car_in_now2 == 318
                    end_flg = 1;
                end
                lan_fig(lane_in2,1,2) = 0;
            end
        end
        if ret_fig(1,1,1) ~= -1
            N_fin = 1;
            % 更新本次送入车道lane_in
            [lane_in1,lane] =
                Car_in(1,lane,t_in1,t_in2,t_out,lan_fig,N_fin);
            flgi = t_in2(lane_in1);
            car_in_now1 = ret_fig(1,1,1);
        else
            if numel(car_seq_in) == 0
            else
```

```
74                N_fin = 0;
75                car_in_now2 = car_seq_in(1);
76                car_seq_in(1) = [];
77                [lane_in2,lane] =
                     Car_in(1,lane,t_in1,t_in2,t_out,lan_fig,N_fin);
78                flgi = t_in1(lane_in2);
79             end
80          end
81       else
82          flgi = flgi - 1;
83          if N_fin == 1
84             result(car_in_now1+1,epoch) = 1;
85          else
86             result(car_in_now2+1,epoch) = 1;
87          end
88       end
89       %% 出车过程
90       if flgo == 0
91          if N_of == 1
92             ret_fig(1,90,1) = car_out_now;
93          end
94          if lane_out ~= 0
95             lan_fig(lane_out,90,1) = -1;
96             lan_fig(lane_out,90,2) = -1;
97             lane(lane_out) = lane(lane_out)-1;
98          end
99          x_seq = x_out;
100         [car_out_now,lane_out] = Renew_lan_out(lan_fig); % 决定接哪条道的车
101         if car_out_now == 0
102         else
103            [if_f,x_out,feq] =
                  Car_out(x_seq,type,color,car_out_now,ret_fig,feq);
104            if if_f == 1
105               flgo = t_out1(lane_out);
106               N_of = 0;
107            else
108               flgo = t_out2(lane_out);
109               N_of = 1;
110            end
```

```
        end
    else
        result(car_out_now+1,epoch) = 2;
        flgo = flgo -1;
    end
    %% 更新过程
    [lan_i,lan_j,lan_car]=Car_lan(lan_fig);
    [ret_i,ret_j,ret_car]=Car_ret(ret_fig);
    [m,~]=size(lan_i);
    for i=1:m
        if result(lan_car(i)+1,epoch) == 0
            result(lan_car(i)+1,epoch)=...
            str2num([num2str(lan_i(i)),num2str(lan_j(i))]);
        end
    end
    [n,~]=size(ret_j);
    for i=1:n
        if result(ret_car(i)+1,epoch) == 0
            result(ret_car(i)+1,epoch)=str2num(['7',...
            num2str(ret_j(i))]);
        end
    end
    for i = 1:numel(x_out)
        if result(x_out(i)+1,epoch) == 0
            result(x_out(i)+1,epoch) = 3;
        end
    end
    for i = 1:numel(car_seq_in)
        if result(car_seq_in(i)+1,epoch) == 0
            result(car_seq_in(i)+1,epoch) = 0;
        end
    end
end
for kk = 1:flgo
    result(car_out_now+1,epoch+kk) = 2;
end
result(car_out_now+1,epoch+flgo+1) = 3;
```

1.2 进车过程子程序-Car_in.m

```
function [L,lane] = Car_in(car_seq,lane,t_in1,t_in2,t_out,lan_fig,inflg)
   ti = zeros(1,numel(lane));
   for i = 1:numel(lane)
      if inflg == 1 %
         ti(i) = lane(i)*t_out+t_in1(i);
      else
         ti(i) = lane(i)*t_out+t_in2(i);
      end
   end
   [~,L] = min(ti);
   while sum(lan_fig(L,1:9,1)) ~= -9 && min(ti) ~= inf
      ti(L) = inf;
      [~,L] = min(ti);
   end
   lane(L) = lane(L)+1;
end
```

1.3 出车过程子程序-Car_out.m

```
function [L,x_out,feq] = Car_out(x_seq,type,color,x,ret_fig,feq)
   x_new = [x_seq x];
   if(0.4*fun_obj11(x_new,type)+0.3*fun_obj22(x_new,color) >
      0.4*fun_obj11(x_seq,type)+0.3*fun_obj22(x_seq,color)+0.2*1) &&
      sum(ret_fig(1,82:90,1)) == -9 && feq > 0
      L = 0;
      x_out = x_seq;
      feq = feq-1;
   else
      L = 1;
      x_out = x_new;
   end
end
```

1.4 出车判断子程序-Renew_lan_out.m

```
function [Lo,idx] = Renew_lan_out(lan_fig)
   maxva = 0;
```

```
    maxid = 0;
    idx = 0;
        for i = 1:6
            if lan_fig(i,90,1) ~= -1
                if lan_fig(i,90,2) > maxva
                    maxva = lan_fig(i,90,2);
                    maxid = lan_fig(i,90,1);
                    idx = i;
                end
            end
        end
        Lo = maxid;
    end
```

1.5 进车道当前状态更新子程序-renew_final.m

```
function lan_fig = renew_final(lan_fig,flgo)
    for i = 1:6
        if lan_fig(i,90,1) == -1
            for j = 89:-1:81
                if lan_fig(i,j,1) ~=-1
                    lan_fig(i,j+1,1) = lan_fig(i,j,1);
                    lan_fig(i,j+1,2) = lan_fig(i,j,2)+1;
                    lan_fig(i,j,1) = -1;
                    lan_fig(i,j,2) = -1;
                end
            end
        end
    end
    for i = 1:6
        for j = 80:-1:1
            if lan_fig(i,j,1) ~= -1 &&
                sum(lan_fig(i,(1+floor(j/9))*9:(1+floor(j/9))*9+8,1)) == -9
                lan_fig(i,j+1,1) = lan_fig(i,j,1);
                lan_fig(i,j+1,2) = lan_fig(i,j,2)+1;
                lan_fig(i,j,1) = -1;
                lan_fig(i,j,2) = -1;
            end
        end
    end
```

```
   end
end
```

1.6 返回道当前状态更新子程序-ret_renew_final.m

```
function ret_fig = ret_renew_final(ret_fig)
   if ret_fig(1,1,1) == -1
      for j = 2:9
         if ret_fig(1,j,1) ~= -1
            ret_fig(1,j-1,1) = ret_fig(1,j,1);
            ret_fig(1,j,1) = -1;
         end
      end
   end
   for i = 10: 90
      if ret_fig(1,i,1) ~= -1 &&
         sum(ret_fig(1,(ceil(i/9)-1)*9:-1:(ceil(i/9)-1)*9-8,1)) == -9
         ret_fig(1,i-1,1) = ret_fig(1,i,1);
         ret_fig(1,i,1) = -1;
      end
   end
end
```

1.7 进车道结果输出子程序-Car_lan.m

```
function [lan_i,lan_j,lan_car]=Car_lan(lan_fig)
   car=lan_fig;
   [lan_i,lan_j]=find(car(:,:,1)~=-1);
   if size(lan_i,1) == 1
      lan_i=lan_i';
      lan_j=lan_j';
   end
   if numel(lan_i)==0
      lan_i=[];lan_j=[];lan_car=[];
      return
   end
   [m,~]=size(lan_i);
   lan_car=[];
   for i=1:m
```

```
        lan_car=[lan_car;car(lan_i(i),lan_j(i),1)];
    end
    lan_j=10-floor((lan_j-1)/9);
end
```

1.8 返回道结果输出子程序-Car_ret.m

```
function [ret_i,ret_j,ret_car]=Car_ret(ret_fig)
    car=ret_fig;
    [ret_i,ret_j]=find(car(:,:,1)~=-1);
    if size(ret_i,1) == 1
        ret_i=ret_i';
        ret_j=ret_j';
    end
    if numel(ret_i)==0
        ret_i=[];ret_j=[];ret_car=[];
        return
    end
    [m,~]=size(ret_i);
    ret_car=[];
    for i=1:m
        ret_car=[ret_car;car(ret_i(i),ret_j(i),1)];
    end
    ret_j=10-floor((ret_j-1)/9);
end
```

1.9 计算 Q_1 和 Q_2 加权和子程序-count_c.m

```
function score = count_c(x)
    sum2 = 0;
    sum4 = 0;
    for i = 1:numel(x)
        if(x(i) == 2)
            sum2 = sum2+1;
        else
            sum4 = sum4+1;
        end
    end
    if sum2 == sum4
```

```
        score = 0;
    else
        score = 1;
    end
end
```

1.10 优化目标 Q_1 得分子程序-fun_obj11.m

```
    function f1 = fun_obj11(x,type)
    if numel(x) == 0
        f1 = 0;
    end
    m = numel(x);
    t1 = zeros(m,1);
    for i = 1:m
        t1(i) = type(x(i));
    end
    col=find(t1==1);
    res=0;
    for i=1:length(col)-1
        if col(i+1)-col(i)-1~=2
            res=res+1;
        end
    end
    f1 = res;
end
```

1.11 优化目标 Q_2 得分子程序-fun_obj22.m

```
    function f2 = fun_obj22(x,color)
    score = 0;
    m = numel(x);
    if m == 0
        f2 = 0;
    else
    flag = 1;
    if color(x(1)) == 4
        for i = 1:m-1
            if color(x(i))==2 && color(x(i+1)) == 4
```

```
            score = score+count_c(color(x(flag:i)));
            flag = i+1;
        end
        if i == m-1
            score = score+count_c(color(x(flag:end)));
        end
    end
  else
      for i = 1:m-1
        if color(x(i))==4 && color(x(i+1)) == 2
            score = score+count_c(color(x(flag:i)));
            flag = i+1;
        end
        if i == m-1
            score = score+count_c(color(x(flag:end)));
        end
      end
  end
  f2 = score;
  end
end
```

附录 B 问题二 PBS 优化调度模型程序代码

2.1 PBS 主程序-main22.m

```
  clear,clc
  value_vec1 = [];
  value_vec2 = [];
  epoch_vec = [];

  for ee = 4:4
  result=zeros(319,10000);
  result(2:319,1)=1:318;
  result(1,2:10000)=0:9998;
  N_of = 0;
  lane_in1 = 0;
  lane_in2 = 0;
  car_in_now1 = -1;
```

```
14    car_in_now2 = -1;
15
16    lan_fig = -ones(6,9*10,2);
17
18    t_in1 = [18, 12, 6, 0, 12, 18];
19    t_in2 = [24, 18, 12, 6, 12, 18];
20    t_out1 = [18, 12, 6, 0, 12, 18];
21    t_out2 = [24, 18, 12, 6, 12, 18];
22    t_out = mean(t_out1);
23    lane_in = 0;
24    lane_out = 0;
25
26    load type
27    load color
28    color = color';
29    type = type';
30    x_in = 1:318;%输入车
31    car_seq_in = x_in;
32    end_flg = 0;
33
34    lane = zeros(1,6);
35    ret_fig = -ones(1,9*10,2);
36    N_fin = 0;
37    car_in_now = -1;
38
39    flgi = 0;
40    flgo = 0;
41    L = 0;
42    inflg = 0;
43    x_out = [];%输出车
44    N = 20000;
45
46    epoch = 0;
47    % feq = 24;
48    feq = ee;
49    while epoch < N
50       %% 终止判定
51       if (numel(car_seq_in) == 0 && sum(ret_fig(1,1:90,1)) == -90 &&
          sum(sum(lan_fig(:,1:90,1))) == -540) || numel(x_out)>=318
```

```
        break;
    end

    epoch = epoch+1;
    ret_fig = ret_renew_final(ret_fig);
    lan_fig = renew_final(lan_fig,flgo);
    %% 进车过程
    if flgi == 0
        if N_fin == 1
            if lane_in1 ~= 0 && car_in_now1 ~= -1
                lan_fig(lane_in1,1,1) = car_in_now1;
                lan_fig(lane_in1,1,2) = 0;
                if ret_fig(1,1,1) ~= -1
                    ret_fig(1,1,1) = -1;
                    car_in_now1 = -1;
                end
            end
        else
            if lane_in2 ~= 0
                if end_flg ~= 1
                    lan_fig(lane_in2,1,1) = car_in_now2;
                    lan_fig(lane_in2,1,2) = 0;
                end
                if car_in_now2 == 318
                    end_flg = 1;
                end
            end
        end
        [inflg,car_idx] = in_decide2(lan_fig,ret_fig,car_seq_in,...
                                t_in1,t_in2,lane,t_out);
        if inflg == 1
            if ret_fig(1,1,1) == -1
            else
                N_fin = 1;
                % 更新本次送入车道lane_in
                [lane_in1,lane] =
                    Car_in(1,lane,t_in1,t_in2,t_out,lan_fig,N_fin);
                flgi = t_in2(lane_in1);
                car_in_now1 = ret_fig(1,1,1);
```

```
90              end
91          else
92              if numel(car_seq_in) == 0
93              else
94                  N_fin = 0;
95                  car_in_now2 = car_seq_in(1);
96                  car_seq_in(1) = [];
97                  [lane_in2,lane] =
                        Car_in(1,lane,t_in1,t_in2,t_out,lan_fig,N_fin);
98                  flgi = t_in1(lane_in2);
99              end
100         end
101     else
102         flgi = flgi-1;
103         if N_fin == 1
104             result(car_in_now1+1,epoch) = 1;
105         else
106             result(car_in_now2+1,epoch) = 1;
107         end
108     end
109
110     %% 出车过程
111
112     if flgo == 0
113         if lane_out ~= 0 && car_out_now ~= 0 && lan_fig(lane_out,90,1)
                ~= -1
114             x_out = [x_out,car_out_now];
115         end
116         if N_of == 1
117             ret_fig(1,90,1) = car_out_now;
118         end
119         if lane_out ~= 0
120             lan_fig(lane_out,90,1) = -1;
121             lan_fig(lane_out,90,2) = -1;
122             lane(lane_out) = lane(lane_out)-1;
123         end
124         x_seq = x_out;
125 %       [car_out_now,lane_out] = Renew_lan_out(lan_fig); %
    决定接哪条道的车
```

```
126         [lane_out,car_out_now,aa] =Renew_lan_out2(lan_fig,t_out1,t_out2);
127 
128         if car_out_now == 0 || lan_fig(lane_out,90,1) == -1
129         else
130     %         x_out = [x_out,car_out_now];
131             if lan_fig(lane_out,90,1) == -1
132             else
133                 [if_f,~,feq] =
                        Car_out(x_seq,type,color,car_out_now,ret_fig,feq);
134                 if if_f == 1
135                     flgo = t_out1(lane_out);
136                     N_of = 0;
137                 else
138                     flgo = t_out2(lane_out);
139                     N_of = 1;
140                 end
141             end
142     %         end
143         end
144     else
145         result(car_out_now+1,epoch) = 2;
146         flgo = flgo -1;
147     end
148 
149     %% 更新过程
150   [lan_i,lan_j,lan_car]=Car_lan(lan_fig);
151     [ret_i,ret_j,ret_car]=Car_ret(ret_fig);
152     [m,~]=size(lan_i);
153     for i=1:m
154         if result(lan_car(i)+1,epoch) == 0
155             result(lan_car(i)+1,epoch)=str2num([num2str(lan_i(i)),...
156             num2str(lan_j(i))]);
157         end
158     end
159     [n,~]=size(ret_j);
160     for i=1:n
161         if result(ret_car(i)+1,epoch) == 0
162             result(ret_car(i)+1,epoch)=str2num(['7',...
163             num2str(ret_j(i))]);
```

```
        end
    end
    for i = 1:numel(x_out)
        if result(x_out(i)+1,epoch) == 0
            result(x_out(i)+1,epoch) = 3;
        end
    end
    for i = 1:numel(car_seq_in)
        if result(car_seq_in(i)+1,epoch) == 0
            result(car_seq_in(i)+1,epoch) = 0;
        end
    end
end
for kk = 1:flgo
    result(car_out_now+1,epoch+kk) = 2;
end
result(car_out_now+1,epoch+flgo+1) = 3;
    epoch_vec = [epoch_vec epoch];
    val_tmp1 = 0.4*(100-fun_obj11(x_out,type))...
    +0.3*(100-fun_obj22(x_out,color)) + 0.2*(100-ee) +
        0.1*(100-0.01*(epoch-2934));
    val_tmp2 = 0.4*(100-fun_obj11(x_out,type))...
    +0.3*(100-fun_obj22(x_out,color));
    %    val_tmp = 0.4*(100-fun_obj11(x_out,type))...
    %          +0.3*(100-fun_obj22(x_out,color))...
    %          + 0.1*(100-0.01*(epoch-2934));
    value_vec1 = [value_vec1 val_tmp1];
    value_vec2 = [value_vec2 val_tmp2];
end
```

2.2 基于 MTT 原则的入车选择子程序-in_decide2.m

```
function [inflg,car_idx] =
    in_decide2(lan_fig,ret_fig,x_seq,t_in1,t_in2,lane,t_out)
%% 决定将哪一辆车设置为进车车辆

car_idxf= 0;
aa = 90;
if sum(ret_fig(1,1:90,1)) == -90 && numel(x_seq) ~= 0
```

```
    inflg = 0;
    car_idx = x_seq(1);
    return;
end
if numel(x_seq) == 0 && sum(ret_fig(1,1:90,1)) ~= -90
    inflg = 1;
    for k = 1:90
        if ret_fig(1,k,1) ~= -1
            car_idx = ret_fig(1,k,1);
            break;
        end
    end
    return;
end
if numel(x_seq) == 0 && sum(ret_fig(1,1:90,1)) == -90
    inflg = 1;
    car_idx = -1;
    return;
end
for i = 1:90
    if ret_fig(1,i,1) ~= -1
        aa = i-1;
        car_idxf = ret_fig(1,i,1);
        break;
    end
end
[fan_L,~] = count_L(lane,t_in1,t_in2,t_out,lan_fig,1);
[seq_L,~] = count_L(lane,t_in1,t_in2,t_out,lan_fig,0);
if aa+t_in2(fan_L) < t_in1(seq_L)
    car_idx = car_idxf;
    inflg = 1;
else
    inflg = 0;
    car_idx = x_seq(1);
end
end
```

2.3 基于 MTT 原则的出车判断子程序-out_decide.m

```
function [outflg,aa,car_idx] = out_decide(lan_fig,t_out1,t_out2)
%% 决定将那一辆车设置为出车车辆

ti = 90*ones(1,6);
idx = zeros(1,6);
% minv = inf;
for i = 1:6
   for j = 90:-1:1
      if lan_fig(i,j,1) ~= -1
         ti(i) = 90-j;
         idx(i) = lan_fig(i,j,1);
         break;
      end
   end
end
[~,outflg] = min(ti+(t_out1+t_out2)/9);
aa = ti(outflg);
car_idx = idx(outflg);
end
```

2.4 出车车辆选取子程序-Renew_lan_out2.m

```
function [Lo,car_idx,aa] = Renew_lan_out2(lan_fig,t_out1,t_out2)
%% 获得出车车辆
% maxva = 0;
% maxid = 0;
idx = 0;
[Lo,aa,car_idx] = out_decide(lan_fig,t_out1,t_out2);
% if Lo ~= 0 && lan_fig(Lo,90,1) ~= -1
%    lan_fig(Lo,90,1) = -1;
%    lan_fig(Lo,90,2) = -1;
% end
end
```