
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 | | 广西大学 |
| 参赛队号 | | **22105930028** |
| 队员姓名 | **1.** | 叶万兴 |
| | **2.** | 陈珊 |
| | **3.** | 田飞宇 |

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：**

近年来，汽车制造涂装厂在同一混合型生产线上能够根据车辆的动力方式以及驱动方式的不同分别进行涂装制造，从涂装 PBS 接车横移机到送车横移机 PBS 总装的顺序进入涂装车间进行生产，然而涂装车间内部存在重排序调度环节，同时要求不同动力方式和驱动方式的车辆按照最优的排序方式进入生产线。因此，混合生产模式下的涂装-总装车间系统的调度问题作为汽车生产系统的一个极其重要的问题，受到各界学者和研究者的关注。一方面，在过去研究者们通常将涂装车间内部的排序问题和进入总装车间的序列一致性排序问题分别作为两个优化调度问题来解决，忽略了两者问题的关联性。另一方面，由于车型、驱动方式、动力方式、颜色等不同问题，针对混合模式下的涂装-总装生产模式而提出的算法均无法快速的给出快速的优化调度方案。

本研究从理论到仿真模拟，分析了实际涂装生产过程中排序调度需要考虑的多方面问题，结合复杂的生产约束、缓存区结构和生产线结构等问题，搭建出了面向涂装-总装混合车间的排序生产流水线问题的管理与优化方案，采用遗传算法和多目标排序调度算法的混合算法可以大大缩短实际的调度与优化时间，且精度高于传统算法，本研究的主要内容如下：

**针对问题 1**，其主要目标是在约束条件的基础上进行涂装生产过程中**排序调度**，我们采用基于遗传算法的解决策略来求解最优的出库方案，首先获取生产计划中未进入流水线的车次作为出车次数，同时获取车次本身的属性，然后根据**遗传算法**求解出最优的出库方案，判断出库方案中的车次的适应度是否为 0，若为 0，则出车的次数加 1，重复此操作，若适应度值大于 0，则按照出库方案完成出车，判断是否完成生产计划，若完成则该调度过程结束，若没有完成，则继续重复此整体操作。通过对于该问题的模型搭建以及**遗传算法**的计算，我们证明了该算法对于车辆混合型生产线的高能性和适用性。

**针对问题 2**，其约束条件在问题 1 的基础上取消了当返回道 10 和进车道 1 有车身时的优先处理问题，增加了整个车次和系统的随机性，于是我们在问题 1 的基础上采用基于**马尔可夫决策的多目标排序调度算法**来求解最优的出库方案，首先根据进入流水线车次的属性来进行采样一个偏好，根据输出的矢量选择接下来的车次的属性动作，在通过当前车辆的动力方式、驱动方式的不同来返回动作决策，进行一个初始化的矢量奖励，再通过下一辆车的属性继续返回奖励矢量并计算下一个状态，之后继续重复此操作。通过对于该问题的模型搭建以及强化学习的多目标排序调度算法的计算，我们证明了该算法对于车辆混合型生产线的高效性和适用性。



**关键词：遗传算法；马尔可夫决策；多目标排序调度算法；生产调度；PBS 缓冲区**

# 目录

# 一、问题重述

## 1.1 问题背景

近年来，汽车制造厂可以同时在不同车间的混合生产线上生产不同颜色、规格的汽车，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。过去学者们一般将涂装车间内部的排序调度问题和涂装完成后进入总装车间的序列一致性排序调度问题作为两个单独的优化问题分别进行研究，却忽略了两车间的联动性，增加了运营成本。当前大规模汽车生产调度涉及多车型、多驱动、多订单的排序调度，但以往的精确算法、启发式算法均无法快速给出优化的调度方案。

因此，混合生产模式下的涂装-总装车间的系统排序调度问题一直是一个汽车生产系统的极为重要的决策问题，如何迅速而有效的调度混流生产系统的产品生产序列是各大车企当前面临的至关重要的挑战。

## 1.2 问题描述

基于以上研究背景，我们主要考虑的是在 PBS 区域调度能力和限制和已知进车顺序的前提下，如何使得总装进车序列尽可能的满足总装生产需求。本文需要解决以下两个问题：

问题 1：在满足 PBS 约束说明的前提下，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开始计算，尽量使得每连续两辆混动车身之间的非混动车身数都为 2。其次，对出车序列进行分块，判断每一分块的四驱车与两驱车之比是否满足 1:1。以最后一个车身进入 PBS 总装接车口的时刻为总完成时间，尽量减少总完成时间，并减少返回道的使用次数。

问题 2：在不优先处理临近接车横移机、送车横移机的前提条件下，对问题 1 再次进行建模求解最优的满足总装生产需求的总装进车序列。

## 1.3 问题提出

**针对问题 1**：我们做出以下问题归纳：

**总体任务**：

以短时间和高效率完成涂装-总装缓存区调序调度优化的计算序列。

**优化目标**：通过讨论，我们推测可以通过以下环节，实现计算优化：

将两驱混动车记为 $f_1$，两驱燃油车记为 $f_2$，将四驱混动车记为 $f_3$，将四驱燃油车记为 $f_4$。

首先将进入涂装 PBS 接车横移机的车子与车道进行匹配；选择与进入涂装 PBS 接车横移机的车子匹配度最高的车道；优先将 $f_1$ 选择 4 号车道，其次将 $f_2$ 选择 3 号车道，之后 $f_3$、$f_4$ 选择 2、5 车道；将多余的车辆后续安排在 1、6 车道，后续多余的车辆选择通道车身数量最少的车道。该优化目标的具体执行步骤如下：

步骤 1：配置车与车道之间规则的优先级，确定车身与车道需要比较的内容。

步骤 2：对进入涂装 PBS 接车横移机的车与车道进行匹配，确定车身与车道间的匹配属性。

步骤 3：选择匹配值最高的车道，优先将 $f_1$ 考虑 4 号车道，如下一辆车为 $f_1$，则安排在 1 车道或 6 车道；如下一辆不为 $f_1$，则按照 $f_2$ 选择 3 号车道，$f_3$、$f_4$ 选择 2、5 车道，以此类推。

步骤 4：车身进入车道后，将车身的属性赋予各个车道，如有不对应的车辆进入不同的车道，则通过返回道返回至 PBS 接车横移机。

# 二、模型假设与符号说明

## 2.1 模型假设

假设 1：假设接车横移机在空闲时，优先处理返回道 10 停车位的车身，以及送车横移机在空闲时间优先处理最先到达 1 停车位的车身。

假设 2：假设接车横移机在空闲时，可以处理任何返回道 10 停车位的车身，以及送车横移机在空闲时间可以处理任何到达 1 停车位的车身。

假设 3：假设接车横移机和送车横移机从初始位置到各个车道的时间与从对应车道返回初始位置时间相同。

## 2.2 符号说明

| 符号表示 | 含义说明 |
|---|---|
| $G$ | 含有 $i$ 的车次的进入序列，$G=\{G(1),G(2),\ldots G(\mathrm{I})\}$，其中 $G(i)$代表在 $G$ 序列中 $i$ 位置的车次 |
| $G_d$ | 车身的驱动序列 |
| $G_q$ | 车身的动力方式序列 |
| $G_m^{in}$ | 分配到 $m$ 车道上的含有 $I^m$ 个车次的多通道选择缓存区填充子序列，$G_m^{in}=\{G_m^{in}(1),G_m^{in}(2),\ldots,G_m^{in}(I^m)\}$ |
| $G_m^{out}$ | 从 $m$ 车道上的多通道选择缓存区释放的含有 $I^m$ 个车次的释放子序列，$G_m^{out}=\{G_m^{out}(1),G_m^{out}(2),\ldots,G_m^{out}(I^m)\}$ |
| $G^e$ | 合并序列，$G^e=\{Ge(1),Ge(2),\ldots,Ge(I)\}$ |
| $G^f$ | 从 PBS 离开的序列，$G^f=\{G^f(1),G^f(2),\ldots,G^f(I)\}$ |
| $I$ | 序列中车身数目，$\forall i\in\{1,2,\ldots I\}$ |
| $I^m$ | 分配到 $m$ 生产线上的子序列的车身数目，$\forall i\in\{1,2,\ldots I^m\}$ |
| $Q$ | 车型数目，$\forall q\in\{1,2,\ldots Q\}$ |
| $C$ | 驱动数目，$\forall c\in\{1,2,\ldots C\}$ |
| $T$ | 序列 $G$ 中某个车身序列的延误数量 |
| $D_q$ | 车型分布 |
| $D_d$ | 驱动分布 |
| $D_c$ | 当前驱动方式 |
| $DCM$ | 驱动方式矩阵 |
| $C_s$ | 驱动方式决策序列 |
| $AD$ | 连续驱动约束矩阵 |

| | |
|---|---|
| $COS$ | 累计的无序车次列表 |
| $N_d$ | 动力切换次数 |
| $L$ | 总的并行车道数，$\forall_L \in \{1,2,\ldots C\}$ |
| $N$ | 每条车道上的停车位数，$n \in \{1,2,\ldots N\}$ |
| $T_1$ | 当前被执行的送车横移机决策车身的 $T$ 值 |
| $T_2$ | 当前被执行的接车横移机决策车身的 $T$ 值 |
| $LD$ | 上一个离开接送决策的车身驱动 |
| $SS$ | 已存储序列的长度与所有计划调度车身数目的比率 |
| $\omega$ | 目标偏好，$\omega \in \Omega$ |
| $N_\omega$ | 偏好抽样个数 |

# 三、问题分析

为了满足快速变化的市场需求，汽车的类型、驱动、动力方式都在不断扩充。为了快速响应市场的需求，每个车间都要尽量满足混流的生产能力，为了节约成本和提高生产效率，汽车企业往往在两个车间之间设置 PBS 缓存区，对不同类型、驱动、动力方式的车身进行重排，使其满足总装车间的生产约束条件。

目前，对线性缓冲区研究的主要内容集中在利用缓冲区的入库策略和出库策略对车身队列顺序进行重排，缓存区出库的车身队列尽可能满足下游车间的约束，很少考虑线性缓冲区带倒库车道的情况和 PBS 自身的运转效率。由于线性缓冲区的入口和出口一般都只设置一个横移机，当缓冲区存在大量倒库车时，会严重阻碍 PBS 的运行效率，甚至导致总装上线点的“空闲”状态，涂装出口处于“堵塞”状态。

## 3.1 问题一分析

问题一在于搭建 PBS 优化调度模型，首先是优化约束条件，接车横移机和送车横移机在完成任意动作后，必须返回中间位置，而对于接车横移机和送车横移机将每台车送至 1-6 车道的时间又分别为[18, 12, 6, 0, 12, 18]秒，且接车横移机优先处理返回道 10 停车位的车身，送车横移机优先处理到达 1 停车位的车身，于是我们先将车辆本身赋予属性，再将车辆本身与车道进行匹配，匹配原则是优先将 $f_1$ 考虑 4 号车道，如下一辆车为 $f_1$，则安排在 1 车道或 6 车道；如下一辆不为 $f1$，则按照 $f_2$ 选择 3 号车道，$f_3$、$f_4$ 选择 2、5 车道，以此类推。在此基础上运用遗传算法求解最优的出库方案，判断出库方案的适应值是否为 0，若为 0，则下一辆车继续进入接车横移机，重复此方案直到适应度的值不为 0 时，则可以出库；按照出库方案完成出车，判断是否完成生产计划；若完成生产计划则该调度过程结束。

首先，我们将同一驱动方式的车辆进行类聚，皆在将同一驱动方式的车辆分配给同一条生产线，从而降低第 2 阶段潜在的问题，以提高动力方式处理性能，降低时间成本，其次，在实际运用中各条车辆道的平衡是一个非常重要的生产指标，在运行过程中必须加以保证其效率，所以引入以下公式：

$$\Omega=\{m\mid\theta(m,c)>\xi,\forall m\in\{1,...,M\},c=G_c(i)\in\{1,2,...,c\},\forall i\in G\} \tag{3.1}$$

将车道线 $m$ 的阈值 $\theta(m,c)$ 与随机数 $\xi$ 进行比较，符合约束条件的组成候选集合 $\Omega$，基于公式 3.2 进行最优车道 $m$*的选择，即被分配车次将要进入的车道线。此时考虑每条车道线的作业负荷率以保证流水线的运行。

$$m^*=\begin{cases} m, & |\Omega|=1, B_j^{\,m}<B_{\max} \\ \arg_m\min(B_j^{\,m}), & |\Omega|=1, B_j^{\,m}\geqslant B_{\max} \\ \arg_m\min(B_j^{\,m}), & |\Omega|\neq 1, I^m>0 \\ \arg_m\max\left(\dfrac{\theta(m,c)}{\sum_{m=1}^{M}\theta(m,c)}\right), & |\Omega|\neq 1, I^m=0 \end{cases} \tag{3.2}$$

如果带有动力方式属性的车次分配给最优生产线 $m^*$，则所有车道线的响应阈值都会根据（3.3-3.4）的公式被更新。其中，生产线 $m^*$对动力方式的阈值减少$\xi$，而所有其他未选定车道线对动力方式的阈值增加$\varepsilon$，即：

$$\theta(m^*,c) \leftarrow \theta(m^*,c) - \xi \tag{3.3}$$

$$\theta(g,c) \leftarrow \theta(g,c) + \varepsilon \tag{3.4}$$

因此，当下一个带有动力属性 $c$ 的车次到达送车横移机时，由于车道线 $m^*$对于动力方式 $c$ 的响应阈值比较低，在作业负荷率满足约束条件的情况下，车道线 $m^*$接收此车次的概率更高，相应地，其他车道线由于对动力方式 $c$ 的响应阈值比较高而降低接收此车次的概率。

### 3.2 问题二分析

问题二的难度在于相较于问题一取消了优先处理返回道 10 停车位上的车身，同时也取消了优先处理到达 1 停车位的车身，增大了随机性，所以在遗传算法的基础上加入了多目标强化学习的多目标排序调度算法，先一个车身进入接车横移机，一个车身离开送车横移机的过程视为一次迭代，每次迭代中必须做出三个重排序决策过程：送入车道过程、送出车道过程以及返回车道过程。送入车道过程决定车本身应该进入哪个车道，送出车道过程决定哪辆车应该优先送出，返回车道过程决定在一定情况下优先将不符合条件的一辆车先返回至接车横移机处。

多目标优化(Multi-Objective Optimization）问题需找到可以表达目标函数并满足约束条件的决策变量向量并优化此向量的函数，如图 3-1 所示。假定优化目标个数为 $m$，且优化目标之间是相互冲突，则多目标优化问题可以定义为:

$$\min F(x) = \{f_1(x), f_2(x), ..., f_m(x)\} \tag{3.5}$$

其中，$x = (x_1, x_2, ...x_h)^T$ 为 $h$ 个决策变量组成的向量；S 为由约束条件定义的可行域。当以下条件满足时,定义为解 a 可以支配（dominate）解 b。

(1)对所有优化目标而言，解 a 不差于解 b;

(2) 至少在某一优化目标下，解 a 严格优于解 b。

涂装车间灵活的重排序策略可以降低运营成本，提高生产效率，但是也给生产控制部门提出订单顺序生产管理的挑战。对于车身车间而言，同样存在重排序现象，但是由于此时只需考虑车型代码这一唯一属性，所以对生产系统重排序的管理问题相对简单。

综上所述，本研究针对涂装车间的排序调度问题进行研究。实际生产环境中，PBS 的容量时有限的，如果某个车次因动力方式生产调度不当而无法按时进入总装车间，将会产生总装生产的阻塞或空闲造成生产浪费。综上所述，将车身经过 PBS 离开时由于离开决策不同而产生的重排序问题作为本研究需要考虑的第二个重排序问题，称为恢复总装需求序列的重排序问题。通过调整 COS 的大小和离开决策进行重排序，由 NSA 度量此重排序问题。

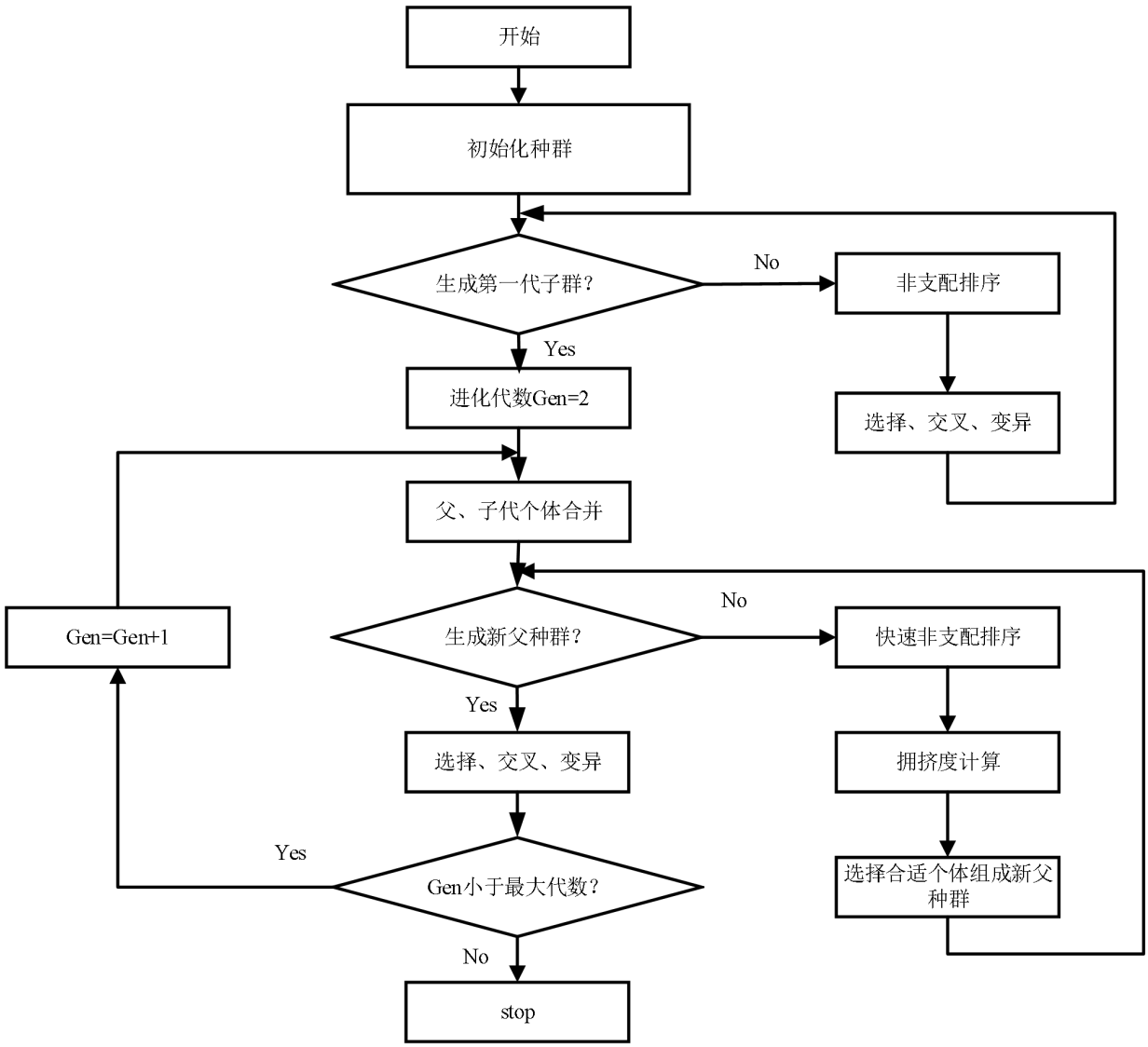
开始
初始化种群
生成第一代子群？
No
非支配排序
选择、交叉、变异
Yes
进化代数Gen=2
父、子代个体合并
No
生成新父种群？
快速非支配排序
Gen=Gen+1
Yes
选择、交叉、变异
拥挤度计算
Yes
Gen小于最大代数？
选择合适个体组成新父种群
No
stop

图 3-1 多目标优化算法

# 四、问题一的求解

## 4.1 问题分析

PSB 缓存区共计六个车道如图 4-1 所示，从上至下的车道编号分别为进车道 8、进车道 5、返回道、进车道 4、进车道 3、进车道 2、进车道 1，每个车道分别有十个停车位，接车横移机和送车横移机的中间初始位在进车道 4 处，考虑到接车横移机和送车横移机从中间初始位置到各个车道接送车并返回中间初始位置的时间不同，从上到下分别为[18,12,6,0,12,18]秒，接车横移机和送车横移机在相邻车道间行进的单程时间可视为 3 秒，车身进入车道后从一个车位移动下一车位耗费的时间为 9 秒。本问题要求按照题中所给约束以及时间数据说明，根据涂装序列的出车序列建立 PBS 优化调度模型，使得总装进车序列尽量满足调度时间最短、效率最高的总装生产需求。针对阐述的问题，本文采用遗传算法进行处理。

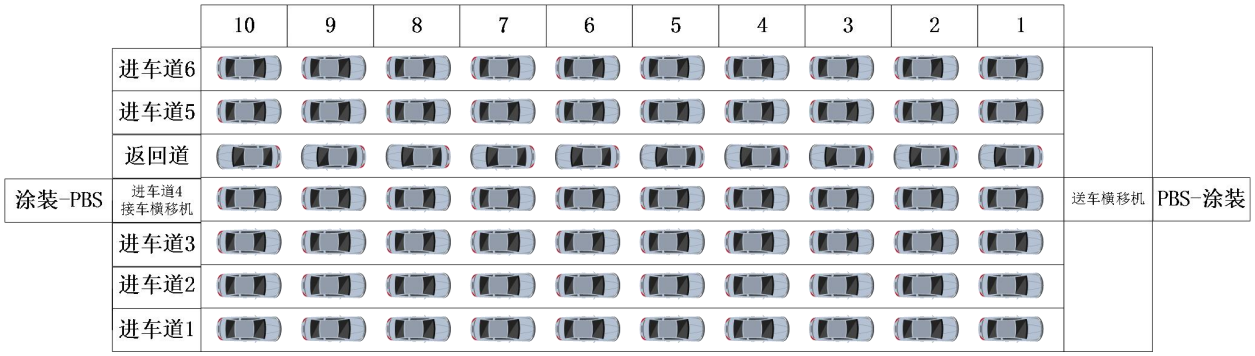


图 4-1 涂装-总装缓存调序区（PBS）示意图

PBS 缓存区由涂装-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。横移机运动时的速度保持一致。

## 4.2 模型建立

### 4.2.1 算法简介

遗传算法（Genetic Algorithm, GA）借鉴了达尔文生物进化论和孟德尔遗传学说，是一种通过人工方式构建的模拟自然进化过程的并行随机搜索最优化方法。遗传算法将“优胜劣汰，适者生存”的生物进化原理引入优化参数形成的编码串群体中，按所选择的适应度函数并通过遗传中的复制、交叉及变异对个体进行筛选，适应度高的个体被保留下来，组成新的群体，新的群体既继承了上一代的信息，又优于上一代。这样周而复始，群体中个体适应度不断提高，直到满足一定的条件。遗传算法的算法简单，可并行处理，并能到全局最优解。

以下对遗传算法的一些相关术语进行介绍：

（1）基因。基因用于表征每个个体的特征。

（2）种群。种群包含所有的个体，是个体的总和。
（3）选择。选择是指以一定概率或者服从某种条件从种群中选择出若干的个体。在遗传算法中一般指筛选旧种群中生命力强的个体以进行繁殖过程。
（4）变异。变异用于模拟生物在自然遗传环境中由于各种偶然因素引起的基因突变，是随机产生的，其出现的概率较小，一旦发生变异，会产生新的染色体，会随机改变遗传基因的值，表现为新的性状。
（5）交叉。交叉模拟自然界中生物繁殖的现象，两个染色体同一个位置的 DNA 通过交换组合的方式产生两个新的染色体。
（6）适应度。适应度用于评价个体对环境的适应程度。

### 4.2.2 算法流程

遗传算法如图 4-2 所示：

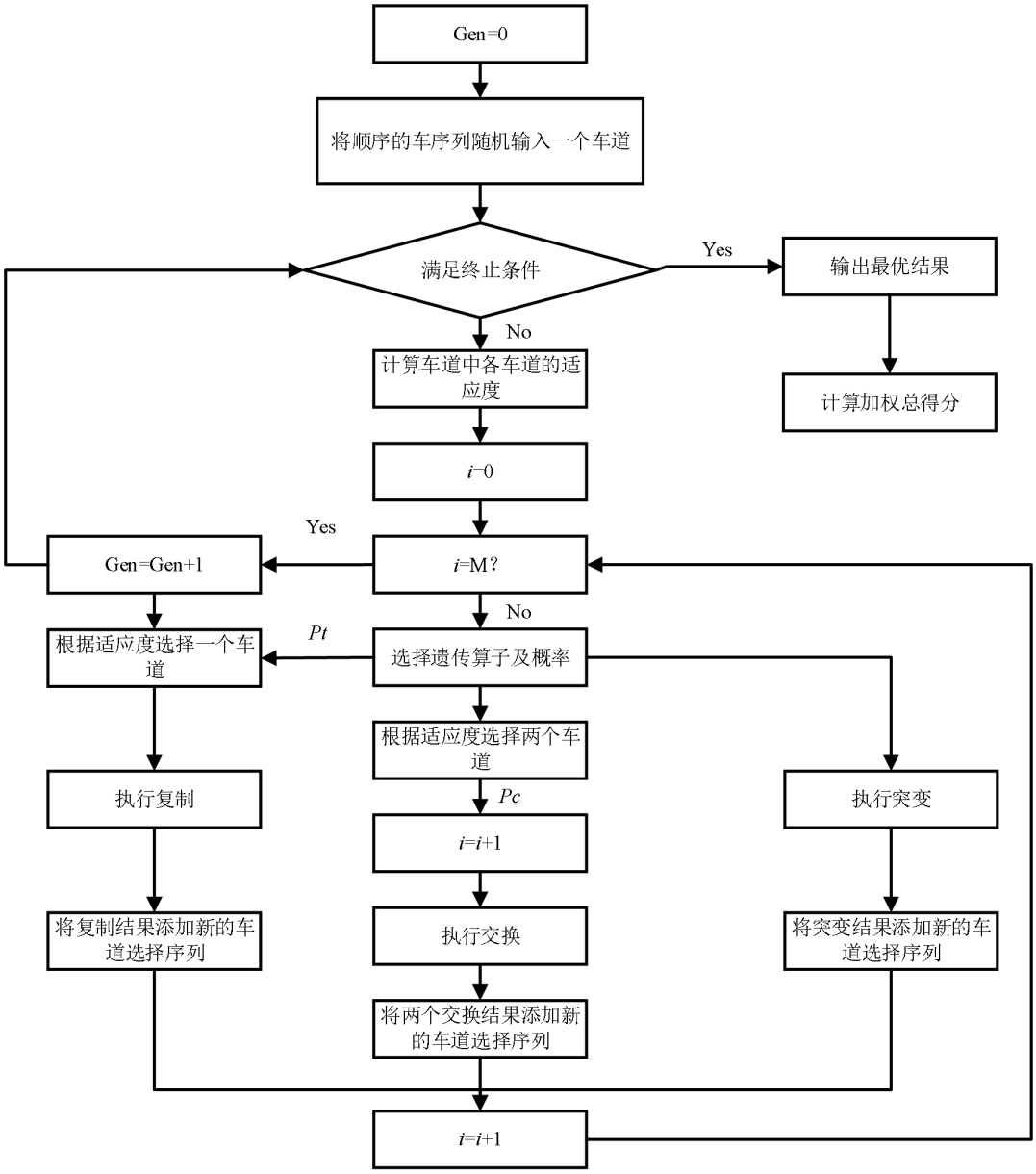


图 4-2 遗传算法流程

Gen：遗传（迭代）的代次。表明遗传算法反复执行的次数，即已产生群体的代次数目。

M：群体中拥有的个体数目。

*i*：已处理个体的累计数，当 *i* 等于 M，表明这一代的个体已全部处理完毕，需要转入下一代群体。

交叉率 *Pc* 就是参加交叉运算的染色体个数占全体染色体总数的比例，记为 *Pc*，取值范围一般为 0.4～0.99。

变异率 *Pm* 是指发生变异的基因位数所占全体染色体的基因总位数的比例，记为 *Pm*，取值范围一般为 0.0001~0.1。

复制概率 *Pt* 用于控制复制与淘汰的个体数目。

**遗传算法主要执行以下四步：**

1、随机地建立由字符串组成的初始群体；

2、计算各个体的适应度；

3、根据遗传概率，利用下述操作产生新群体：

a. 复制。将已有的优良个体复制后添入新群体中，删除劣质个体。

b. 交换。将选出的两个个体进行交换，所产生的新个体添入新群体中。

c. 突变。随机地改变某一个体的某个字符后添入新群体中。

反复执行上述操作后，一旦达到终止条件，选择最佳个体作为遗传算法的结果。

下面对遗传算法的步骤进行扩充说明：

1、编码

遗传算法首先要对实际问题进行编码，用字符串表达问题。这种字符串相当于遗传学中的染色体。每一代所产生的字符串个体总和称为群体。为了实现的方便，通常字符串长度固定，字符选 0 或 1。

2、计算适应度

衡量字符串（染色体）好坏的指标是适应度，它也就是遗传算法的目标函数。

3、复制

根据相对适应度的大小对个体进行取舍，将性能最优的个体予以复制繁殖，性能较差的个体将其淘汰，如此就产生了下一代群体。复制后产生的新一代群体的平均适应度会呈现明显增加的特点。

4、交换

利用随机配对的方法，决定个体之间位置的交换。再利用随机定位的方法，确定这两对母体交叉换位的位置的起始位置。交换开始的位置称交换点。

5、突变

将个体字符串某位符号进行逆变，如由 1 变为 0 或由 0 变为 1，由此得到新个体。遗传算法中，个体是否进行突变以及在哪个部位突变，都由事先给定的概率决定。通常，突变概率很小。

上述 2～5 步反复执行，直至得出满意的最优解。

综上可以看出，遗传算法参考生物中有关进化与遗传的过程，利用复制、交换、突变等操作，不断循环执行，逐渐逼近全局最优解。

### 4.2.3 算法实现

1、编码与解码

将不同的实数表示成不同的 0，1 二进制串表示就完成了编码，因此我们并不需要去了解一个实数对应的二进制具体是多少，我们只需要保证有一个映射能够将十进制的数编码为二进制即可。再将编码后的二进制串转换为十进制串，即求其逆映射，也就是将二进制转化为十进制，也就是解码。

另外需要注意的是一个基因可能存储了多个数据的信息，在进行解码时注意将其分开，如一个基因含有 x,y 两个数据，该基因型的长度为 20，可以用前 10 位表示 x，后 10 位表示 y，解码时分开进行解码。

2、适应度

在实际问题中，有时希望适应度越大越好（如赢利、劳动生产率），有时要求适应度越小越好（费用、方差）。为了使遗传算法有通用性，这种最大、最小值问题宜统一表达。通常都统一按最大值问题处理，而且不允许适应度小于 0。

对于最小值问题，其适应度按下式转换：

$$f(x)=\begin{cases} C_{\max}-g(x) & \text{当}\mathrm{g}(x)<C_{\max} \\ 0 & \text{其他情况} \end{cases} \tag{4.1}$$

其中，$f(x)$为转换后的适应度；$g(x)$为最小值问题下的适应度；$C_{\max}$为足够大的常数，可取$g(x)$的最大值。

为了保证适应度不出现负值，对于有可能产生负值的最大值问题，可以采用下式进行变换：

$$f(x)=\begin{cases} U(x)+C_{\min} & \text{当}U(x)+C_{\min}>0 \\ 0 & \text{其他情况} \end{cases} \tag{4.2}$$

其中，$f(x)$为转换后的适应度；$g(x)$为最小值问题下的适应度；$C_{\max}$为足够大的常数。

3、选择

有了适度函数，然后就可以根据某个基因的适应度函数的值与所有基因适应度的总和的比值作为选择的依据，该值大的个体更易被选择，可以通过有放回的随机采样来模拟选择的过程。

4、交叉和变异

交叉和变异都是随机发生的，对于交叉而言，随机选择其双亲，并随机选择交叉点位，按照一定的概率进行交叉操作。可以通过以下方式实现：首先选择种群中的一个个体作为父亲，然后通过产生一个[0,1]随机数,将其与定义的交叉概率比较，如果小于该数，则在种群中随机选择另外的母亲，随机选择交叉点位进行交叉。

## 4.3 模型结果及分析

根据优化目标具体量化逻辑得到最终加权总分的公式如下：

$$Score=\min\left(0.4score1+0.3score2+0.2score3+0.1score4\right) \tag{4.3}$$

其中，*Score* 为最终目标得分；*score*1 为优化目标 1 得分；*score* 2 为优化目标 2 得分；*score* 3 为优化目标 3 得分；*score* 4 为优化目标 4 得分。

本文根据选题给出的条件以及说明，考虑 PBS 区域调度能力及限制，建立了 PBS 优化调度模型。问题 1 采用遗传算法进行全局最优解的求解，通过 1000 次的迭代获得满足题意要求的最佳出车序列，分别求得附件 1 与附件 2 的出车序列所花费的时间分别为 3816 秒、3744 秒，通过加权得分公式（4.3）分别求得附件 1 与附件 2 的最终得分分别为-82.4000 分、-54.6000 分。如图 4-3 和图 4-4 所示：

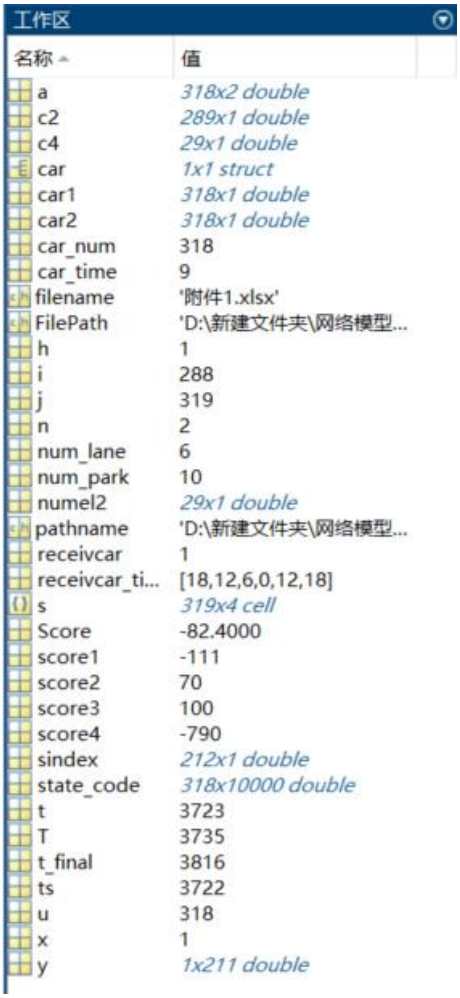


图 4-3 附件 1 数据调度运算结果

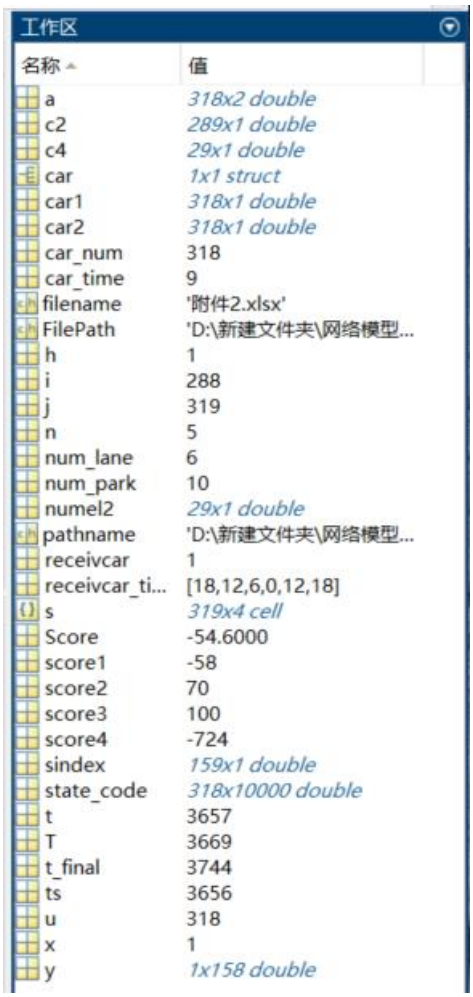


图 4-4 附件 2 数据调度运算结果

# 五、问题二的求解

## 5.1 问题分析

如果 PBS 缓存区容量无限大，则任意序列的涂装车量都可以重排序为总装需求的生产顺序，但是建立过大的缓存区则会增加缓存区的传送设备的采购投入，增加运营成本，因此，PBS 缓存区的容量有限，而问题二的难度在于相较于问题一取消了优先处理返回道 10 停车位上的车身，同时也取消了优先处理到达 1 停车位的车身，增大了随机性，所以在遗传算法的基础上加入了多目标强化学习的多目标排序调度算法，此策略既要考虑运营时间问题，又要考虑车的型号与总装需求序列相比的列延迟量以及接车横移机、送车横移机的优先排序、进一步降低运营成本等问题。

## 5.2 模型建立

针对上述重新排序调度的问题，本节以排序、时间指标为两个优化目标，定量描述调度约束并建立数学模型。定义进入多通道选择缓存区的序列由 $G^{in}$ 表示，每个车次代表一辆车身，每个车身都有两个属性：动力方式和驱动方式。则序列 $G^{in}$ 被描述为如下基于动力方式的序列（5.1）和基于驱动方式的序列（5.2）：

$$G_d^{in}=\left\{G_d^{in}(1),G_d^{in}(2),...,G_d^{in}(I)\right\},\forall G_d^{in}(i)\in\{1,2,...D\} \tag{5.1}$$

$$G_q^{in}=\left\{G_q^{in}(1),G_q^{in}(2),...,G_q^{in}(I)\right\},\forall G_q^{in}(i)\in\{1,2,...Q\} \tag{5.2}$$

相应地，用 $G^{out}$ 表示从多通道选择缓存区中释放序列，序列 $G^{out}$ 被描述为如下基于驱动方式（5.3）和基于动力方式的序列（5.4）：

$$G_d^{out}=\left\{G_d^{out}(1),G_d^{out}(2),...,G_d^{out}(I)\right\},\forall G_d^{out}(i)\in\{1,2,...D\} \tag{5.3}$$

$$G_q^{out}=\left\{G_q^{out}(1),G_q^{out}(2),...,G_q^{out}(I)\right\},\forall G_q^{out}(i)\in\{1,2,...Q\} \tag{5.4}$$

## 5.3 模型求解

### 5.3.1 基于多目标的车辆模型

每条车道均服从 FIFO 原则，因此，车身只能从车道的前端进入，在车道的末端离开。WIP 始终保持不变。初始化 WIP，每次都将送入的车序列的一个车身加入到送车横移机中，同时是否一个车身离开接车横移机。当车道序列 $G^{in}$ 为空时，则 CRS 一个条件结束，形成一个释放的序列 $G_{out}$.

每条车道进入的车身数均小于车道位数，车身必须从非空的车道释放。

### 5.3.2 基于多目标排序调度描述

将一个车身从 PBS 接车横移机进入 10 号停车位，一个车身离开 1 号停车位进入送车横移机的过程视为一次迭代，每次迭代中必须做出三个重新排序的决策过程：送入车道过程、送出车道过程以及返回车道过程。送入车道过程决定车身应该进入多通道选择缓存区的哪条车道；送出车道过程决定哪几个同驱动方式不同动力方式的车身之间交换其动力方式属性，每一个送出车道过程决策 $G^{VP}(i,k)\in\{0,1\}$ 决定多通道选择缓存区中 $i$ 位置的一个车身是否将其动力属性转移到 $k$ 位置的一个车身；返回车道过程决定位于哪条车道的车身应该离开多通道选择缓存区。因此，在一个迭代中，生成了三个决策序列：送入车道决策（5.5）、送出车道决策（5.6）以及返回车道决策（5.7）。

$$G^{ip}=\left\{G^{ip}(1),G^{ip}(2),...,G^{ip}(I)\right\},\forall G^{ip}(i)\in\{1,2,...L\} \tag{5.5}$$

$$G^{rp}=\left\{G^{rp}(1),G^{rp}(2),...,G^{rp}(I)\right\},\forall G^{rp}(i)\in\{1,2,...L\} \tag{5.6}$$

$$G^{vp}=\left\{G^{vp}(1),G^{vp}(2),...,G^{vp}(I)\right\},\forall G^{vp}(i)\in\mathbb{R}^{WIP*WIP} \tag{5.7}$$

其中，$G^{vp}(i)=\begin{cases}1,\text{将多通道选择缓存区中的属性}i\text{ 传递下去}\\0,\text{多通道选择缓存区中的}i\text{ 属性保持不变}\end{cases}$，$\forall i\in\{1,...,I\}$。

定义驱动方式矩阵为 $DCM\in R^{C\times C}$，其中每一个元素 $DCM_{i,j}$ 代表从动力方式 $i$ 切换到动力方式 $j$ 的代价，公式见（5.8）：

$$DCM_{G_c(i),G_c(i+1)}=\begin{cases}1,G_c(i)\neq G_c(i+1)\\0,\quad\text{其他}\end{cases},\ \forall G_c(i)\in\{1,...,C\} \tag{5.8}$$

对于带有多通道选择缓存区的问题而言，第一个目标是最小化释放出多通道选择缓存区的序列 $G^{out}$ 的总量。第二个目标是最小化释放出多通道选择缓存区的序列与进入多通道选择缓存区的序列相比的返回总数。通过确定三个决策序列（5.5-5.7），建立如下的数学模型使得动力驱动切换总成本与返回量 $T$ 最小。

$$[P_1]F_1=total\ \cos t=\min_{G^{ip},G^{rp},G^{vp}}\sum_{i=1}^{I}DCM_{G_c^{out}(i),G_c^{out}(i+1)} \tag{5.9}$$

$$[P_1]F_2=total\ tardiness=\min_{G^{ip},G^{rp},G^{vp}}\left[\sum_{i=1}^{I}\min\left\{T_i\left(G^{out}\right),0\right\}\right] \tag{5.10}$$

将具有相同车型属性的车身集中在一条车道上，使每条车道前端位置的车型分布多样化，可以增加释放过程步骤的车型选择。

重新定义释放的过程。从多通道选择缓存区中释放车辆时，可以选择与上一个释放车身相同的驱动方式，则环境可以直接连续释放该驱动方式的车身而不需要再次执行新的驱动选择决策，此时释放的驱动方式一定是最优决策，即 $N_d$ 不增加，总驱动数目 $C$ 不增加。由于考虑的不止 $C$ 最小化这一目标，还要考虑时间的最小化目标，每释放一个车身所做的决策需要兼顾两个目标的矢量奖励。将释放过程分为两个步骤：

步骤 1：驱动方式选择。选择一种驱动方式作为当前的驱动方式。

步骤 2：释放一辆车。找到一个可以选择当前驱动方式的车身，令 $C_c$=$C$, 将符号 $MC_{C,q}>0$，时间量最大的车身从某车道中释放出来，相应地，$MC_c$ 减 1，然后再调用步骤 1。

值得注意的是，在目标 $F_2$ 中最小化时间，然而在释放过程中选择时间最大的车身释放。这是因为应该尽量释放被拖欠最久的车身以优化总延迟的指标，如某延迟车身的延迟量为 $n$，下一个该驱动方式的车身与该延迟车身的驱动方式和动力方式属性不同，则意味着该延迟车身的延迟量将加 1 变为 $n$+1，进而每一个延迟车身的时间值都会随着动作的更新而

持续增加。因此在释放过程中，算法优选释放当前延迟最大的车身，以防后续造成更大的延迟。

通过释放算法获得释放决策，多目标排序调度算法获得选择决策，该数学模型实现返回车道过程中的交换决策，将问题对应于三个决策变量（5.5-5.7）简化为唯一决策变量，即决策选择每一个车身的动力方式和驱动方式，形成驱动方式决策序列 $C_s$。

$$G^{cs}=[G^{cs}(1),G^{cs}(2),...G^{cs}(I)],\forall G^{cs}(i)\in\{1,2,...c\} \tag{5.11}$$

因而，基于数学模型的目标函数可以被进一步描述如下：

$$[P_2]F_1=total\ \cos t=\min_{G^{cs}}\sum_{i=1}^{n}CCM_{G_C{}^{out(i)},G^{out(i+1)}} \tag{5.12}$$

$$[P_2]F_2=total\ \text{tardiness}=\min_{G^{cs}}[\sum_{i=1}^{G}\min\{T_i(G_c{}^{out}),0\}] \tag{5.13}$$

由于求解的是多目标的优化问题，因而采用多目标马尔科夫决策过程对此多目标优化问题进行转化。

### 5.3.3 多目标马尔可夫决策过程

一般的马尔科夫决策过程可以按照以下公式来表示：

$$<S,A,T,r,\Omega，f_\Omega> \tag{5.14}$$

$$T=P_s(S'|s,a) \tag{5.15}$$

$$r=P_r(s,a) \tag{5.16}$$

$$\sum_{i=1}^{\varpi}\omega_i=1 \tag{5.17}$$

$$f_\omega(r(s,a))=\omega^T(s,a),\omega\in\Omega \tag{5.18}$$

式（5.14）中，$S$ 为系统的状态空间；$A$ 为决策空间；$T$ 是状态的转移概率分布，服从式（5.16），$s$ 表示系统的当前状态；$a$ 表示当前状态下做出的决策；$S'$ 表示系统的下一个状态；$P_s$ 表示转移概率分布；$r$ 是矢量化奖励函数，服从公式（5.16）；$P_r$ 表示在状态 $s$ 下给定动作 $a$ 时可以获得奖励 $r$ 的概率，$r$ 的每一个元素对应一个优化目标；Ω是目标的偏好空间，每个目标偏好 $\omega\in\Omega$ 服从公式（5.17）；$f_\Omega$ 是偏好函数。若当前策略下的奖励为 $r$（$s,a$），$f_\omega(r)$ 将多目标背景下的矢量奖励映射为标量值，表示使用偏好 $\omega$ 时的标量效用。故 $f_\Omega$ 服从公式（5.18）。

## 5.4 模型结果及分析

问题二通过定义车型属性与车道进行匹配，采用多目标排序调度算法进行决策选择，通过三个决策变量的简化变量形成驱动方式决策序列，同时采用多目标马尔科夫决策过程对此问题进行转化求解。分通过上述过程别求得附件 1 与附件 2 的出车序列所花费的时间分别为 3666 秒、3702 秒，通过加权得分公式（4.3）分别求得附件 1 与附件 2 的最终得分分别为-68.6000 分、-49.2000 分。如图 5-1 和图 5-2 所示：


图 5-1 附件 1 数据调度运算结果


图 5-2 附件 2 数据调度运算结果

## 参考文献


[1] Lambora A, Gupta K, Chopra K. Genetic algorithm-A literature review[C]//2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon). IEEE, 2019: 380-384.

[2] Niu X, Wang J. A combined model based on data preprocessing strategy and multi-objective optimization algorithm for short-term wind speed forecasting[J]. Applied Energy, 2019, 241: 519-539.

[3] Singgih I K, Yu O, Kim B I, et al. Production scheduling problem in a factory of automobile component primer painting[J]. Journal of Intelligent Manufacturing, 2020, 31(6): 1483-1496.

[4] Das R, Wang Y, Busawon K, et al. Real-time multi-objective optimisation for electric vehicle charging management[J]. Journal of Cleaner Production, 2021, 292: 126066.

[5] Yang Q, Meng X, Zhao H, et al. Sustainable operations-oriented painting process optimisation in automobile maintenance service[J]. Journal of Cleaner Production, 2021, 324: 129191.

[6] 阎哲，汪民乐，汪江鹏，闫少强，吴丰轩．基于混合遗传算法的海军航空兵场站物资配送车辆调度智能优化[J]．系统工程与电子技术：1-10，2022.

# 附录

部分代码如下：

```
clc
clear
%% 读取初始数据
[filename,pathname] = uigetfile('*.xlsx','Please pick a file');          % 选择数据文件
FilePath = strcat(pathname,filename);                                    % 文件路径
car=importdata(FilePath);
s=car.textdata;
for i =1:numel(car.data(:,:))
    a(i,1)=double(strcmp(cellstr(s(i+1,3)),'燃油'));                     % 混动/燃油：0/1
    a(i,2)=2*double(strcmp(cellstr(s(i+1,4)),'四驱'))+2;                 % 两驱/四驱：2/4
end
car1=a(:,1);
car2=a(:,2);

%% 参数设置
car_num=318;                                                             % 车数量
num_lane=6;                                                              % 车道数量
num_park=10;                                                             % 停车位数量
receivcar_time=[18,12,6,0,12,18];                                        % 车口至车道运送时间
car_time=9;                                                              % 该车位至下一车位运送时间
state_code=zeros(318,10000);                                             % 车辆状态信息
receivcar=0;                                                             % 0 表示空闲，1 表示工作

%% 计算所有汽车在不同时刻的代码状态
T=2;   j=0;
for t=2:10000
if receivcar==0
    j=j+1;
    receivcar=1;
end
if t==T&&t~=2
    receivcar=0;
end
if T==t
n=ceil(rand(1)*num_lane);
ts=T;
switch n
    case 1
    T=receivcar_time(n)/2+T;
```

```
        state_code(j:j,ts-1:T-1)=1;
        for h=num_park:-1:1

state_code(j,T-1+car_time*(abs(h-num_park)+1)-8:T-1+car_time*(abs(h-num_park)+1))=1*(fix(h/10)*10*9)+
10+h;
        end
        state_code(j:j,T-1+car_time*num_park+1:T-1+car_time*num_park+receivcar_time(n)/2)=2;
        state_code(j:j,T-1+car_time*num_park+receivcar_time(n)/2+1:end)=3;
              case 2
        T=receivcar_time(n)/2+T;
          state_code(j:j,ts-1:T-1)=1;
        for h=num_park:-1:1

state_code(j,T-1+car_time*(abs(h-num_park)+1)-8:T-1+car_time*(abs(h-num_park)+1))=2*(fix(h/10)*10*9)+
20+h;
        end
        state_code(j:j,T-1+car_time*num_park+1:T-1+car_time*num_park+receivcar_time(n)/2)=2;
        state_code(j:j,T-1+car_time*num_park+receivcar_time(n)/2+1:end)=3;
              case 3
        T=receivcar_time(n)/2+T;
          state_code(j:j,ts-1:T-1)=1;
        for h=num_park:-1:1

state_code(j,T-1+car_time*(abs(h-num_park)+1)-8:T-1+car_time*(abs(h-num_park)+1))=3*(fix(h/10)*10*9)+
30+h;
        end
        state_code(j:j,T-1+car_time*num_park+1:T-1+car_time*num_park+receivcar_time(n)/2)=2;
        state_code(j:j,T-1+car_time*num_park+receivcar_time(n)/2+1:end)=3;
              case 4
        T=receivcar_time(n)/2+T;
        for h=num_park:-1:1

state_code(j,T-1+car_time*(abs(h-num_park)+1)-8:T-1+car_time*(abs(h-num_park)+1))=4*(fix(h/10)*10*9)+
40+h;
        end
        state_code(j:j,T-1+car_time*num_park+1:end)=3;
              case 5
        T=receivcar_time(n)/2+T;
          state_code(j:j,ts-1:T-1)=1;
        for h=num_park:-1:1

state_code(j,T-1+car_time*(abs(h-num_park)+1)-8:T-1+car_time*(abs(h-num_park)+1))=5*(fix(h/10)*10*9)+
50+h;
        end
```

```
        state_code(j:j,T-1+car_time*num_park+1:T+car_time*num_park+receivcar_time(n)/2)=2;
        state_code(j:j,T-1+car_time*num_park+receivcar_time(n)/2+1:end)=3;
            case 6
        T=receivcar_time(n)/2+T;
          state_code(j:j,ts-1:T-1)=1;
        for h=num_park:-1:1

state_code(j,T-1+car_time*(abs(h-num_park)+1)-8:T-1+car_time*(abs(h-num_park)+1))=6*(fix(h/10)*10*9)+
60+h;
        end
        state_code(j:j,T-1+car_time*num_park+1:T-1+car_time*num_park+receivcar_time(n)/2)=2;
        state_code(j:j,T-1+car_time*num_park+receivcar_time(n)/2+1:end)=3;
end
T=T+1+receivcar_time(n)/2;
if t==1
        T=T-1;
end
end
if j>car_num
        break
end
end
t_final=t+90+receivcar_time(n)-1;
```