**第一章 程序说明**

1.数据包TD.mat

D:起吊位置 卸载位置

T：起吊时间 卸载时间

ta：第i个数：第i个任务最晚结束时间 第2\*i个数:第i个任务最晚开始时间

2.主要变量

a:当前时刻执行的任务

b:当前时刻的冲突情况

c:当前时刻的天车状态

d:当前时刻的目标位置

p:当前时刻的位置

population：任务分配

3.子函数说明

运行时只运行test\_ga.m即可

3.1 Copy\_of\_select\_car.m 生成初代种群

每次计算时，开始位置可能不同，需要手动更改

3.2 hangche.m 计算个体适应度等参数+画轨迹图

1切换不同任务时，需要更改时间数，否则可能时间数过大影响计算速度

2任务有变化时，需要重新导入数据包，更改程序中的数据包的名字

3 每次计算时，开始位置可能不同，需要手动更改

4 速度可能不同，需要更改

5 画图部分的代码在最下面，需要画图的话，将任务的分配情况赋值给population，单独运行这个函数

3.3 isjam.m 判断是否冲突

3.4 my\_fitness.m 记录种群信息

3.5 my\_ga.m 遗传算法

任务有变化时，需要重新导入数据包，更改程序中的数据包的名字

3.6 select\_car.m 选择使用哪辆车（没用到）

3.7 test\_ga.m 启动遗传算法（主函数）

以上数据有变化时需要改变算法的参数

3.8 update\_b.m 更新冲突车辆

3.9 update\_mode.m 更新行车状态

3.10 update\_unb.m 取消冲突

3.11 unb.m 依次取消冲突

**第二章 新型行车调度规则设计**

行车调度方案通过设计一定的规则，表达吊运任务选择行车的过程、行车执行吊运任务的过程以及冲突处理的过程，从而可得到每个吊运任务选择的行车以及行车执行此吊运任务的过程，确定出实际起吊时间和实际卸载时间。

首先当吊运任务发生后，需要依据择车规则来选择行车，完成行车选择后，通过行车状态的变化来表征吊运任务的执行过程，若行车运行过程中发生空间冲突，则需要进行冲突消解。

2.1 吊运任务择车规则设计

当吊运任务发生时需要为此任务分配行车，由于负载行车正在执行其他吊运任务，因此只能选择空载行车。依据各空载行车与此吊运任务的匹配程度来分配行车[10]。吊运任务发生时首先依据空载行车与其的相对位置关系确定各行车与此吊运任务的匹配度，进而在匹配度最高的行车集合中选择距离此吊运任务的起吊工位最近的行车。用表示行车与吊运任务的匹配度，匹配规则如表3所示。

表3 空载行车与任务匹配规则

|  |  |  |
| --- | --- | --- |
| 类型 | 判断条件 | 空载行车与任务的匹配度 |
| 类型1 | 起止工位间不存在负载状态的行车 |  |
| 类型2 | 起止工位间存在负载行车且反向行驶 |  |
| 起止工位间存在负载行车且正向行驶 |  |

在确定各行车与吊运任务的匹配度时，依据起止工位间是否存在负载状态的行车分为类型1和类型2，若不存在则依据类型1赋值；若存在则依据类型2赋值。类型1是根据吊运任务的起止工位将车间划分为3块，位于起止工位之间的空载行车与吊运任务的匹配度最高为3，靠近起始工位的外侧空载行车为1，靠近卸载工位的外侧空载行车为1。类型2根据车间中负载状态的行车将车间划分为2块，若行车反向行驶，此行车左侧的其他空载行车匹配度为1，右侧的为2；反之，此行车左侧的其他空载行车匹配度为2，右侧的为1。

2.2 行车状态更新规则设计

在每个仿真时刻，需要更新所有行车的状态，以表征吊运任务的执行过程。行车状态包含：所处位置、负载任务状态、被动移动状态、所处阶段、目标位置。

1）所处位置为行车当前时刻在车间内的位置。

2）负载任务状态分为空载状态和负载状态。当行车执行某一个吊运任务时，为负载状态，负载任务状态为此吊运任务编号。当行车不执行吊运任务时，即行车空载状态，负载任务状态为 0。

3）被动移动状态是指当前时刻行车是否被动。若此行车与其他行车未发生空间冲突，则此行车不被动移动，被动移动状态为 0；当发生冲突，若此行车所执行的吊运任务的优先级较高，则此行车不被动移动，被动移动状态为 0；相反，此行车执行的吊运任务的优先级较低，则此行车被动移动，被动移动状态为任务优先级高的行车编号。

4）所处阶段是行车当前时刻的移动的目标位置。行车执行一个吊运任务的过程为行车从空载位置移动至起吊位置，到达起吊位置后吊起任务，再从起吊位置移动至卸载位置，最后在卸载位置完成任务的卸载。可以将行车执行吊运任务的过程划分为 4 个阶段：空载起吊移动阶段，即行车从空载位置向起吊位置移动的过程，取值为 1；起吊阶段，即行车在起吊位置上吊起任务的过程，取值为 2；负载移动阶段，即行车从起吊位置向卸载位置移动的过程取值为 3；卸载阶段，即行车在卸载位置上卸载任务的过程，取值为 4。此外行车不执行吊运任务时，为空载阶段，取值为 0。

5）目标位置是行车在当前时刻所要移动的终点位置。当行车不被动行驶时：若行车空载，则目标位置为当前位置，代表行车暂停在当前位置。若行车执行吊运任务，则空载起吊移动阶段，行车目标位置为吊运任务的起吊位置，表示行车处于空载起吊移动阶段时，行车要向起吊位置移动；起吊阶段，行车目标位置为起吊位置，表示行车暂时停留在起吊位置进行起吊；负载移动阶段行车目标位置为吊运任务的卸载位置，表示行车在此阶段要向卸载位置移动；卸载阶段，行车目标位置为卸载位置，表示行车暂时停留在卸载位置进行卸载。当行车被动行驶时，行车的目标位置与执行优先级高的吊运任务的目标位置距离。

在行车调度时间范围内，需要在每个时刻对所有行车的所有状态依次进行更新，以行车为例说明其在不被动移动情形下各状态的更新规则，如图4所示。

1）若当前时刻行车的负载任务状态值为0，即时行车处于空载状态。行车暂时停留在当前位置不动，则按空载阶段更新各状态。

2）当行车被一个吊运任务选择，则状态更新规则改变。若行车此刻不处于起吊位置，行车处于空载起吊移动阶段，各状态按照空载移动状态进行更新。当行车到达起吊位置时，时，进入起吊阶段，各状态按照起吊阶段更新。

3)当行车所处阶段值为2，行车处于起吊阶段。若处于此阶段的时间未达到时，则各状态按起吊状态更新。当达到时，即行车已完成起吊，开始向卸载位置移动，进入负载移动阶段，各状态按卸载阶段更新。

4）当行车所处阶段值为3，且行车未达到卸载位置时，即且时，行车处于负载移动阶段，各状态按卸载阶段更新。当行车到达卸载位置时，进入卸载阶段，各状态按负载移动阶段更新。

5）当行车所处阶段值为4，行车处于卸载阶段。若处于此阶段的时间未达到时，则按负载移动状态更新。当达到时，即行车已完成卸载，各状态按空载阶段更新。

在更新行车状态时，首先需要判断是否会与其他行车发生冲突，若发生冲突是否处于被动移动状态，若行车被动移动状态为0，则根据行车所处阶段更新行车状态，否则状态更新规则改变。



图4 行车状态更新规则

2.3 冲突消解规则设计

由于车间内所有行车只能在同一轨道上运行，而且行车之间相对位置不变，所以当相邻行车彼此间距缩小为安全距离时就会发生空间冲突，行车冲突类型如图5所示。

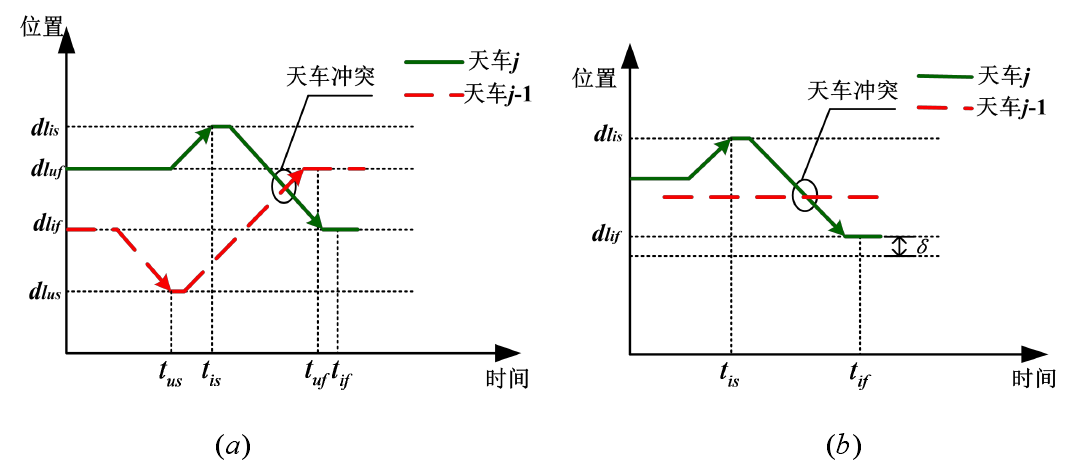


图5 行车冲突类型

图5为相邻行车与发生空间冲突的2种类型。分图(a)中行车执行吊运任务，行车执行吊运任务，二者在执行吊运任务过程中相向而行发生冲突，分图(b)中行车空载，行车执行吊运任务，且行车执行吊运任务过程中与相邻空载行车发生冲突。

当相邻行车发生冲突后，依据判断行车执行的吊运任务的优先级高低，让执行优先级低的吊运任务的行车先跟随另一台行车被动移动的方式进行冲突消解。本文中吊运任务的优先级按照任务执行时间顺序进行排序，先执行的任务具有更高的优先级。优先级具体取值如下：

1）空载行车的优先级最低，为0。

2）当行车处于起吊阶段或卸载阶段，表明行车正在起吊或卸载吊运任务，此时任务的优先级改变为（无穷大），直到完成当前阶段再恢复原来的优先级。

冲突消解的规则为：发生冲突后，比较两台行车执行的吊运任务的优先级，吊运任务优先级低的行车改变移动方向和目标位置，跟随另一台行车被动移动，直到其到达目标位置后才恢复原方向和原目标位置，如图7所示。

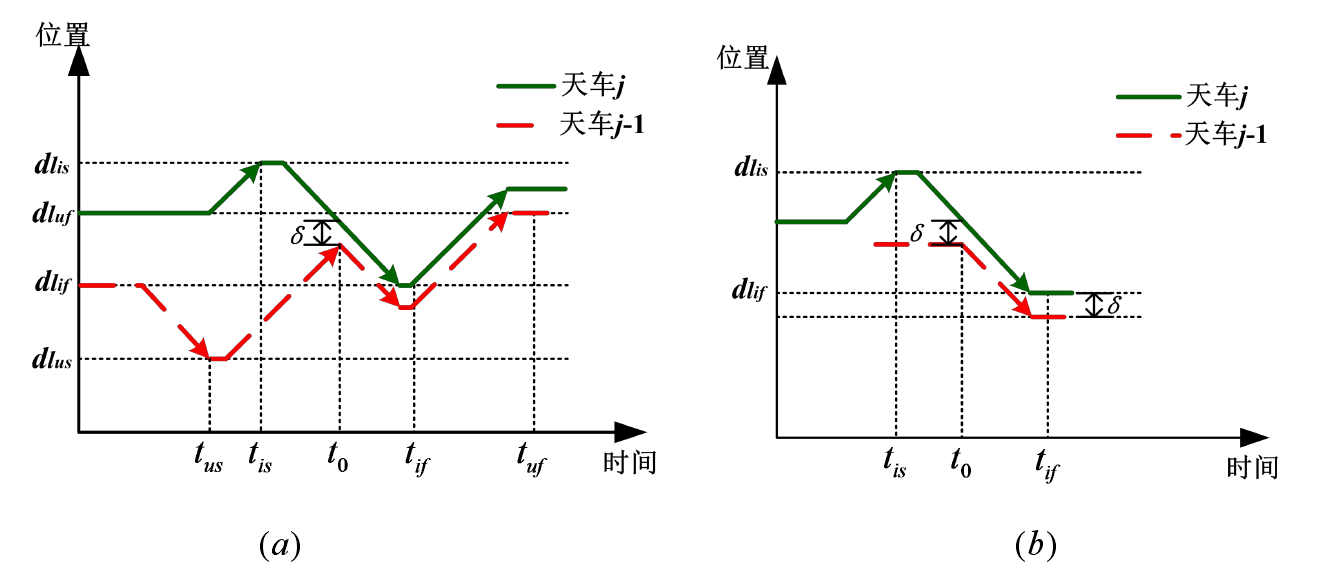


图6 冲突消解过程

图6 (a)为图5 (a)的消解过程，用表示冲突发生的时刻。若行车执行的吊运任务的优先级高于行车，则按照表4进行如下更新：

1）时刻发生冲突，行车执行的吊运任务的优先级较高，行车较低，则行车改变行驶方向跟随行车被动移动。此刻行车处于卸载阶段，目标位置为，则行车被动移动至的位置，即行车的各状态更新规则不变，而行车的各状态更新规则按规则1更新。

2）行车到达目标工位完成卸载后，优先级为0，低于行车的任务优先级。因此行车恢复原来的行驶方向，行车被动移动，此刻行车的目标位置为，则行车被动移动至的位置，即行车的状态按规则2更新，行车的状态按规则3更新。

图6(b)为图5(b)的消解过程，行车是空载状态，则行车跟随行车被动移动，此刻行车的目标位置为，则行车被动移动至的位置。当相邻行车的冲突引起到其他行车冲突，例如行车与行车发生冲突，而行车优先级较低，则跟随行车被动行驶，当时，也会与行车发生冲突。此时需要比较行车和行车执行的吊运任务优先级，当行车执行的吊运任务优先级较高，那么行车也改变行驶方向、被动移动状态、目标位置：，，直到行车完成当前阶段的运输再恢复行驶方向和目标位置，被动移动状态恢复为0。相反若行车执行的吊运任务优先级较高，则行车和行车都改变行驶方向、被动移动状态和目标位置，，，，直到行车完成当前阶段的运输再恢复行驶方向、目标位置，被动移动状态恢复为0。

表4 冲突消解阶段更新规则

|  |  |  |
| --- | --- | --- |
| 更新规则1 | 更新规则2 | 更新规则3 |
|  |  |  |

2.4 三车辆的冲突消解规则设计

当行车数量达到三辆及以上时，行车的冲突消解规则将会与两辆时明显不同。这是因为处于两边的车辆会受到中间位置车辆的牵连，从而当中间位置的车辆发生冲突时，有可能影响到两边的车辆，使得冲突消解规则变得更为复杂。本文以三辆行车为例，设计了适用于三辆行车的冲突消解规则。与两辆行车相比，三辆行车的冲突消解规则存在以下两点不同：更新行车状态的顺序和所有行车更新后的检查冲突部分。

2.4.1 更新顺序

当其中某行车发生冲突后，不能按照正常的更新顺序更新天车状态，否则会发生错误。应该按照冲突发生的顺序更新，先更新冲突车辆，后更新被冲突的车辆。对于三辆行车，共有8种冲突情况，3种更新顺序。具体情况如表所示。其中，第一行表示冲突情况的编号，第2、3、4行分别表示与其他车发生冲突，0为不与任何车发生冲突，数字为当前发生冲突的车辆。

例如第二种情况，1车与2车冲突，只有当2车完成任务以后，1车才能继续原来的任务。所以更新是应该先更新2车，再更新1车。此时，3车可以先更新，也可以最后更新。

第5行表示更新顺序。由于行车之间存在一定间隔，所以只存在相领行车间的冲突，跨行车之间不存在冲突。

表5 冲突下更新状态

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 冲突情况 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1车 | 0 | 2 | 0 | 0 | 0 | 2 | 2 | 0 |
| 2车 | 0 | 0 | 1 | 3 | 0 | 0 | 3 | 1 |
| 3车 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 2 |
| 更新顺序 | 1 2 3 | 2 1 3 | 1 2 3 | 3 2 1 | 2 1 3 | 2 1 3 | 3 2 1 | 1 2 3 |

2.4.2 检查冲突

由于处于中间的行车位置特殊，在全部行车更新完成后，中间位置的行车仍然有可能与其相邻的行车发生冲突。因此，当中间位置行车的冲突情况发生变化时，需要检查全部行车的冲突情况。具体过程为，按编号依次判断相邻的行车是否发生冲突，如果发生冲突，更新行车状态，直到所有行车的冲突情况更新为合理的状态。

根据以上规则进行仿真流程设计如下：

1、生成吊运任务序列。根据一定时间段内车间的生产作业计划，生成吊运任务序列，同时确定各任务的优先级。

2、参数初始化。初始化行车数、各行车在车间内的位置；初始化参数任务标号，行车标号。仿真时钟置于吊运任务序列中第一个任务的最早起吊时间，即设置。

3、判断当前时刻是否为吊运任务的最早起吊时间，即判断是否成立，若成立，则依据择车规则选择行车，并根据选择的行车当前时刻的位置按空载移动阶段或起吊阶段更新行车各个状态。否则令，再依次判断当前时刻发生的吊运任务，并为发生的任务选择行车，直至。

4、判断行车是否会与相邻行车冲突。若存在，则若行车执行的吊运任务优先级较低，按表4中规则1更新行车的各个状态。否则根据行车的负载任务状态和所处阶段更新状态。若则令，依次更新状态，直至。

5、仿真时钟推进。若到达仿真的结束时刻，则输出仿真结果，否则推进仿真时钟，令并转至步骤3。

**第三章 新型行车调度模型优化**

以上建立的仿真模型用于生成可行的行车调度方案。然而单纯的仿真难以达到优化的效果，于是在前文对行车调度问题分析的基础上，建立行车调度数学模型。

根据该钢厂现实生产中，不同条件下对行车调度的要求不同。同一时间段内的吊运任务较多，而行车数量有限，因此在行车资源紧张环境下，行车调度效率最高可作为优化目标。而对于行车资源不紧张情况下，稳定性和安全性应成为首要优化目标[11]。行车调度的效率体现在所有吊运任务的等待被运输时间最短、总运输时间最短，而稳定性安全性体现在行车之间负载时间差异最小[12]。因此本章以所有吊运任务等待被运输时间最短、总运输时间最短、行车之间负载时间差异最小为优化目标，考虑行车空间位置约束，以及吊运任务的时间约束建立行车调度模型。

3.1假设条件

针对钢厂连铸车间行车调度问题特点，作如下假设：

1）行车、加工工位设备都能正常运行，不考虑行车、工位设备故障；

2）吊运任务执行期间不能中断；

3）行车上的小车沿车间宽度方向移动并吊起或卸载吊运任务时间相同，为常量；

4）各行车沿车间长度方向的运行速率相同；

5）当行车执行完一个吊运任务，若不处于被动移动状态，则暂时停留在此吊运任务的卸载工位，直到开始执行下一个吊运任务。

3.2 数学模型

行车调度数学模型目标函数建立如下：







式（3.1）为所有吊运任务的实际起吊时间与最早起吊时间的差值最小，即等待被运输的时间最短，其中为吊运任务的实际起吊时间。式（3.2）为所有吊运任务的总运输时间最短，其中为吊运任务的实际卸载时间。式（3.3）为所有行车之间负载时间差异最小，其中为行车在调度时间范围内的负载时间，；为各行车负载时间平均值。

约束条件包括：



















式（3.4）表示对于任意一个吊运任务只能由一台行车执行。式（3.5）表示对于任意一台行车，同一时刻最多只能执行一个吊运任务。式（3.6）表示对于任意一个吊运任务，若行车执行吊运任务，那么在卸载前，行车会一直被任务占用；若行车不执行吊运任务，则不会被其占用。式（3.7）表示任意一个吊运任务，其实际起吊时间不小于最早起吊时间。式（3.8）表示任意一个任务的实际起吊时间不超过最晚起吊时间。式（3.9）表示任意一个任务的实际卸载时间不大于最晚卸载时间。式（3.10）表示行车在车间中的相对顺序不变，且相邻行车间距不小于安全距离。式（3.11）表示行车沿车间长度方向运行的速度约束，行车下一时刻位置与当前位置的约束。式（3.12）表示车间中各行车的运行范围，为车间总长。

3.3 模型评价

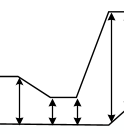
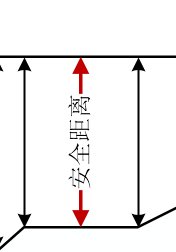
该模型通过设计一系列规则精确刻画行车的调度过程。规则主要包括：针对当前任务选择行车；在特定时间内预测冲突；根据任务优先级进行冲突消解；实时更新行车状态。通过仿真模型可确定每个调运任务选择的行车及行车执行调运任务过程，生成可行的行车调度方案，综合考虑行车效率、安全、稳定性建立最优目标数学模型及空间、时间约束条件。该模型与原模型相比主要从以下方面进行了改进：

1、从并行性看，新方法建立模型以任务为入手点，在仿真模型构建上从多方面描述任务的执行过程，采用状态更新机制，在规则约束下按照时间推进可同时对多台行车进行调度。例如，图7为原方法任务分配时两台行车的运动轨迹，可以明显看出，开始阶段行车1接到任务开始执行，行车2一直待在原地不动，而下一阶段行车2开始执行任务，行车1待在原地不动。细节如图8。

如此这样执行，直到行车2开始向行车1方向移动执行任务，出于安全距离约束条件的限制，行车1开始被动避障使两车始终存在在安全距离。

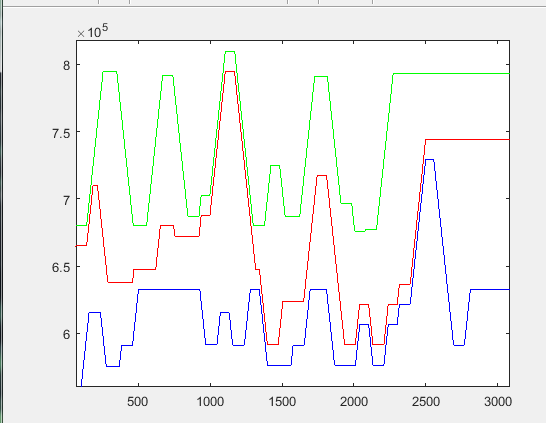
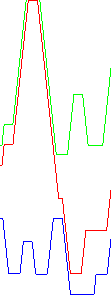


图7 原模型行车相对位置图

1. （b）

图8 原模型细节分析图

1. （b）

图9 本模型行车轨迹

2、从阶段划分看，同之前方法相比，新方法对整个调度过程的表示采用状态更新机制，即在每个仿真时刻更新行车所处位置、负载状态、被动移动状态、所处阶段及目标位置。相比于原方法采用位移函数进行阶段划分，新方法从多角度表示行车状态变化，原方法中的等待和避让行驶过程在新方法中可通过负载状态、被动移动状态及目标位置的变化所替代。

原方法中阶段4对应新方法中的空载移动阶段，5、8、9分别对应新方法中的起吊阶段、负载移动阶段和卸载阶段，这些阶段未见明显区别。以最复杂的一种情况为例：某行车在执行任务过程中某一时刻为空载等待状态，这时在其运行轨道上会和某负载行车发生冲突，按照优先级比较空载行车需避让，此过程对应了原方法的1、2阶段。而在新方法中，因未分配任务，此过程为空载状态，被动移动状态为1，目标位置为需避让的负载行车的目标位置加上安全距离。当该行车分配任务后负载前往目标位置时，若再次因冲突避障其他行车，则对应原方法中的6、7阶段。在新方法中该阶段被统一定义为负载移动阶段，而是在目标位置和和被动移动状态处发生相应改变。

综上，新方法通过增加刻画维度来更准确形象的描述行车状态变化过程，简化了阶段的划分，这也正是新方法机制优于原方法机制的原因。

3.3仿真运行流程

仿真运行过程流程图如图10。



图10 仿真运行过程流程图

**第四章 遗传算法智能优化求解**

本章中首先在前章节中对行车调度问题分析的基础上，在仿真模型中引入遗传算法求解数学模型，旨在利用遗传算法的迭代进化思想实现行车选择的优化，从而得到可行且智能优化优化的行车调度方案。

仿真模型仅仅是对行车执行吊运任务过程的演化，未优化行车调度方案，因此引入遗传算法，旨在利用遗传算法的迭代进化思想实现行车调度方案的智能优化。遗传算法作为计算机科学人工智能领域中用于解决最优化的一种搜索启发式算法，由于并行性、自组织、强鲁棒性等特点被广泛应用于求解生产调度问题，因此本研究也采用遗传算法进行求解。

4.1 编码设计

遗传算法需要通过编码来处理行车调度问题的参数，所以需要将求解的参数按照一定的结构组成染色体。连铸车间行车调度问题求解的是行车调度方案，包括求解吊运任务序列中每个吊运任务选择的行车、求解实际起吊时间及实际卸载时间。在编码时，采用实数编码方式，编码由安排执行吊运任务序列所对应的行车编号构成，如图11所示，一个吊运任务对应一个行车，保证满足约束（3.4）。即将一个行车序列作为一个解，作为编码的依据，一个完整的编码序列就是一个染色体，依此进行后续的计算，确定每个吊运任务的实际起吊时间和实际卸载时间[14]。在进行遗传算法求解时，吊运任务序列的顺序保持不变，但执行吊运任务的行车可以不同，即编码序列会发生变化。

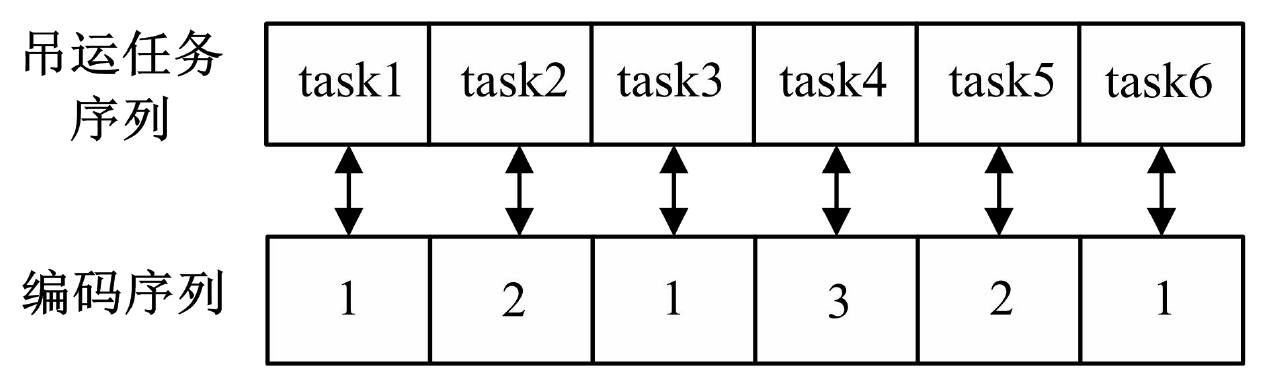


图11 编码示意图

4.2 种群初始化

一般的种群初始化操作都是采用随机初始化的方法，但由于初始种群性能的好坏对于遗传算法的搜索效率非常关键。因此为了提高初始种群的性能，采用基于调度规则的初始化方法。此外由于行车调度是时间、空间多尺度问题，各吊运任务也并不是在同一时刻发生，因此种群初始化也并不是在算法初始时刻全部完成，而是随着算法的推进、时间的推进，当一个吊运任务发生，利用调度规则对染色体上相应位置的基因初始化。待吊运任务序列中，最后一个吊运任务发生时，染色体上对应位置基因初始化完成后，一条染色体才完成初始化。如此经过多次仿真运行就可生成初始种群。

仿真模型中是根据吊运任务的择车规则来初始化种群，但为了保证种群的多样性，设置一些随机因素，具体为：①吊运任务序列中第一个吊运任务发生时，随机选择一台行车，其相应状态由空载状态更新为负载状态；②后续吊运任务发生时，根据空载行车与吊运任务的匹配规则确定每台行车的匹配度，在匹配度最高的行车集合中随机选择一台行车，更新选择的行车状态为负载状态，当最后一个吊运任务完成后，即所有任务已完成行车匹配，对应地生成了一条染色体，依此产生种群规模数量的染色体。

4.3 适应度函数设计

当行车完成了调度时间范围内所有的吊运任务后，就形成一个完整的染色体。需要计算染色体的适应度值来对其性能进行评价[16]。行车调度的模型中有三个目标函数，将三个目标加权得到单目标，再通过变换得到适应度函数如式（4.1）：



式（4.1）中，使得目标函数值最小的染色体适应度值最小。、、分别为三个目标的相对重要程度，实际应用时可根据调度需求进行设计。例如在行车调度资源不紧张情况下，可考虑增加行车之间负载时间差异最小的权值，适当减小所有吊运任务的等待被运输时间和总运输时间的权值，使得稳定性和安全性成为首要追求目标；而若行车资源很紧张，则考虑减小行车之间负载时间差异最小的权值，增加所有吊运任务的等待被运输时间和总运输时间的权值，使得行车调度效率最高成为首要追求目标。

当适应度计算完之后还需检查染色体是否合格。检查原则有三条：

1、检查任务的实际运输时间是否满足不超过对应任务的预计起止时间范围约束，即是否满足约束条件式（3.7）、（3.9）。

2、检查任务实际起吊时间是否满足不超过最晚起吊时间，不小于最早起吊时间约束，即是否满足约束条件式（3.7）、（3.8）。

3、检查行车调度方案是否满足不存在两个执行重钢包任务的行车冲突的情况。上述三条原则均满足，视为合格。对于不合格染色体，改变适应度值为极小值，以降低被后代个体选择的概率。

4.4 遗传算子设计

为了保证种群的多样性以及最优的个体被选中，选择算子采用最优保存法和锦标赛选择法相结合的算法，操作过程为[17]：

1）通过比较种群中所有个体的适应度值，将适应度最高的个体复制到下一代种群中；

2）在当前种群中随机选择两个个体，将适应度值较高的个体复制到下一代；

3）重复2）直到下一代种群中个体数量已经达到初始种群数量。

交叉算子采用均匀交叉法。步骤为：按照交叉概率选择两个个体，对两个个体的每对基因，生成一个随机数0或1，随机数为0则不变，随机数为1时交换基因。变异算子采用单点变异法，按照变异概率选择一个个体，随机选定变异位置，随机选择一台行车，将基因值改变为相应的行车编号。

4.5 停止迭代准则

当迭代次数到达最大迭代次数或迭代误差收敛到设定限度，则认为算法收敛，停止迭代。检查最终的行车调度方案是否满足吊运任务起止时间约束，是否存在执行重钢包任务的两台行车冲突的情况。若不满足时间约束或者存在两个重钢包冲突情况，则认为此方案不可行，重新运行程序直到行车调度方案可行。

遗传算法流程如图12所示。



图12 遗传算法流程图