智能 RGV 的感知预判调度策略

摘要

随着自动化技术的不断发展，加工车间的自动化水平和生产效率也不断提高。在一个有 8 台计算机数控机床（CNC）、1 台可移动式自动导引车（RGV）的系统中，针对不同的情况我们提出了相应的动态调度策略。

针对情况一，物料需要单工序加工，我们提出了三种 RGV 动态调度策略。（1）完全队列策略：通过 RGV 的任务队列，实现其按照 CNC 呼叫的先后次序前往进行上下料工作。这种调度算法简单易实现，但整体效率低；（2）一呼百应策略：根据启发式算法，通过启发函数来计算不同 CNC 的启发值，仅当有一个 CNC 进行呼叫时，RGV 更新所有 CNC 的启发值，利用最小优先队列来进行任务调度，从而节约整个系统时间；（3）感知预判策略：我们重新定义启发函数，将 RGV 与 CNC 移动时间和 CNC 当次加工的剩余时间取最大值作为启发函数中的一项，从而实现 RGV 移动与 CNC 加工同时进行、RGV 更加精准选择 CNC，进一步提高系统的效率。

针对情况二，物料需要双工序加工。（1）加工前 CNC 选取最优配置：我们通过非线性规划计算理想情况最大产率，对不同工序 CNC 分配最佳数量，之后对不同位置依次运行无故障情况的感知预判策略，以产率最高为标准，得出加工前的最佳配置。（2）双工序调度策略：我们扩展改进单工序的感知预判调度策略，提出双层感知预判策略，修改启发函数，既能将一二道工序独立来看，也能在寻找第一道工序 CNC 时也能将眼光放长远，找到一条两道工序消耗时间和最短的路径。

针对情况三，CNC 发生故障。CNC 故障后，完全队列策略中 CNC 不呼叫，另两个策略中的启发值变为无穷大，从而告知 RGV。在修复完成后，完全队列策略中的 CNC 入队；另两个策略中的 CNC 的启发值更新为初始值。

确认好策略后，分别进行测试。从最终效果来看，无论有无故障，单工序中感知预判策略都有最高的产率和效率，且产率接近理论极限；在双工序中，双层感知预判策略的两道工序的产率和效率相近，也没有明显短板效应。

我们的模型结构简单，时间复杂度低。且在考虑故障率的情况下经过多次测试具有较高的稳定性，对实际工厂流水作业有较高的应用价值。

关键词：启发式算法；感知预判调度策略；智能调度算法；短板效应

1、问题重述

1.1 问题背景

一种智能加工系统由 8 台 CNC（计算机数控机床）、1 辆 RGV（轨道式自动引导车）、RGV 直线轨道以及上料下料传送带等附属设备组成。通过 RGV 在固定轨道上智能运行来完成对 CNC 的上下料和清洗物料等作业任务。其中对 RGV 的调度策略对系统能否高效地工作至关重要。

1.2 问题提出

（1）对于需要一道工序加工的物料，且在不发生故障的情况下，建立 RGV 的动态调度模型并给出相应求解算法；

（2）对于需要两道工序加工的物料，且在不发生故障的情况下，由于两道工序不能由同一台 RGV 完成，且作业过程中也不允许更换刀具，在此基础上确定完成第一道、第二道工序的 CNC 数量、位置，建立 RGV 的动态调度模型并给出相应的求解算法；

（3）考虑 CNC 加工过程中的故障情况：故障发生率约 1%，故障排除时间介于 10～20 分钟之间，且在故障排除之后即可恢复作业。在此前提下考虑（1）和（2）中的问题；

（4）将表 1 中三组系统作业参数代入模型中，检验模型的实用性和算法的有效性，同时得出 RGC 具体的调度策略和系统的作业效率。

2、模型假设

1.忽略上料、下料传送带在题目中的作用。按照题目要求，由于传送带既能连动，也能独立运动，我们假设在任何时刻 RGV 都能够从上料传送带中获得需要的生料，任何时刻下料传送带都有位置运送加工完成后由 RGV 取下的熟料；

2. RGV 连续移动一段距离比分开移动同样的距离所花费的时间更短；

3.所有物料性质均相同；

4.在一个班次中，RGV 和 CNC 进行各种相同操作所需时间相同；

5.RGV 对上料传送带一侧的上下料时间比下料传送带一侧的上下料时间短；

6.忽略 RGV 从接收指令到采取行动所需时间；

7.RGV 对一个 CNC 进行上下料时，生料的上料开始时间与熟料的下料开始

时间为同一时刻；

8.对于可能发生故障的情况，我们将故障的发生概率定为 1%，这里的 1%我们定义为频率概率，即 CNC 每进行 100 个工序的加工，会在一个工序中的时候发生故障，且每次加工时发生故障的概率相等。考虑到 CNC 的加工时间比等待时间和上下料时间长得多，我们忽略 CNC 在未加工状态下故障的情况。

9.对于需要双工序加工的工件，我们假设当 RGV 对完成一道工序的 CNC 完成上下料操作后，接下来必须去寻找到另一个已经完成第二道工序（或者空闲的用于加工第二道工序）的 CNC，而不能进行其他操作。

3、符号说明

表 3.1 主要符号变量及说明

符号	说明
t_{m_i}	RGV 移动i个单位所需的时间$(i = 1,2,3)$
t_{11}	CNC 加工完成一个一道工序的物料所需时间
t_{21}	CNC 加工完成一个两道工序物料的第一道工序所需时间
t_{22}	CNC 加工完成一个两道工序物料的第二道工序所需时间
t_c	RGV 为 CNC（奇数编号或偶数编号）进行一次上下料所用的时间
t_o	RGV 为奇数编号 CNC 进行一次上下料所需时间
t_e	RGV 为偶数编号 CNC 进行一次上下料所需时间
t_w	RGV 完成一个物料的清洗作业所需的时间
n_{ij}	奇数编号$(i = 2)$或偶数编号$(i = 1)$完成第j道工序的 CNC 的数目
t_{bi}	第i个 CNC 故障后故障排除所需时间
N_i	策略i对应一班次内生产的成料总数
t_{f_i}	编号为i的 CNC 在一个班次的时间内的总空闲时间

η_i	策略i对应一班次内系统的作业效率

【注】没有具体列出的符号，我们将在文章第一次出现时给出具体的说明。

4、模型的分析和建立

题目给出的三种具体情况可以分为单工序、双工序的调度算法以及故障处理三部分，于是我们分别对不同情况下的 RGV 运行的策略进行了初步分析。首先总体来看，无论是对需要一道工序或者是两道工序物料的加工，我们最终的目标都是要在一定的时间（给定的一班次 8 小时）内，生产出尽量多的工件。为此我们需要尽量减少 CNC 的等待（即 CNC 完成一个物料的加工作业后，RGV 没能即刻到达该 CNC 所在的位置为其上下料）时间，使得 CNC 处于作业的时间尽量长。在加工物料时间近似不变的情况下，即可得到尽量多的加工成熟的工件数。

4.1 单工序下 RGV 的调度算法

4.1.1 完全队列调度策略

首先我们考虑一道工序且 CNC 不发生故障这一最简单的物料加工作业情况。在这种情况下，对 RGV 最简单的调度策略即为每接收一个空闲状态的 CNC 发出的上料需求信号，若 RGV 处于停止等待状态，则立刻前往该 CNC 为其上下料。也就是说，将发出上料需求信号的 CNC 存储到一个等待队列中，并按照其进入队列的顺序进行访问。我们将这种算法称为“完全队列调度策略”。

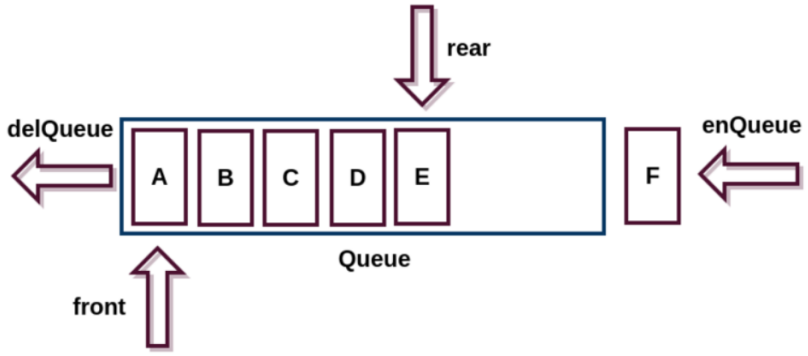

图 4.1 队列说明

4.1.2 一呼百应调度策略

完全队列策略在实现上固然简单，但我们可以很容易想到它在某些情况上的漏洞。由于此时没有对入队的各个 CNC 与 RGV 的距离进行度量，倘若在 RGV 前往当前队列第一位的 CNC 的途中，恰好距其更近的 CNC（甚至此时两者位于

同一位置）向 RGV 发出上料需求信号，这个时候明显先为更近的 CNC 进行上下料操作，比为当前队列第一位的 CNC 上下料效率更高。考虑到 8 个小时足够完成约上百组的成料加工，这种情况应该会占一定的比例，因此我们有必要将这种情况考虑进去。这里我们将其称为“一呼百应调度策略”。

我们将寻路算法中常用的 Dijkstra 算法[1]进行了修改，在完全队列的调度策略中，我们利用一个简单队列实现对 CNC 上下料顺序和时间的规划。而在一呼百应策略中，相当于用一个定时更新的最小优先队列来管理 CNC 调度顺序，例如 1 号 CNC 呼叫可能使 RGV 前往其他 CNC 进行上下料作业，即启发式的调度算法思想，每次寻找相对最优的解。在这个启发式的调度算法中，启发项即为对每个 CNC 的启发函数$H_1(n)$，最小优先队列即为存放启发函数的定时更新队列：

$$H_1(n) = \ t_{mi} + t_c + t_w \tag{4.1}$$

$H_1(n)$表示 RGV 投入到 n 号 CNC 上下料和重获自由的时间差，因此如果 H(n)越小，说明 RGV 能用更短的时间完成一个 CNC 上下料，从而节省整个系统的无效等待时间。

下图为调度算法的示意图，当有一个 CNC 进行呼叫后，RGV 会更新最小优先队列，找到启发函数最小的 CNC 并前往进行上下料工作。

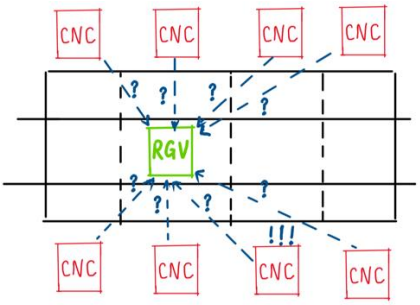

图 4.2 一呼百应调度策略示意图

4.1.3 感知预判调度策略

在以上两种调度中我们都是默认 RGV 本身无“记忆性”，也就是说，RGV 并不能“记住”它之前为那些 RGV 进行过上料，这也会对 RGV 的调度策略产生一定的影响。例如我们考虑一种较为极端的情况，由于 CNC 分别排列在 RGV 的两侧，倘若我们为同样位置一侧的 CNC 完成上下料一小段时间Δt $(t_{m_1} \leq \Delta t \leq t_{m_3})$ 之后，与其相同位置不同侧的 CNC 完成了对物料的加工，但 RGV 在收到该 CNC 的上料需求信号后已经一定的距离，即使这时应用上面的“一呼百应调度算法”，也将相当一部分的时间浪费在了 RGV 的移动上。而由于这里我们假

[1] [1]Thomas H.Cormen,Charles E.Leiserson,Ronald L.Rivest,Clifford Stein,殷建平,徐云,王刚,刘晓光,苏明,邹恒明,王宏志.算法导论(原书第 3 版)[J].计算机教育,2013(10):51.

设对物料的加工时间是一定的，因此我们可以通过为 RGV 赋予“记忆性”让其能够预判这种情况从而能够提前移动到预定位置准备上下料，记为“感知预判调度策略”。

感知预判策略即对一呼百应策略中的启发项进行了修改，为了达到时间的重用率，即 RGV 的移动与 CNC 的加工同时进行，由时间的长板效应，我们将启发项中的移动时间修改为移动时间和 CNC 剩余加工时间的最大值：

$$H_2(n) = \max\{t_{m_i}, t_{left}\} + t_C + t_w \tag{4.2}$$

其中，t_{left}为 n 号 CNC 该次加工的剩余时间(如果加工完成则为 0)。通过取最大值，能够得知不同 CNC 的距真正开始上下料的时间，从而进一步节约了整个系统的时间。

下图为感知预判模型的示例图。在某个 CNC 将要但未完成当次加工时，RGV 先移动至该 CNC 位置进行准备，提高整个系统的加工效率。

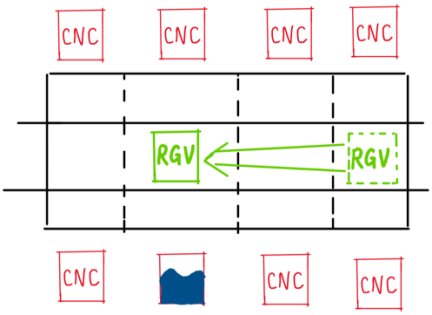

图 4.3 感知预判调度策略示意图

4.2 双工序下的 RGV 调度策略

由单工序的模型分析中可以看出，两次修正后的感知预判调度策略的时间利用率最高，而在双工序模式下，我们进一步将感知预判调度策略进行扩展。

由于两道工序的先后性和 CNC 刀具的不可更替性，我们需要对不同位置的 CNC 种类在加工开始前进行分配，在开始加工后，需要时刻关注 CNC 是否有下料[2]：

① 若第一道工序没有下料则 RGV 不继续前往第二道工序的 CNC，而是重新寻找第一道工序的呼叫状态 CNC；

② 若第二道工序没有下料则 RGV 不进行最后的清洗工件过程。

4.2.1 对执行两道工序的 CNC 分配

[2] [2]张桂琴,张仰森.直线往复式轨道自动导引车智能调度算法[J].计算机工程,2009,35(15):176-178+181.

根据题目附件中对 CNC 的说明，在对需要两道工序的物料进行加工时，两道工序所花费的时间并不相同，而且物料的第一道工序和第二道工序需要在不同的 CNC 上依次加工完成，因此我们需要对完成第一道工序和完成第二道工序的 CNC 进行分配。这其中包括对 CNC 数量和位置的分配。

① 对 CNC 数量的分配

根据题目附件中的条件，我们容易知道

$$\begin{cases} n_{11} + n_{12} \leq 4 \\ n_{21} + n_{22} \leq 4 \end{cases} \tag{4.3}$$

同时，若将第一道工序和第二道工序分开考虑，由于必须经过第一道工序加工的物料才能进入第二道工序进行加工，同时只有经过第二道工序加工的物料才能成为成料。这里考虑无缝理想加工的情况，根据短板效应，我们将优化目标确定为

$$\hat{Y} = \min\left\{\frac{8\times 3600}{n_{11}(t_{21}+t_e)} + \frac{8\times 3600}{n_{21}(t_{21}+t_o)}, \frac{8\times 3600}{n_{12}(t_{22}+t_e)} + \frac{8\times 3600}{n_{22}(t_{22}+t_o)}\right\} \tag{4.4}$$
$$s.t. \quad Y = \hat{Y}_{max}$$

即为在一个班次的时间内，令能够经过第一道工序加工后的熟料和经过第二道工序加工后的成料二者的最小值取到最大。

这样问题就转化为决策变量为离散型的非线性规划问题

这里有必要说明的是，虽然我们最终得到的成料数等于经过第二道工序加工并下料的物料数。但是在这一步优化中我们将第一道工序和第二道工序分开考虑，倘若将优化目标定为使完成第二道工序的物料数最大，很明显这将会使得优化模型趋向 8 个 CNC 全部为用于完成第二道工序的 CNC 这种极端情况，很明显这是不合理的。而且事实上，根据两道工序的关系，完成第一道工序的 CNC 与完成第二道工序的 CNC 相匹配应该是我们想要的结果。所以为了避免这种极端情况的发生，我们采用了(4.4)的优化目标，可以有效避免上述的极端情况。

在确定了系统的各个参数后，由于 CNC 的数量取值离散，我们可以通过遍历的方法得到该目标函数的优化结果。

② 对 CNC 位置的分配

在完成两道工序的 CNC 的数量分配确定以后，CNC 分配的可能情况就大大减少。倘若想要确定如何放置已有的 CNC 才能使得在一个班次内生产出的成料数目最多，我们只需要穷举出有限的几种情况（事实上无论 CNC 的数量是何种情况，在确定 CNC 数目后，位置的分配情况不会超过$\frac{4!}{2!\times 2!}\times\frac{4!}{2!\times 2!}=36$种），取

最终使得一班次内生产的成料数目最多的一种分配方式即可。

4.2.2 加工开始后两道工序的配合策略

在感知预判调度策略基础上，分析两道工序上下料时的策略。

1）第一道工序的上下料

在两道工序开始后，如果在第一道工序上下料后有下料工件产出，则去寻找并进行第二道工序；否则重新寻找新的第一道工序 CNC。

2）第二道工序的上下料

需在第一道工序有产出工件的前提下进行，且在第二道工序上下料后如果有下料工件产出，则进行清洗环节，之后即有一个完整的成品产出；若没有下料工件产出，则对第二道工序上料后结束，重新寻找第一道工序的 CNC。

总的来看，在两道工序加工过程中，由于前后工序的先后顺序，无法直接对第二道工序进行上下料，必须按照第一、第二道的顺序进行。而在进行感知预判调度策略时，我们采取防故障模式的改进感知预判调度策略，即对启发函数$H(n)$改变为二元函数$H(n,m)$，且在一二道工序之间增加一种专门寻找启发值最低的第二道工序 CNC 的策略，称其为“双层感知预判调度策略”。

双层感知预判调度策略具体阐述为，在 RGV 空闲时，RGV 只能去第一道工序的 CNC 进行上下料，判断去哪一个时，通过计算所有的第一道->第二道可能，找到启发值最低时间最短的第一道工序 CNC 与第二道工序 CNC 的组合，得到这个路径组合后，仅去组合中第一道工序 CNC 进行上下料，若有下料产出，则重新专门预判去哪一个第二道工序 CNC 的时间最短，再执行进行第二道工序上下料。

在这个调度策略中，用到了两类启发函数，其中一类是 RGV 将进行第一道工序上下料时对一二道组合的最优路径寻找，但仅执行第一道，舍弃第二道；之后重新计算时用仅计算当前位置与所有第二道 CNC 的启发值，用来寻找哪一个第二道 CNC 的时间消耗最小；总的来说第一类启发函数用来计算前往哪一个第一道 CNC，而第二类启发函数用来计算前往哪一个第二道 CNC。在计算第一类类启发函数时却也时常考虑到后续第二道 CNC 的情况。

第一类启发函数定义为：遍历所有一二道 CNC 组合，两道工序 CNC 分别记为i,j ,若 i 号 CNC 无工件，则启发函数仅为 RGV 与 i 号 CNC 的感知预判启发函数(不含清洗时间)；若 i 号有工件，则启发函数为 i 号 CNC 的感知预判启发函数（不含清洗时间）与从 i 到 j 的感知预判启发函数之和。

$$h_1(n,m)=\begin{cases}H_2(n)-t_c & status(n)=0\\ H_2(n)+H_2'(n,m)-t_C & status(n)\neq 0\end{cases} \tag{4.5}$$

其中$H_2'(n,m)$为 RGV 位于n位置时，位于m位置的执行第二道工序的 CNC 的启发函数；$status(n)=0$表示执行第一道工序的 CNC 上没有加工完成的物料。

第二类启发函数定义为：遍历所有第二道 CNC，启发函数为当前位置 RGV 到目标 CNC 的感知预判启发函数。

$$h_2(m)=H_2(m) \tag{4.6}$$

通过两类启发函数构成的双层感知预判策略，从而实现两道工序的智能化调度。

4.3 故障模型及处理

4.3.1 单工序故障及处理

① 完全队列调度策略的故障及处理

在完全队列调度策略中，各 CNC 是通过完成当前物料时向 RGV 发出上料需求信号完成入队。而在我们的假设中，我们认为 CNC 的故障均发生在加工过程中。这也就是说，倘若发生故障，这时 CNC 并没有完成当前物料的加工，也就不会向 RGV 发出上料信号。因此在完全队列调度策略中，当 CNC 发生故障时，我们认为在该台 CNC 故障期间，除了该故障 CNC 以外的 CNC 参与正常入队。

在该 CNC 的故障修复之后，它会回归初始状态，立刻向 RGV 发出上料需求信号，即立刻入队。

② 一呼百应调度策略和感知预判调度策略的故障及处理

对于一呼百应调度策略和感知预判调度策略，决定 RGV 每次调度的是启发函数h_i，在处于停止等待状态时，RGV 会选择向当前启发函数值最小的一项移动。因此一旦有 CNC 发生故障，故障的 CNC 就会向 RGV 发出信号，将对应 CNC 的启发函数赋给无穷大的标称值（具体为将$H_1(n)$或$H_2(n)$赋予无穷大的标称值），这样除非所有 CNC 都发生故障（这种情况概率非常小，但在实际情况中仍有可能发生，为此我们设置这种情况下 RGV 处于停止等待状态），否则该 CNC 对应的启发函数值一定不会是最小值，RGV 也就不会前往该 CNC 所在位置进行上下料。

在该 CNC 的故障修复之后，它回到初始状态，同时向 RGV 发出上料需求信号，RGV 立刻重新计算它的启发函数值。

4.3.2 双工序的故障及处理

双工序的故障及处理与单工序相比更为复杂。由于 CNC 的类型增加为两种，

即两种 CNC 都会发生故障。而每一个物料都需要经过两道工序才能成为成料，由此可知，相比于单工序的故障，尽管每个 CNC 的故障率仍为 1%，实际上对于一个物料来说，其遭遇故障导致报废的概率增大为 1.99%。

根据单工序感知预判调度策略的故障处理的方法，RGV 会将故障 CNC 对应的启发函数值更新为无穷大的标称值。具体为如果执行第一道工序的 CNC 发生故障，则将公式(4.5)所示的$h_1(n,m)$赋予无穷大的标称值；如果执行第二道工序的 CNC 发生故障，则将公式(4.5)所示的$h_2(m)$赋予无穷大的标称值。在故障修复后，CNC 回到初始状态，其在双层感知预判调度策略中的启发函数也回到了空状态。对于$h_1(n,m)$即回到了$status(n)=0$的状态。

5、模型的求解和结果分析

我们给出了模型求解的主要流程图，全程通过 Python 实现模型的最终求解。为了更方便直观的说明问题，我们首先给出模型代入第一组参数的检验结果。在本部分的最后给出所有参数模型的结果。

5.1 模型的求解流程

5.1.1 单工序完全队列调度

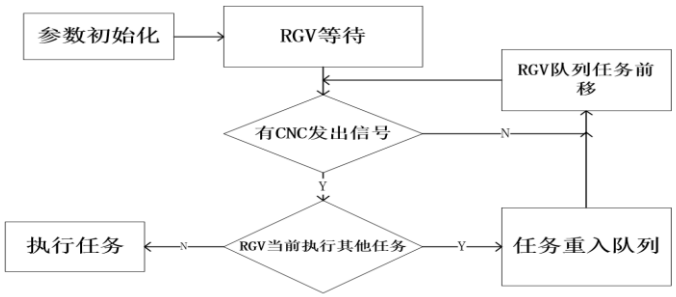

图 5.1 单工序完全队列调度流程图

在完全队列调度中，RGV 只有在 CNC 发出信号后按照发出信号的先后次序前往进行上下料工作。

5.1.2 单工序一呼百应调度策略

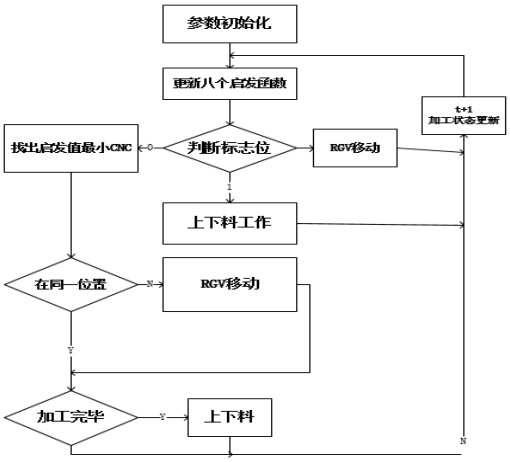

图 5.2 单工序一呼百应调度策略流程图

在一呼百应策略中，判断标志位的条件中，只有某个 CNC 发出呼叫信号后 RGV 才可以进行移动和工作。即 CNC 发出呼叫信号后，RGV 进行启发函数值队列的更新，找到最小值，并前往进行上下料工作。

5.1.3 单工序感知预判调度策

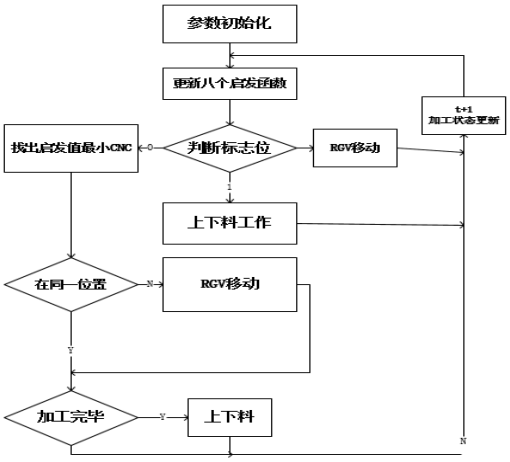

图 5.3 单工序感知预判调度策略流程图

判断感知调度策略的流程图与一呼百应策略看起来一样，但有两点不同：首先是判断标志位的条件有区别，预判感知策略中 RGV 根据“记忆性”，能够预先移动到将要完成加工的 CNC 位置上；其次是预判感知策略中启发函数的定义不同，考虑到 RGV 移动时间与工件当次加工的剩余时间同时进行。这两点使得 RGV 按照预判感知调度策略能够进行效率更高的工作。

5.1.4 双工序 CNC 的数量及位置分配求解

由(4.3)、(4.4)确定的非线性优化模型得到三组参数下 CNC 的数量分配。按照确定的数量遍历所有位置分配得到的一个班次内的结果，选取其中数量最大的一组分配作为 CNC 的分配结果。

求解后得到的 CNC 位置与数量分配如下图。

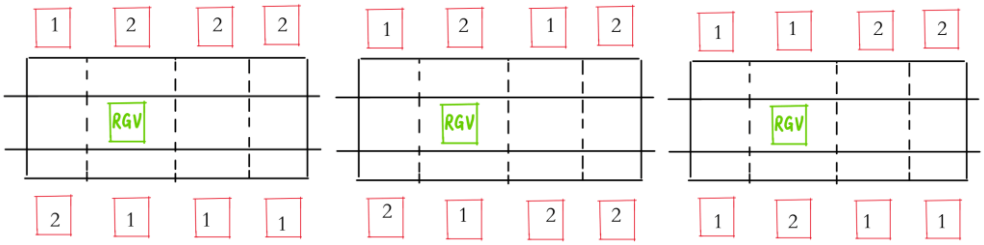

图 5.4 三组 CNC 分配（从左至右依次为一二三组）

5.1.5 双工序的双层感知预判调度策略

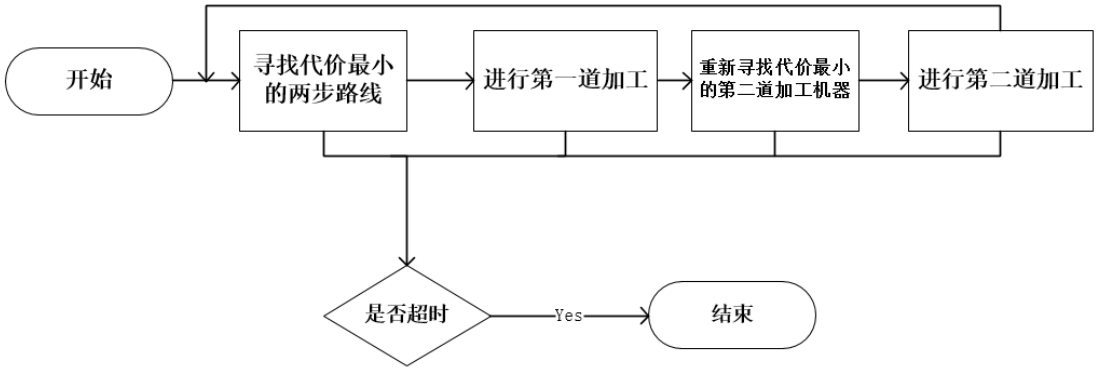

图 5.5 双层感知预判调度策略流程图

双层感知预判调度策略是从单工序的感知预判调度策略的基础上进行扩展。考虑到双工序的先后顺序和故障的可能性，我们采取更加独立的判断策略。在 RGV 寻找第一道 CNC 时，寻找的是启发值最小的两步路线，移动至第一个完成上下料后，如果该 CNC 有下料，则再次利用预判感知策略寻找合适的第二道 CNC。双层感知预判调度能够有效地兼容故障等问题，在第一道 CNC 发生故障或第二道发生故障时能够有效地重新选择最佳机器防止死等。

5.3 模型产量结果与性能分析

对于无故障情况，每一种策略结果不会改变，而对于有故障情况，我们将其运行 100 次作序列及平均值，将有故障与无故障进行对比，同时也将单工序中的三种策略、双工序中两道工序之间横向对比。

5.2.1 单工序的三种策略比较

表 5.1 三种策略单工序产量比较

	无故障	有故障(故障率 1%)
完全队列策略	340	283.1386

一呼百应策略	344	311.0495
感知预判策略	368	325.4185

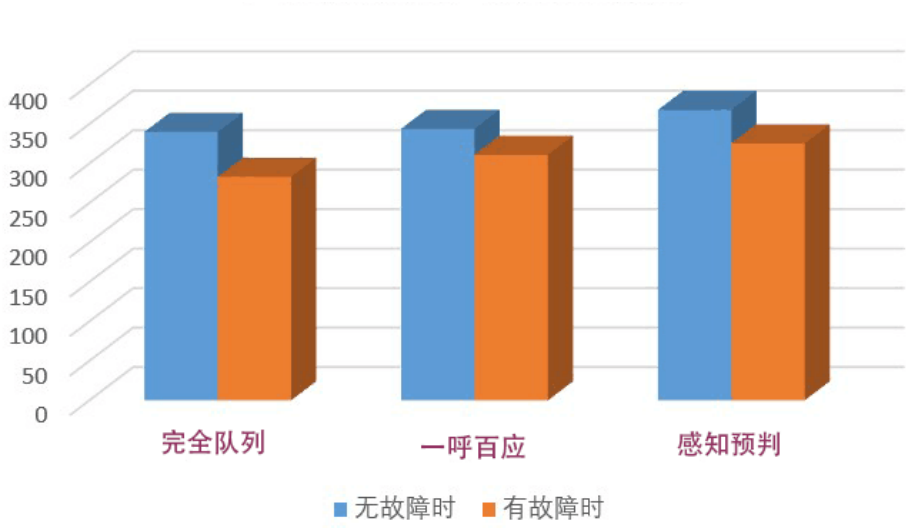

图 5.6 三种策略生产总成料数对比

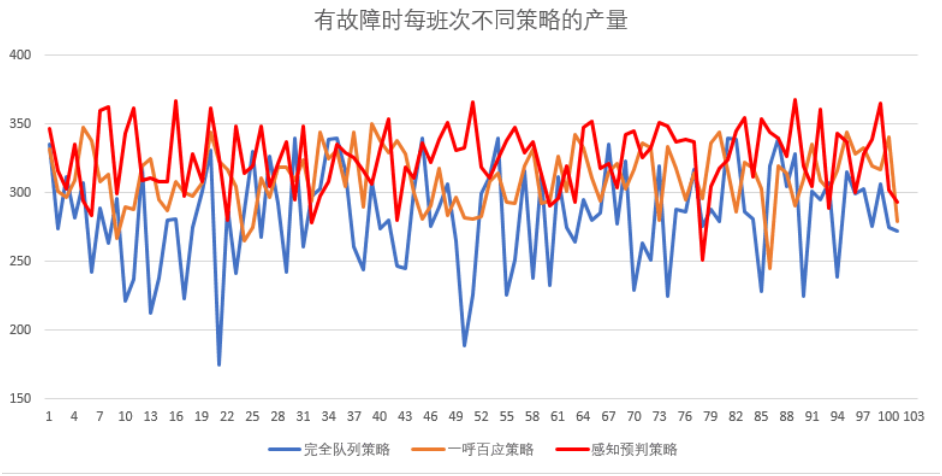

图 5.7 有故障时每班次不同策略的产量

在无故障的理想情况下，可以看到产量关系为：感知预判>一呼百应>完全队列。为了检验算法鲁棒性，在同等故障率情况下，三种算法结果的平均值大小关系保持不变，感知预判策略的效果良好。

5.2.2 双工序中的两类 CNC 比较

表 5.2 双工序中两类 CNC 比较

	无故障	有故障(故障率 1%)
第一道工序	242	240.2
第二道工序	237	233.496

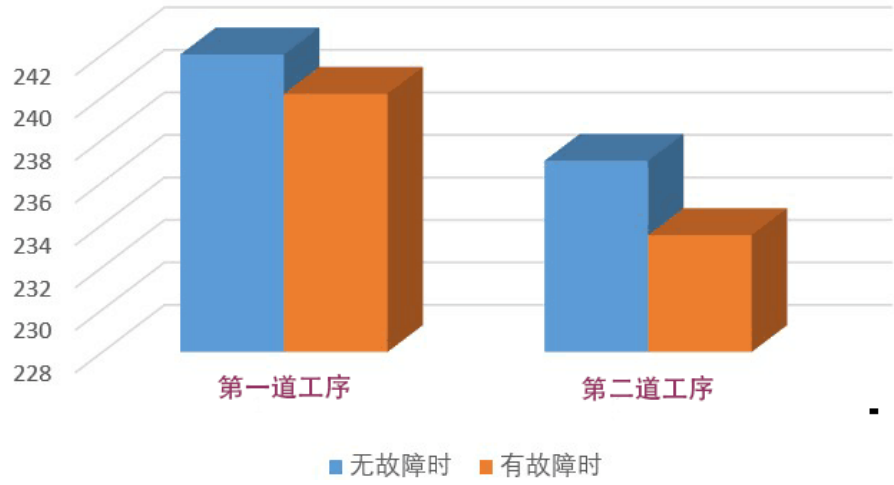

图 5.8 第一道工序和第二道工序总产量对比

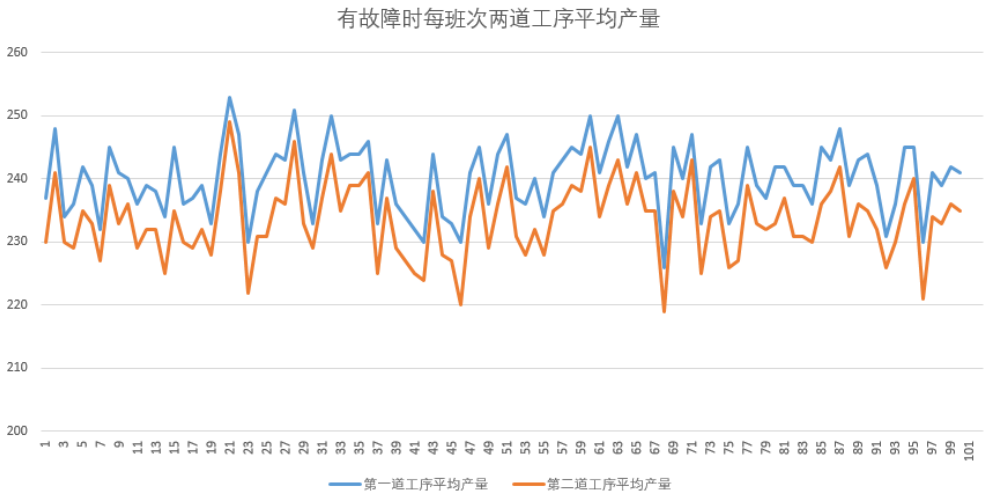

图 5.9 有故障时每班次两道工序平均产量

在双工序的产量分析中，每个工件需要两道工序且有先后关系，所以无论是否有故障，第二道工序的产出量小于第一道工序产出量，即在当天班次截止时，仍有工件在工作，完成了第一道工序但未完成第二道工序，与结果符合度很好。

而有故障时，由于更多的工件因故障而未完成第二道工序，故两道工序的产量差值会更大。从结果来看，第二道工序与第一道工序的产量差距相对不大，也明显体现了双工序在加工前对机器配置和加工后的双层感知预判策略的优势。

5.3 模型效率结果与性能分析

在效率分析时，我们定义效率为：

$$\eta = 1 - \frac{t_f}{t_{all}} \tag{5.1}$$

t_{all} 为一班次的工作时间，呼叫 RGV 后等待的时间越久，效率越低，我们

对有无故障、不同策略情况下分别计算效率进行对比。

5.3.1 单工序的三种策略比较

表 5.3 单工序三种策略效率对比

	无故障	有故障(故障率 1%)
完全队列策略	92%	85.57%
一呼百应策略	92%	93.50%
感知预判策略	99%	98.97%

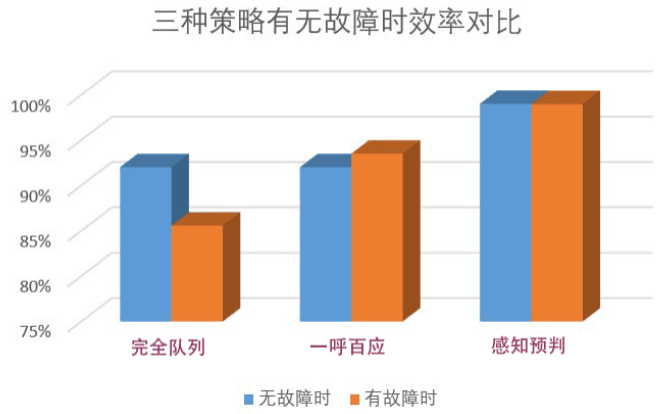

图 5.10 单工序三种策略有无故障效率对比（柱形图）

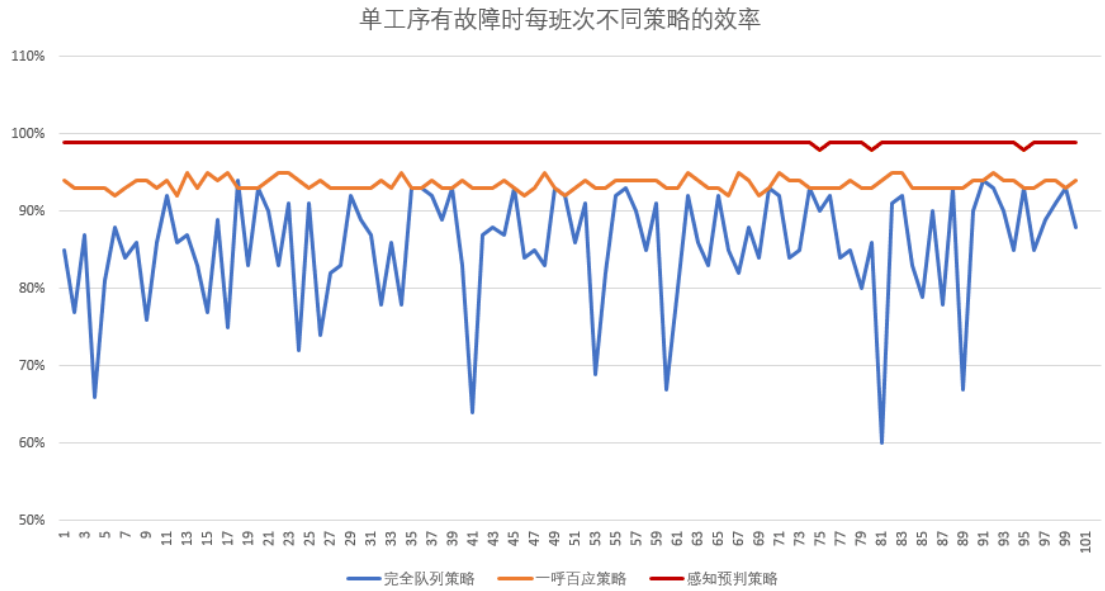

图 5.11 单工序有故障时每班次不同策略的效率

从理想情况看，在无故障情况下，感知预判的效率接近百分之百，相比于感知预判策略有十足的进步，而一呼百应较完全队列策略的进步不明显。

但从鲁棒性来看，在有故障情况下，完全队列策略效率有明显降低，而采取一呼百应策略的效率反而升高，这是由于通过调度的改进，在有故障发生时，其他呼叫状态的 CNC 更有机会被快速处理，从而使得效率不降反增。而感知预判策略的鲁棒性很好，没有明显降低，说明感知预判策略相对实用性比较强。

5.3.2 双工序中的两类 CNC 比较

表 5.4 双工序中的两类 CNC 对比

	无故障	有故障 (故障率 1%)
第一道工序	93%	92.54%
第二道工序	91%	90.74%

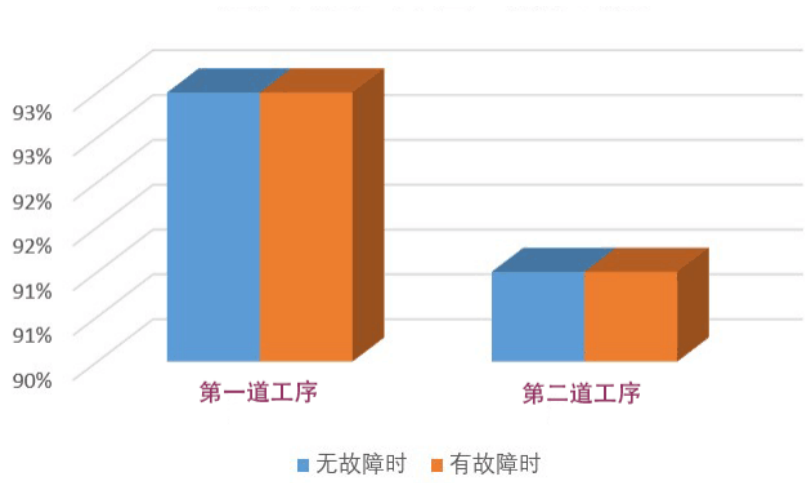

图 5.12 第一道工序和第二道工序有无故障时的效率对比

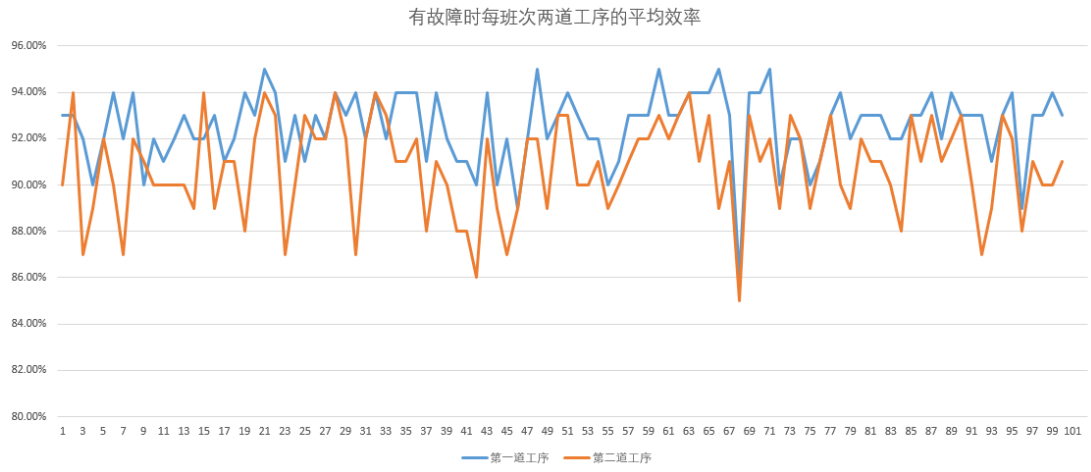

图 5.13 有故障时每班次两道工序的平均效率

而在双工序情况下，我们采取的是双层感知预判策略，对两道工序的等待时

间采取分别取平均值进行计算处理。发现第二道工序始终小于第一道工序的效率，这是由于在第二道 CNC 完成好后，如果没有第一道 CNC 的下料，RGV 不会先去处理第二道 CNC，否则将没有半成品带给 CNC 导致其等待时间更长，还可能耽误整个系统的运作。同时注意到双层感知预判策略的效率鲁棒性较好，故障对效率影响不大。

5.3.3 模型极限分析

① 单工序

首先我们分析单工序情况下的一个班次内加工成料理论极限值。在不考虑 CNC 故障的情况下，假设 8 台 CNC 有 8 台 RGV 为其上下料，保证其一直处于加工或更换物料的状态，同时计入 RGV 为最后一个熟料的清洗的时间。则

$$N_{1i} = \frac{(8 \times 3600 - t_w) \times 4}{(t_{11} + t_o)} + \frac{(8 \times 3600 - t_w) \times 4}{(t_{11} + t_e)} \tag{5.2}$$

代入各组参数可得对应的一个班次内加工成料的理论极限值，如下表所示。

表 5.5 单工序理论极限成料数与实际成料数对比

i	1	2	3
N_{0i}	390.5	375.8	400.7
N_{3i}	368	350	376

② 双工序

在双工序条件下，限制一个班次内加工成料理论极限值的是 CNC 完成第一道工序时间与第二道工序时间的最大值。同样考虑的只有 CNC 的加工时间、更换物料时间和 RGV 清洗最后一个熟料的时间。则有

$$N_{2i} = \min\left\{\sum_{c=o,e} \frac{8 \times 3600}{t_{21} + t_c}, \sum_{c=o,e} \frac{8 \times 3600 - t_w}{t_{22} + t_c}\right\} \tag{5.3}$$

代入各组参组参数可得对应的一个班次内加工成料的理论极限值，如下表所示。

表 5.6 双工序理论极限成料数与实际成料数对比

i	1	2	3
N_{2i}	268.7	270.4	297.5
N_{1i}	249	194	235

我们可以看出，由于第二组参数 CNC 数目和位置采用了不均匀分配的方法，使得可以加工成料可以达到的理论极限值大幅提高，从而达到提高实际产量的目

的。

我们将极限与结果对比，发现第一组的产率很高。第二组和第三组略有下降，观察第二组和第三组系统的参数我们可以得知，第二组和第三组在双工序的第一道工序和第二道工序的加工时间都有较大差距。这时我们输出第二组和第三组的各个 CNC 的等待时间，如下图。

```
235
>>> waittime
[5397, 5063, 5623, 16896, 5486, 6052, 5474, 5570]
```

图 5.14 第三组各 CNC 等待时间

由此可知，位于边缘的 8 号 CNC（执行第二道工序）等待时间过长。因此我们将第三组对应的 8 号 CNC 改为执行第一道工序的 CNC，并输出等待时间，得到结果如下图。

```
224
>>> waittime
[27799, 836, 1364, 28110, 1433, 1909, 1179, 1393]
```

图 5.15 第三组更改 CNC 分配后之后各 CNC 等待时间

由图可知，这样虽然使得仅剩的两个 CNC 的等待时间减少，但由于执行第一道工序的 CNC 过多，导致两台执行第一道工序的 CNC 几乎没有工作，使得总体效率反而降低。对第二组也进行类似推测，同样有产量变低的情况出现。

```
194
>>> waittime
[8956, 2431, 4529, 11415, 2861, 9469, 2819, 10744]
```

图 5.16 第二组 CNC 等待时间

```
191
>>> waittime
[26288, 1375, 5547, 1460, 1707, 5817, 1105, 12863]
```

图 5.17 第二组更改 CNC 分配之后各 CNC 等待时间

因此我们可以推测，这是由于当前系统的 CNC 数目过少导致无法逼近最优解。当系统中的 CNC 数目足够多时，我们可以找到更合适的一组分配，使得调度策略更接近连续情况下的最优解。

5.4 其他参数结果

以上是按照题目中第一组参数所得的结果和性能进行分析，篇幅所限，下面仅给出另两组对应的性能计算数据结果，可以看出上述各项结论均很适用。

5.4.1 第二组参数结果

表 5.7 第二组参数单工序结果

	无故障	无故障	有故障	有故障
	产量	效率	产量	效率

完全队列策略	295	84%	289	86%
一呼百应策略	322	91%	256	94%
感知预判策略	350	99%	315	98%

其中有故障的结果是 100 次运行结果的平均值。

表 5.8 第二组参数双工序结果

	产量	**效率**
第一道 CNC 平均	200	73%
第二道 CNC 平均	194	79%

5.4.3 第三组参数结果

表 5.9 第三组参数单工序结果

	无故障	**无故障**	**有故障**	**有故障**
	产量	效率	产量	效率
完全队列策略	347	92%	275	80%
一呼百应策略	352	92%	349	93%
感知预判策略	376	99%	329	99%

其中有故障的结果是 100 次运行结果的平均值。

表 5.10 第三组参数双工序结果

	产量	**效率**
第一道 CNC 平均	238	79%
第二道 CNC 平均	235	64%

6、模型的评价与改进

6.1 模型的优点

6.1.1 单工序的感知预判调度策略

1. 基于启发式算法，按照时间复用的基本思想，尽量减少 RGV 和 CNC 的空闲时间。

2. 模型算法易于理解，时间复杂度低，对不同参数的系统适应性强。

3. 相较于完全队列和一呼百应策略，能够提高整个系统的效率，同时鲁棒性强，在有故障发生时的平均效率影响很低。

4. 计算量很低，相较于粒子群算法等需要大量计算的算法而言，感知预判调度通过启发函数值的最小优先队列，实现低时间复杂度的寻找相对最优路径，从结果看，效果良好。

6.1.2 双工序的 CNC 配置策略

1. 数量分配上，采用时间的短板效应分析，选取理论上连续不间断工作能使产量最大的一二道机器数量分配，在大量生产中更具有理论依据。

2. 位置分配上，在数量分配后，遍历所有不同的可能性，在无故障情况下分别试验，选取产量最高的位置分配，在故障率低时，使对应的产率尽可能接近最大值。

通过先数量分配找出科学合理的数量分配比，之后在此更加有限的所有组合中遍历理想实验，取结果最好的具体分配情况。

6.1.3 双工序的双层感知预判策略

从单工序的感知预判调度策略出发，考虑两道工序的依赖性，两道工序没有下料分别的影响和给启发函数计算上带来的影响。同时采用双层感知预判策略，能够减少第二道故障对第一道 CNC 选取的影响。

6.2 模型的缺点

6.2.1 单工序的感知预判调度策略

本文设计的感知预判调度策略适用于单 RGV 移动，如果轨道有多个 RGV 移动上下料，则需要进一步改进。同时本文的感知预判调度需要维护一个最小优先队列，如果系统中不只有 8 个 CNC，而是含有大量 CNC，则需要对计算启发函数、更新函数值速度做出一定的优化。

6.2.2 双工序的 CNC 分配策略

我们的分配策略分两步进行，对于第一步的计算短板效应而言，如果 RGV 移动时间等相较过大，会对计算结果产生一定不确定性影响，对于第二步而言，如果故障率很高，做出位置的预实验的意义不明显。

同时在双工序和总 CNC 数量不高的情况下，由于只能分配整数个机器，参数对结果影响很大，在特殊参数下，遍历所有分配策略，也没有理想的产量和效率，这是由于一二道机器数量和时间的配对不均匀导致的。

6.3 模型的总结

本文的目的是设计一种 RGV 的调度策略，我们经过查阅资料发现粒子群算法等计算量大的算法对于这种应用场景在工厂的设备并不是非常实用的解决方案，通过我们自己模拟假设场景和遇到的各类影响效率的情况进行统一优化，我们借鉴了 Dijkstra 等启发式寻路算法和最简单的队列数据结构，在完全队列模型上提出了启发式的一呼百应策略，又考虑到 RGV 移动时间和 CNC 加工时间的复用会使得系统效率提高，于是提出感知预判模型。在双工序中，首先我们利用概率统计和暴力求解法结合手段快速确定出最佳的 CNC 数量和位置分配，之后对单工序的感知预判策略做出扩展，提出适用于双工序的双层感知预测策略。对于故障的发生和解决，上面的几个模型受故障的影响均进行了测试和结果分析，发现感知预测策略的解法鲁棒性较高，在参数理想时结果很好，参数不理想时能够达到相对最佳的结果。

同时我们所有的模型均通过 python 代码实现，没有调用任何复杂的算法包，通过我们讨论各类情况的弊端和解决方案、调度算法设计，搭建并实现了有无故障、CNC 分配和各类调度算法设计。

七、参考文献

[1]Thomas H.Cormen,Charles E.Leiserson,Ronald L.Rivest,Clifford Stein,殷建平,徐云,王刚,刘晓光,苏明,邹恒明,王宏志.算法导论(原书第 3 版)[J].计算机教育,2013(10):51.

[2]张桂琴,张仰森.直线往复式轨道自动导引车智能调度算法[J].计算机工程,2009,35(15):176-178+181.

八、附录

1. 单工序无故障完全队列策略

```
import random
import csv
import math
import numpy as np
file = 'c1_1.csv'
fn = open(file,"w",newline = '')
writer= csv.writer(fn)
out = []
#c.index(sorted(c)[2])
#参数
move = [20,33,46]
process = 560 #一道工序
change =[31,28] #上下料时间，偶，奇
clean = 25

product_num = 0

yingshe = {0:2,1:4,2:6,3:8,4:1,5:3,6:5,7:7}

#故障发生函数：
def guzhang():
    temptemp = random.randint(1,100)
    if temptemp == 1:
        return 1
    else:
        return 0
#故障数量
gz_num = 0
def weixiu():
```

```
    tempp = random.randint(10*60,20*60)
    return tempp

#计算移动时间：
def movetime(i,j):
    if i == j:
        return 0
    else:
##      print(i)
##      print(j)
        return move[int(math.fabs(i-j)-1)]

#当前位置 0 1 2 3
loc = 0

#消耗函数
def xiaohao(status,time):
    xiao = [0] * 8
    for i in range(8):
        if status[i] == 0 : # 空
            xiao[i] = movetime(loc,i%4) + change[int(i/4)]
        elif status[i] == 2:
            xiao[i] = movetime(loc,i%4) + change[int(i/4)] +
clean
        elif status[i] == 1:
            xiao[i] = max(movetime(loc,i%4),process - time[i]) +
change[int(i/4)] + clean
        else:
            xiao[i] = 10000
    return xiao
# 得到消耗最小的机器编号
def get_least(xiao):
##    for i in range(len(xiao)):
```

```
##       if CNC_status[xiao.index(sorted(xiao)[0])]
    return xiao.index(sorted(xiao)[0])



# 八个机器的状态码，0表示空，1表示正在加工，2表示呼叫抓手,3表示故障，4表示待上下料
CNC_status = [0]*8
# 八个机器的运行时间
CNC_time = [0] * 8
#八个机器的当前运行的工件编号
current_num = [0]*8
#总时间
alltime = 8*3600
#故障发生
breaktime = [0] *8
#回复时间
overtime = [0] *8
#等待时间
waittime = [0] *8

RGV_move = 0
RGV_time=0
RGV_work = 0

last_least = -1
last_work = -1
t = 0
while t <= alltime:
##    print("loc:")
##    print(loc)
    #消耗最小的机器编号
    least = get_least(xiaohao(CNC_status,CNC_time))
```

```
##    print(least)
##    print(xiaohao(CNC_status,CNC_time))
#    print(loc)
    #如果需要移动,且未在工作和移动！！：
    if  loc != least % 4 and RGV_move == 0 and RGV_work == 0:
        #移动到对应位置
        target = least % 4
        delta_t = movetime(loc,target)
        RGV_time = t+delta_t
        RGV_move = 1
        last_least = least

    if RGV_move == 1 and t >= RGV_time:
        loc = last_least % 4
        RGV_move = 0  #停止移动

    if RGV_work ==1 and t>=RGV_time:
        RGV_work = 0
        CNC_status[last_work] = 1
        CNC_time[last_work] = 0

#执行更替,一开始或呼叫状态时执行
    if RGV_work == 0 and RGV_move == 0 and loc == least % 4 and
(CNC_status[least] == 2 or CNC_status[least] == 0):
        t0 = t
        #更替时间
        delta_t = change[int(least/4)]
        if CNC_status[least] == 2:
            delta_t += clean
            RGV_work = 1
            RGV_time = t + delta_t
            #完成的工作台
            print(least,end = ' : ')
```

```
            print("下料开始",end =',')
            print(t0,end = ',')
            #打印加工后的当前工件编号
            print(current_num[least])
            out[current_num[least]-1].append(t0)

        elif CNC_status[least] == 0:
            RGV_work = 1
            RGV_time = delta_t + t

        #放入新工件
        product_num += 1
        print(least,end = ' : ')
        print("上料开始",end=',')
        print(t0,end=',')
        print(product_num)
        out.append([yingshe[least],t0])



        current_num[least] = product_num
        last_work  = least


        #状态更新
        CNC_status[least] = 4 #等待上下料完成
        CNC_time[least] = 0
##
##        #计算故障
##        gz = guzhang()
##        if gz == 1:
##            #真正加工后的多久开始故障
##            breaktime[least] = random.randint(1,process) + t
```

```
##            gz_num += 1
##            overtime[least] = weixiu() +breaktime + t
##
##


    #如果完成
    for i in range(8):
        if CNC_time[i] >= process and CNC_status[i] == 1:
            CNC_status[i] = 2 #呼叫
            CNC_time[i] = process
    # 时间流逝
    for i in range(8):
        if CNC_status[i] == 1:
            CNC_time[i] += 1
        if CNC_status[i] == 0 or CNC_status[i] == 2:
            waittime[i] += 1


    t += 1
for i in range(len(out)):
    if len(out[i]) == 3:
        writer.writerow([i+1,out[i][0],out[i][1],out[i][2]])
    else:
        writer.writerow([i+1,out[i][0],out[i][1]])
fn.close()
```

2. 单工序无故障一呼百应策略

```
import random
import csv
import math
import numpy as np
##file = 'c1_1.csv'
##fn = open(file,"w",newline = '')
```

```
##writer= csv.writer(fn)
out = []


#c.index(sorted(c)[2])
#参数
move = [20,33,46]
process = 560 #一道工序
change =[31,28] #上下料时间，偶，奇
clean = 25

product_num = 0

yingshe = {0:2,1:4,2:6,3:8,4:1,5:3,6:5,7:7}

#故障发生函数：
def guzhang():
    temptemp = random.randint(1,100)
    if temptemp == 1:
        return 1
    else:
        return 0
#故障数量
gz_num = 0
def weixiu():
    tempp = random.randint(10*60,20*60)
    return tempp

#计算移动时间：
def movetime(i,j):
    if i == j:
        return 0
    else:
```

```
##       print(i)
##       print(j)
       return move[int(math.fabs(i-j)-1)]

#当前位置 0 1 2 3
loc = 0

#消耗函数
def xiaohao(status,time):
    xiao = [0] * 8
    for i in range(8):
        if status[i] == 0 : # 空
            xiao[i] = movetime(loc,i%4) + change[int(i/4)]
        elif status[i] == 2:
            xiao[i] = movetime(loc,i%4) + change[int(i/4)] +
clean
        #elif status[i] == 1:
           # xiao[i] = movetime(loc,i%4) + change[int(i/4)] +
clean
        else:
            xiao[i] = 10000
    return xiao
# 得到消耗最小的机器编号
def get_least(xiao):
##    for i in range(len(xiao)):
##       if CNC_status[xiao.index(sorted(xiao)[0])]
    return xiao.index(sorted(xiao)[0])



# 八个机器的状态码，0表示空，1表示正在加工，2表示呼叫抓手,3表示故障，4表示
待上下料
CNC_status = [0]*8
```

```
# 八个机器的运行时间
CNC_time = [0] * 8
#八个机器的当前运行的工件编号
current_num = [0]*8
#总时间
alltime = 8*3600
#故障发生
breaktime = [0] *8
#回复时间
overtime = [0] *8
#等待时间
waittime = [0] *8

RGV_move = 0
RGV_time=0
RGV_work = 0

last_least = -1
last_work = -1
t = 0
#CNC 允许移动
move_CNC = 0
while t <= alltime:
##    print("loc:")
##    print(loc)
    #消耗最小的机器编号
    least = get_least(xiaohao(CNC_status,CNC_time))
##    print(least)
##    print(xiaohao(CNC_status,CNC_time))
#    print(loc)
    #如果需要移动,且未在工作和移动!!:
    if  loc != least % 4 and RGV_move == 0 and RGV_work == 0
and move_CNC == 1:
```

```
            #移动到对应位置
            target = least % 4
            delta_t = movetime(loc,target)
            RGV_time = t+delta_t
            RGV_move = 1
            last_least = least

        if RGV_move == 1 and t >= RGV_time:
            loc = last_least % 4
            RGV_move = 0  #停止移动
            move_CNC = 0
        if RGV_work ==1 and t>=RGV_time:
            RGV_work = 0
            CNC_status[last_work] = 1
            CNC_time[last_work] = 0

        if loc == least % 4 and move_CNC == 1:
            move_CNC = 0
    #执行更替,一开始或呼叫状态时执行
        if move_CNC == 0 and RGV_work == 0 and RGV_move == 0 and
    loc == least % 4 and (CNC_status[least] == 2 or
    CNC_status[least] == 0):
            t0 = t
            #更替时间
            delta_t = change[int(least/4)]
            if CNC_status[least] == 2:
                delta_t += clean
                RGV_work = 1
                RGV_time = t + delta_t
                #完成的工作台
                print(least,end = ' : ')
                print("下料开始",end =',')
                print(t0,end = ',')
```

```
            #打印加工后的当前工件编号
            print(current_num[least])
            out[current_num[least]-1].append(t0)

        elif CNC_status[least] == 0:
            RGV_work = 1
            RGV_time = delta_t + t

        #放入新工件
        product_num += 1
        print(least,end = ' : ')
        print("上料开始",end=',')
        print(t0,end=',')
        print(product_num)
        out.append([yingshe[least],t0])



        current_num[least] = product_num
        last_work  = least


        #状态更新
        CNC_status[least] = 4 #等待上下料完成
        CNC_time[least] = 0
##
##        #计算故障
##        gz = guzhang()
##        if gz == 1:
##            #真正加工后的多久开始故障
##            breaktime[least] = random.randint(1,process) + t
##            gz_num += 1
##            overtime[least] = weixiu() +breaktime + t
```

```
##
##

    #如果完成
    for i in range(8):
        if CNC_time[i] >= process and CNC_status[i] == 1:
            CNC_status[i] = 2 #呼叫
            CNC_time[i] = process


    # 时间流逝
    for i in range(8):
        if CNC_status[i] == 1:
            CNC_time[i] += 1
        if CNC_status[i] == 0 or CNC_status[i] == 2:
            waittime[i] += 1
            move_CNC = 1  #允许移动
    t += 1
##for i in range(len(out)):
##    if len(out[i]) == 3:
##        writer.writerow([i+1,out[i][0],out[i][1],out[i][2]])
##    else:
##        writer.writerow([i+1,out[i][0],out[i][1]])
##fn.close()
```

3. 单工序无故障感知预判策略

```
import random
import csv
import math
import numpy as np
file = 'c1_1.csv'
fn = open(file,"w",newline = '')
```

```
writer= csv.writer(fn)
out = []
#c.index(sorted(c)[2])
#参数
move = [20,33,46]
process = 560 #一道工序
change =[31,28] #上下料时间，偶，奇
clean = 25

product_num = 0

yingshe = {0:2,1:4,2:6,3:8,4:1,5:3,6:5,7:7}

#故障发生函数：
def guzhang():
    temptemp = random.randint(1,100)
    if temptemp == 1:
        return 1
    else:
        return 0
#故障数量
gz_num = 0
def weixiu():
    tempp = random.randint(10*60,20*60)
    return tempp

#计算移动时间：
def movetime(i,j):
    if i == j:
        return 0
    else:
##      print(i)
##      print(j)
```

```
        return move[int(math.fabs(i-j)-1)]

#当前位置 0 1 2 3
loc = 0

#消耗函数
def xiaohao(status,time):
    xiao = [0] * 8
    for i in range(8):
        if status[i] == 0 : # 空
            xiao[i] = movetime(loc,i%4) + change[int(i/4)]
        elif status[i] == 2:
            xiao[i] = movetime(loc,i%4) + change[int(i/4)] +
clean
        elif status[i] == 1:
            xiao[i] = max(movetime(loc,i%4),process - time[i]) +
change[int(i/4)] + clean
        else:
            xiao[i] = 10000
    return xiao
# 得到消耗最小的机器编号
def get_least(xiao):
##    for i in range(len(xiao)):
##        if CNC_status[xiao.index(sorted(xiao)[0])]
    return xiao.index(sorted(xiao)[0])



# 八个机器的状态码，0表示空，1表示正在加工，2表示呼叫抓手,3表示故障，4表示
待上下料
CNC_status = [0]*8
# 八个机器的运行时间
CNC_time = [0] * 8
```

```
#八个机器的当前运行的工件编号
current_num = [0]*8
#总时间
alltime = 8*3600
#故障发生
breaktime = [0] *8
#回复时间
overtime = [0] *8
#等待时间
waittime = [0] *8

RGV_move = 0
RGV_time=0
RGV_work = 0

last_least = -1
last_work = -1
t = 0
while t <= alltime:
##    print("loc:")
##    print(loc)
    #消耗最小的机器编号
    least = get_least(xiaohao(CNC_status,CNC_time))
##    print(least)
##    print(xiaohao(CNC_status,CNC_time))
#    print(loc)
    #如果需要移动,且未在工作和移动!!:
    if  loc != least % 4 and RGV_move == 0 and RGV_work == 0:
        #移动到对应位置
        target = least % 4
        delta_t = movetime(loc,target)
        RGV_time = t+delta_t
        RGV_move = 1
```

```
        last_least = least

    if RGV_move == 1 and t >= RGV_time:
        loc = last_least % 4
        RGV_move = 0  #停止移动

    if RGV_work ==1 and t>=RGV_time:
        RGV_work = 0
        CNC_status[last_work] = 1
        CNC_time[last_work] = 0

#执行更替,一开始或呼叫状态时执行
    if RGV_work == 0 and RGV_move == 0 and loc == least % 4 and
(CNC_status[least] == 2 or CNC_status[least] == 0):
        t0 = t
        #更替时间
        delta_t = change[int(least/4)]
        if CNC_status[least] == 2:
            delta_t += clean
            RGV_work = 1
            RGV_time = t + delta_t
            #完成的工作台
            print(least,end = ' : ')
            print("下料开始",end =',')
            print(t0,end = ',')
            #打印加工后的当前工件编号
            print(current_num[least])
            out[current_num[least]-1].append(t0)

        elif CNC_status[least] == 0:
            RGV_work = 1
            RGV_time = delta_t + t
```

```
            #放入新工件
            product_num += 1
            print(least,end = ' : ')
            print("上料开始",end=',')
            print(t0,end=',')
            print(product_num)
            out.append([yingshe[least],t0])



            current_num[least] = product_num
            last_work  = least


            #状态更新
            CNC_status[least] = 4 #等待上下料完成
            CNC_time[least] = 0
##
##            #计算故障
##            gz = guzhang()
##            if gz == 1:
##                #真正加工后的多久开始故障
##                breaktime[least] = random.randint(1,process) + t
##                gz_num += 1
##                overtime[least] = weixiu() +breaktime + t
##
##

        #如果完成
        for i in range(8):
            if CNC_time[i] >= process and CNC_status[i] == 1:
                CNC_status[i] = 2 #呼叫
                CNC_time[i] = process
```

```
    # 时间流逝
    for i in range(8):
        if CNC_status[i] == 1:
            CNC_time[i] += 1
        if CNC_status[i] == 0 or CNC_status[i] == 2:
            waittime[i] += 1

    t += 1
for i in range(len(out)):
    if len(out[i]) == 3:
        writer.writerow([i+1,out[i][0],out[i][1],out[i][2]])
    else:
        writer.writerow([i+1,out[i][0],out[i][1]])
fn.close()
```

4. 单工序有故障完全队列策略

```
import random
import csv
import math
import numpy as np
file = 'c2_1.csv'
file2 = 'c2_2.csv'
file3 = 'c2_3.csv'
fn = open(file,"w",newline = '')
fn2 = open(file2,'w',newline = '')
fn3 = open(file3,'w',newline = '')
writer3 = csv.writer(fn3)
writer2 = csv.writer(fn2)
writer = csv.writer(fn)
out = []
out2 = {}
#c.index(sorted(c)[2])
```

```
#参数
move = [20,33,46]
process = 560 #一道工序
change =[31,28] #上下料时间，偶，奇
clean = 25

product_num = 0

yingshe = {0:2,1:4,2:6,3:8,4:1,5:3,6:5,7:7}

#故障发生函数：
def guzhang():
    temptemp = random.randint(1,100)
    if temptemp == 1:
        return 1
    else:
        return 0
##    return 0
#故障数量
gz_num = 0
def weixiu():
    tempp = random.randint(10*60,20*60)
    return tempp

#计算移动时间：
def movetime(i,j):
    if i == j:
        return 0
    else:
##        print(i)
##        print(j)
        return move[int(math.fabs(i-j)-1)]
```

```
#当前位置 0 1 2 3
loc = 0

#消耗函数
def xiaohao(status,time):
    xiao = [0] * 8
    for i in range(8):
        if status[i] == 0 : # 空
            xiao[i] = movetime(loc,i%4) + change[int(i/4)]
        elif status[i] == 2:
            xiao[i] = movetime(loc,i%4) + change[int(i/4)] +
clean
        #elif status[i] == 1:
         #   xiao[i] = max(movetime(loc,i%4),process - time[i])
+ change[int(i/4)] + clean
        else:
            xiao[i] = 10000
    return xiao
# 得到消耗最小的机器编号
def get_least(xiao):
##    for i in range(len(xiao)):
##        if CNC_status[xiao.index(sorted(xiao)[0])]
    return xiao.index(sorted(xiao)[0])


for i in range(1):
    # 八个机器的状态码，0表示空，1表示正在加工，2表示呼叫抓手,3表示故障，4
表示待上下料
    CNC_status = [0]*8
    # 八个机器的运行时间
    CNC_time = [0] * 8
    #八个机器的当前运行的工件编号
    current_num = [0]*8
```

```
    #总时间
    alltime = 8*3600
    #故障发生
    breaktime = [0] *8
    #回复时间
    overtime = [0] *8
    #等待时间
    waittime = [0] *8

    RGV_move = 0
    RGV_time=0
    RGV_work = 0

    last_least = -1
    last_work = -1
    t = 0
    l = [0,1,2,3,4,5,6,7]
    move_CNC = 0

    t=0
    while t <= alltime:
    ##    print("loc:")
    ##    print(loc)
        #消耗最小的机器编号
        if len(l) > 0:
            least = l[0]
            ##print(least)
            ##print(xiaohao(CNC_status,CNC_time))
            #print(loc)
            #如果需要移动,且未在工作和移动！！：
            if  loc != least % 4 and RGV_move == 0 and RGV_work
== 0 and move_CNC == 1:
                #移动到对应位置
```

```
                target = least % 4
                delta_t = movetime(loc,target)
                RGV_time = t+delta_t
                RGV_move = 1
                last_least = least

            if RGV_move == 1 and t >= RGV_time:
                loc = last_least % 4
                RGV_move = 0  #停止移动

            if RGV_work ==1 and t>=RGV_time:
                RGV_work = 0
                CNC_status[last_work] = 1
                CNC_time[last_work] = 0
            if loc == least % 4 and move_CNC == 1:
                move_CNC = 0
        #执行更替,一开始或呼叫状态时执行
            if move_CNC == 0 and RGV_work == 0 and RGV_move == 0
and loc == least % 4 and (CNC_status[least] == 2 or
CNC_status[least] == 0):
                t0 = t
                #更替时间
                delta_t = change[int(least/4)]
                if CNC_status[least] == 2:
                    delta_t += clean
                    RGV_work = 1
                    RGV_time = t + delta_t
                    #完成的工作台
                    print(least,end = ' : ')
                    print("下料开始",end =',')
                    print(t0,end = ',')
                    #打印加工后的当前工件编号
                    print(current_num[least])
```

```
            out[current_num[least]-1].append(t0)

        elif CNC_status[least] == 0:
            RGV_work = 1
            RGV_time = delta_t + t

        #放入新工件
        product_num += 1
        print(least,end = ' : ')
        print("上料开始",end=',')
        print(t0,end=',')
        print(product_num)
        out.append([yingshe[least],t0])
        l.pop(0)



        current_num[least] = product_num
        last_work  = least


        #状态更新
        CNC_status[least] = 4 #等待上下料完成
        CNC_time[least] = 0


        #计算故障
        gz = guzhang()
        if gz == 1:
            #真正加工后的多久开始故障
            breaktime[least] = random.randint(1,process) +
t

            gz_num += 1
```

```
                overtime[least] = weixiu() +breaktime[least] +
t



        #如果完成
        for i in range(8):
            if CNC_time[i] >= process and CNC_status[i] == 1:
                CNC_status[i] = 2 #呼叫
                CNC_time[i] = process
                l.append(i)
        # 时间流逝
        for i in range(8):
            if CNC_status[i] == 1:
                CNC_time[i] += 1
            if CNC_status[i] == 0 or CNC_status[i] == 2:
                waittime[i] += 1
                move_CNC = 1  #允许移动
            #故障发生处理判别
            if t == breaktime[i] and CNC_status[i] ==1:
                CNC_status[i] = 3
                print(i,end=':')
                print("故障发生",end=',')
                print(t,end=',')
                print(current_num[i])
                writer3.writerow([i,'故障发生',t,current_num[i]])
                #故障的物料、CNC!!、开始时间
                out2[current_num[i]] = [yingshe[i],t]
            if CNC_status[i] == 3 and t>=overtime[i]:
                CNC_status[i]=0
                CNC_time[i] = 0
                print(i,end=':')
                print("故障解除",end=',')
```

```
                print(t)
                writer3.writerow([i,'故障解除',t])
                out2[current_num[i]].append(t)

        if t == alltime:
            print(least,end = ' : ')
            print("上料开始",end=',')
            print(t0,end=',')
            print(product_num)
            writer3.writerow([least,'上料开始',t0,product_num])
        t += 1

for i in range(len(out)):
    #仅输出有下料的
    if len(out[i]) == 3:
        writer.writerow([i+1,out[i][0],out[i][1],out[i][2]])

for i in out2:
    print(i)
    print(out2[i])
    if len(out2[i]) == 3:
        writer2.writerow([i,out2[i][0],out2[i][1]])
    else:
        writer2.writerow([i,out2[i][0],out2[i][1]])
fn.close()
fn2.close()
fn3.close()
```

5. 单工序有故障一呼百应策略

```
import random
import csv
import math
```

```
import numpy as np
file = 'c2_1.csv'
file2 = 'c2_2.csv'
fn = open(file,"w",newline = '')
fn2 = open(file2,'w',newline = '')
writer2 = csv.writer(fn2)
writer= csv.writer(fn)
out = []
out2 = {}
#c.index(sorted(c)[2])
#参数
move = [20,33,46]
process = 560 #一道工序
change =[31,28] #上下料时间，偶，奇
clean = 25

product_num = 0

yingshe = {0:2,1:4,2:6,3:8,4:1,5:3,6:5,7:7}

#故障发生函数：
def guzhang():
    temptemp = random.randint(1,100)
    if temptemp == 1:
        return 1
    else:
        return 0
#故障数量
gz_num = 0
def weixiu():
    tempp = random.randint(10*60,20*60)
    return tempp
```

```
#计算移动时间：
def movetime(i,j):
    if i == j:
        return 0
    else:
##      print(i)
##      print(j)
        return move[int(math.fabs(i-j)-1)]

#当前位置 0 1 2 3
loc = 0

#消耗函数
def xiaohao(status,time):
    xiao = [0] * 8
    for i in range(8):
        if status[i] == 0 : # 空
            xiao[i] = movetime(loc,i%4) + change[int(i/4)]
        elif status[i] == 2:
            xiao[i] = movetime(loc,i%4) + change[int(i/4)] +
clean
        #elif status[i] == 1:
         #   xiao[i] = max(movetime(loc,i%4),process - time[i])
+ change[int(i/4)] + clean
        else:
            xiao[i] = 10000
    return xiao
# 得到消耗最小的机器编号
def get_least(xiao):
##    for i in range(len(xiao)):
##       if CNC_status[xiao.index(sorted(xiao)[0])]
    return xiao.index(sorted(xiao)[0])
```

```
# 八个机器的状态码，0表示空，1表示正在加工，2表示呼叫抓手,3表示故障，4表示待上下料
CNC_status = [0]*8
# 八个机器的运行时间
CNC_time = [0] * 8
#八个机器的当前运行的工件编号
current_num = [0]*8
#总时间
alltime = 8*3600
#故障发生
breaktime = [0] *8
#回复时间
overtime = [0] *8
#等待时间
waittime = [0] *8

RGV_move = 0
RGV_time=0
RGV_work = 0

last_least = -1
last_work = -1
t = 0
move_CNC = 0
while t <= alltime:
##    print("loc:")
##    print(loc)
    #消耗最小的机器编号
    least = get_least(xiaohao(CNC_status,CNC_time))
##    print(least)
##    print(xiaohao(CNC_status,CNC_time))
```

```
#    print(loc)
    #如果需要移动,且未在工作和移动！！：
    if  loc != least % 4 and RGV_move == 0 and RGV_work == 0
and move_CNC == 1:
        #移动到对应位置
        target = least % 4
        delta_t = movetime(loc,target)
        RGV_time = t+delta_t
        RGV_move = 1
        last_least = least

    if RGV_move == 1 and t >= RGV_time:
        loc = last_least % 4
        RGV_move = 0  #停止移动

    if RGV_work ==1 and t>=RGV_time:
        RGV_work = 0
        CNC_status[last_work] = 1
        CNC_time[last_work] = 0
    if loc == least % 4 and move_CNC == 1:
        move_CNC = 0
#执行更替,一开始或呼叫状态时执行
    if move_CNC == 0 and RGV_work == 0 and RGV_move == 0 and
loc == least % 4 and (CNC_status[least] == 2 or
CNC_status[least] == 0):
        t0 = t
        #更替时间
        delta_t = change[int(least/4)]
        if CNC_status[least] == 2:
            delta_t += clean
            RGV_work = 1
            RGV_time = t + delta_t
            #完成的工作台
```

```
        print(least,end = ' : ')
        print("下料开始",end =',')
        print(t0,end = ',')
        #打印加工后的当前工件编号
        print(current_num[least])
        out[current_num[least]-1].append(t0)

    elif CNC_status[least] == 0:
        RGV_work = 1
        RGV_time = delta_t + t

    #放入新工件
    product_num += 1
    print(least,end = ' : ')
    print("上料开始",end=',')
    print(t0,end=',')
    print(product_num)
    out.append([yingshe[least],t0])



    current_num[least] = product_num
    last_work  = least


    #状态更新
    CNC_status[least] = 4 #等待上下料完成
    CNC_time[least] = 0


    #计算故障
    gz = guzhang()
    if gz == 1:
```

```
            #真正加工后的多久开始故障
            breaktime[least] = random.randint(1,process) + t
            gz_num += 1
            overtime[least] = weixiu() +breaktime[least] + t



#如果完成
for i in range(8):
    if CNC_time[i] >= process and CNC_status[i] == 1:
        CNC_status[i] = 2 #呼叫
        CNC_time[i] = process
# 时间流逝
for i in range(8):
    if CNC_status[i] == 1:
        CNC_time[i] += 1
    if CNC_status[i] == 0 or CNC_status[i] == 2:
        waittime[i] += 1
        move_CNC = 1  #允许移动
    #故障发生处理判别
    if t == breaktime[i] and CNC_status[i] ==1:
        CNC_status[i] = 3
        print(i,end=':')
        print("故障发生",end=',')
        print(t,end=',')
        print(current_num[i])
        #故障的物料、CNC!!、开始时间
        out2[current_num[i]] = [yingshe[i],t]
    if CNC_status[i] == 3 and t>=overtime[i]:
        CNC_status[i]=0
        CNC_time[i] = 0
        print(i,end=':')
        print("故障解除",end=',')
```

```
            print(t)
            out2[current_num[i]].append(t)

    t += 1
for i in range(len(out)):
    #仅输出有下料的
    if len(out[i]) == 3:
        writer.writerow([i+1,out[i][0],out[i][1],out[i][2]])

for i in out2:
    print(i)
    print(out2[i])
    if len(out2[i]) == 3:
        writer2.writerow([i,out2[i][0],out2[i][1],out2[i][2]])
    else:
        writer2.writerow([i,out2[i][0],out2[i][1]])
fn.close()
fn2.close()
```

6. 单工序有故障感知预判策略

```
import random
import csv
import math
import numpy as np
file = 'c2_1.csv'
file2 = 'c2_2.csv'
fn = open(file,"w",newline = '')
fn2 = open(file2,'w',newline = '')
writer2 = csv.writer(fn2)
writer= csv.writer(fn)
out = []
out2 = {}
```

```
#c.index(sorted(c)[2])
#参数
move = [20,33,46]
process = 560 #一道工序
change =[31,28] #上下料时间，偶，奇
clean = 25

product_num = 0

yingshe = {0:2,1:4,2:6,3:8,4:1,5:3,6:5,7:7}

#故障发生函数：
def guzhang():
    temptemp = random.randint(1,100)
    if temptemp == 1:
        return 1
    else:
        return 0
#故障数量
gz_num = 0
def weixiu():
    tempp = random.randint(10*60,20*60)
    return tempp

#计算移动时间：
def movetime(i,j):
    if i == j:
        return 0
    else:
##      print(i)
##      print(j)
        return move[int(math.fabs(i-j)-1)]
```

```
#当前位置 0 1 2 3
loc = 0

#消耗函数
def xiaohao(status,time):
    xiao = [0] * 8
    for i in range(8):
        if status[i] == 0 : # 空
            xiao[i] = movetime(loc,i%4) + change[int(i/4)]
        elif status[i] == 2:
            xiao[i] = movetime(loc,i%4) + change[int(i/4)] +
clean
        elif status[i] == 1:
            xiao[i] = max(movetime(loc,i%4),process - time[i]) +
change[int(i/4)] + clean
        else:
            xiao[i] = 10000
    return xiao
# 得到消耗最小的机器编号
def get_least(xiao):
##    for i in range(len(xiao)):
##        if CNC_status[xiao.index(sorted(xiao)[0])]
    return xiao.index(sorted(xiao)[0])



# 八个机器的状态码，0表示空，1表示正在加工，2表示呼叫抓手,3表示故障，4表示
待上下料
CNC_status = [0]*8
# 八个机器的运行时间
CNC_time = [0] * 8
#八个机器的当前运行的工件编号
current_num = [0]*8
```

```
#总时间
alltime = 8*3600
#故障发生
breaktime = [0] *8
#回复时间
overtime = [0] *8
#等待时间
waittime = [0] *8

RGV_move = 0
RGV_time=0
RGV_work = 0

last_least = -1
last_work = -1
t = 0
while t <= alltime:
##    print("loc:")
##    print(loc)
    #消耗最小的机器编号
    least = get_least(xiaohao(CNC_status,CNC_time))
##    print(least)
##    print(xiaohao(CNC_status,CNC_time))
#    print(loc)
    #如果需要移动,且未在工作和移动！！：
    if  loc != least % 4 and RGV_move == 0 and RGV_work == 0:
        #移动到对应位置
        target = least % 4
        delta_t = movetime(loc,target)
        RGV_time = t+delta_t
        RGV_move = 1
        last_least = least
```

```
    if RGV_move == 1 and t >= RGV_time:
        loc = last_least % 4
        RGV_move = 0  #停止移动

    if RGV_work ==1 and t>=RGV_time:
        RGV_work = 0
        CNC_status[last_work] = 1
        CNC_time[last_work] = 0

#执行更替,一开始或呼叫状态时执行
    if RGV_work == 0 and RGV_move == 0 and loc == least % 4 and
(CNC_status[least] == 2 or CNC_status[least] == 0):
        t0 = t
        #更替时间
        delta_t = change[int(least/4)]
        if CNC_status[least] == 2:
            delta_t += clean
            RGV_work = 1
            RGV_time = t + delta_t
            #完成的工作台
            print(least,end = ' : ')
            print("下料开始",end =',')
            print(t0,end = ',')
            #打印加工后的当前工件编号
            print(current_num[least])
            out[current_num[least]-1].append(t0)

        elif CNC_status[least] == 0:
            RGV_work = 1
            RGV_time = delta_t + t

        #放入新工件
        product_num += 1
```

```
        print(least,end = ' : ')
        print("上料开始",end=',')
        print(t0,end=',')
        print(product_num)
        out.append([yingshe[least],t0])



        current_num[least] = product_num
        last_work  = least


        #状态更新
        CNC_status[least] = 4 #等待上下料完成
        CNC_time[least] = 0


        #计算故障
        gz = guzhang()
        if gz == 1:
            #真正加工后的多久开始故障
            breaktime[least] = random.randint(1,process) + t
            gz_num += 1
            overtime[least] = weixiu() +breaktime[least] + t



    #如果完成
    for i in range(8):
        if CNC_time[i] >= process and CNC_status[i] == 1:
            CNC_status[i] = 2 #呼叫
            CNC_time[i] = process
    # 时间流逝
```

```
    for i in range(8):
        if CNC_status[i] == 1:
            CNC_time[i] += 1
        if CNC_status[i] == 0 or CNC_status[i] == 2:
            waittime[i] += 1
        #故障发生处理判别
        if t == breaktime[i] and CNC_status[i] ==1:
            CNC_status[i] = 3
            print(i,end=':')
            print("故障发生",end=',')
            print(t,end=',')
            print(current_num[i])
            #故障的物料、CNC!!、开始时间
            out2[current_num[i]] = [yingshe[i],t]
        if CNC_status[i] == 3 and t>=overtime[i]:
            CNC_status[i]=0
            CNC_time[i] = 0
            print(i,end=':')
            print("故障解除",end=',')
            print(t)
            out2[current_num[i]].append(t)

    t += 1
for i in range(len(out)):
    #仅输出有下料的
    if len(out[i]) == 3:
        writer.writerow([i+1,out[i][0],out[i][1],out[i][2]])

for i in out2:
    print(i)
    print(out2[i])
    if len(out2[i]) == 3:
        writer2.writerow([i,out2[i][0],out2[i][1],out2[i][2]])
```

```
    else:
        writer2.writerow([i,out2[i][0],out2[i][1]])
fn.close()
fn2.close()
```

7.双工序位置判定+双层感知预判策略

注：由于该代码缩进过多，在论文中阅读效果差，故特附于附件中。

```
bencijisuan = -1
temp = -1
kk= [0] *4
import math
import random
import csv
import numpy as np
#计算移动时间：
def movetime(i,j):
##    if i == j:
##        return 0
##    else:
##        return int(move[int(math.fabs(i-j)-1)])
    return 0



t11 = 400 + 31
t12 = 400 + 28 #第一道工序，下
t21 = 378 + 31
t22 = 378 + 28 #第二道工序，下
move = [20,33,46]

process = [400,378] #工序
change =[31,28] #上下料时间，偶，奇  （上排，下排）
```

```
clean = 25

yingshe = {0:2,1:4,2:6,3:8,4:1,5:3,6:5,7:7}

#故障发生函数：
def guzhang():
##    temptemp = random.randint(1,100)
##    if temptemp == 1:
##        return 1
##    else:
##        return 0
    return 0

def weixiu():
    tempp = random.randint(10*60,20*60)
    return tempp
#计算移动时间：
def movetime(i,j):
    if i == j:
        return 0
    else:
##        print(i)
##        print(j)
        return move[int(math.fabs(i-j)-1)]

#两轮消耗函数
def xiaohao(c1,c2,status,time):
    zuhe = {}
    for i in range(len(c1)):
        for j in range(len(c2)):
            temp = 0
            if status[c1[i]] == 0 or status[c1[i]] == 2:
                temp +=movetime(loc,c1[i]%4) +
```

```
change[int(c1[i]/4)]
            elif status[c1[i]] == 1:
                #一道工序 process
                temp += max(movetime(loc,c1[i]%4),process[0] -
time[c1[i]]) + change[int(c1[i]/4)]
            else:
                temp += 10000

#只有一不为空才考虑后续的消耗：
            if status[c1[i]] != 0:
                # && j对应的消耗
                if status[c2[j]] == 0 :#空
                    temp +=movetime(loc,c2[j]%4) +
change[int(c2[j]/4)]
                elif status[c2[j]] == 2:
                    temp +=movetime(loc,c2[j]%4) +
change[int(c2[j]/4)] + clean
                elif status[c2[j]] == 1:
                    #二道工序 process
                    temp += max(movetime(loc,c2[j]%4),process[1] -
time[c2[j]]) + change[int(c2[j]/4)]
                else:
                    temp += 10000
            zuhe[c1[i],c2[j]] = temp
    return zuhe

def get_least(zidian):
    c = sorted(zidian.items(),key = lambda
item:item[1])[0][0][0]
    d = sorted(zidian.items(),key = lambda
item:item[1])[0][0][1]
    return [c,d]  #第一步机器，第二步机器
```

```
# 第二道故障，第一道重新寻路：
def ercichazhao(loc,c2,status,time):
    xiao = [0] * len(c2)
    for i in range(len(c2)):
        if status[c2[i]] == 0 : # 空
            xiao[i] = movetime(loc,c2[i]%4) +
change[int(c2[i]/4)]
        ####新更新! lastday
        elif status[c2[i]] == 2:
            xiao[i] = movetime(loc,c2[i]%4) +
change[int(c2[i]/4)] + clean
        elif status[c2[i]] == 1:
            #二道工序 process
            xiao[i] = max(movetime(loc,c2[i]%4),process[1] -
time[c2[i]]) + change[int(c2[i]/4)] + clean
        else:
            xiao[i] = 10000

    return c2[xiao.index(sorted(xiao)[0])]
```

```
tup=[t11,t12]
tdown=[t21,t22]
t = 8*3600
for i in range(1,5): #11
    for j in range(1,5): #12
        for m in range(1,5): #21
```

```
            for n in range(1,5): #22
                if (i+m) <= 4 and (j+n) <= 4:

                    k =
min(i*(t/t11)+j*(t/t12),m*(t/t21)+n*(t/t22))
                    if temp == -1:
                        temp = k
                        kk[0] = i
                        kk[1] = j
                        kk[2] = m
                        kk[3] = n

                    elif temp < k:
                        temp = k
                        kk[0] = i
                        kk[1] = j
                        kk[2] = m
                        kk[3] = n
print(kk)

a = [-1] * 8
temp_dis = -1
for a1 in range(2):
    for a2 in range(2):
        for a3 in range(2):
            for a4 in range(2):
                for a5 in range(2):
                    for a6 in range(2):
                        for a7 in range(2):
                            for a8 in range(2):
                                if (a1+a2+a3+a4) ==kk[2] and
(a5+a6+a7+a8)==kk[3]:
                                    bencijisuan = 0
```

```
#当前位置 0 1 2 3
loc = 0


ceshizongshu = 0
#一二道工序机器的编号
CNC1 = []
CNC2 = []

if a1 == 0:
    CNC1.append(0)
else:
    CNC2.append(0)
if a2 == 0:
    CNC1.append(1)
else:
    CNC2.append(1)
if a3 == 0:
    CNC1.append(2)
else:
    CNC2.append(2)

if a4 == 0:
    CNC1.append(3)
else:
    CNC2.append(3)
if a5 == 0:
    CNC1.append(4)
else:
    CNC2.append(4)
if a6 == 0:
    CNC1.append(5)
```

```
            else:
                CNC2.append(5)
            if a7 == 0:
                CNC1.append(6)
            else:
                CNC2.append(6)
            if a8 == 0:
                CNC1.append(7)
            else:
                CNC2.append(7)


            out = []
            out_guzhang = {}
            #故障数量
            gz_num = 0
            # 八个机器的状态码，0表示空，1表示正在加工，2表示呼叫抓手,3表示故障，4表示待上下料
            CNC_status = [0]*8
            # 八个机器的运行时间
            CNC_time = [0] * 8
            #八个机器的当前运行的工件编号
            current_num = [0]*8
            #总时间
            alltime = 8*3600
            #故障发生
            breaktime = [0] *8
            #回复时间
            overtime = [0] *8
            #等待时间
            waittime = [0] *8
            #是否有故障
            breakflag = [0] * 8
```

```
                                gz_time = [0] * 8

                                RGV_move = 0
                                RGV_time=0
                                RGV_work = 0
                                RGV_curnum = -1
                                xialiao = 0
                                #add,未修改
                                last_least = -1
                                last_least2 = -1
                                last_work = -1
                                last_work2 = -1
                                jiuyuan = -1
                                product_num = 0
                                t = 0
                                RGV_status = 0
                                while t <= alltime:
                                    #故障处理
                                    for i in range(8):
                                        if breakflag[i] == 1 and
t >= breaktime[i] and t < overtime[i] and CNC_status[i] == 1:
#加工状态故障
                                            CNC_status[i] = 3  #故
障状态
                                            print("故障",end= ' :')
                                            print(i,end=',')

print(current_num[i],end=',')
                                            print(t)

out_guzhang[current_num[i]] = [current_num[i],yingshe[i],t]
                                            if last_least == i:
```

```
                                        RGV_status = 0

                                    elif breakflag[i] == 1 and
t >= overtime[i] and CNC_status[i] == 3:
                                        CNC_status[i] = 0 #修复
后，工件作废！！
                                        CNC_time[i] = 0
                                        breakflag[i] = 0
                                        print("故障修复
",end=':')

print(current_num[i],end=',')
                                        print(t)

out_guzhang[current_num[i]].append(t)



                                if RGV_status == 0 or
RGV_status == 4:
                                    #消耗最小的机器编号
                                    path =
get_least(xiaohao(CNC1,CNC2,CNC_status,CNC_time))
                                    least1 = path[0]
                                    least2 = path[1]   #第二道的
下家
                                    least = least1
                                    jiuyuan = least%4
                                    RGV_status = 1 #去执行1道工
序
                                ##
print(xiaohao(CNC1,CNC2,CNC_status,CNC_time))
                                #第一部分执行完毕
```

```
                                elif RGV_status == 2:
                                    #二次重新查找
                                    least =
ercichazhao(jiuyuan,CNC2,CNC_status,CNC_time)
                                    RGV_status += 1


                                #    print(loc)

                                    #如果需要移动,且未在工作和移
动！！：
                                    if  loc != least % 4 and
RGV_move == 0 and RGV_work == 0:
                                        #移动到对应位置
                                        target = least % 4
                                        delta_t =
movetime(loc,target)
                                        RGV_time = t+delta_t
                                        RGV_move = 1
                                        last_least = least
                                        last_least2 = least2
                                    if RGV_move == 1 and t >=
RGV_time:
                                        loc = last_least % 4
                                        RGV_move = 0  #停止移动

                                    #上下料结束
                                    if RGV_work ==1 and
t>=RGV_time:
                                        RGV_work = 0
                                        CNC_status[last_work] = 1
                                        CNC_time[last_work] = 0
                                        RGV_status += 1
```

```
                                    if xialiao == 0:
                                        RGV_status = 0



                                #执行更替,一开始或呼叫状态时执
行,RGV_status = 1/3
                                if (RGV_status ==1 or
RGV_status == 3) and RGV_work == 0 and RGV_move == 0 and loc
== least % 4 and (CNC_status[least] == 2 or CNC_status[least]
== 0):
                                    t0 = t
                                    #更替时间
                                    delta_t =
change[int(least/4)]

############################################################
###
                                    if RGV_status == 1:

                                        if CNC_status[least]
== 2 :
                                            #完成的工作台
                                ##              print(least,end =
' : ')
                                ##              print("下料开始
",end =',')
                                ##              print(t0,end =
',')
                                ##              #打印加工后的当前工
```

```
件编号
                                ##
print(current_num[least])

                                        #是否拿到半成品
                                        xialiao = 1
                                        RGV_curnum =
current_num[least]


out[current_num[least]-1].append(t)

                                    #未拿到半成品
                                    else:
                                        xialiao = 0

                                #放入新工件
                                    product_num += 1
                                    ceshizongshu += 1
                            ##          print(least,end =
' : ')
                            ##          print("上料开始
",end=',')
                            ##          print(t0,end=',')
                            ##          print(product_num)

                                #第一道工序的故障初始化
                                    breakflag[least] =
guzhang()
                                    if breakflag[least] ==
1:
                                        breaktime[least] =
t + random.randint(1,process[0]) + change[0]
```

```
                                                overtime[least] =
breaktime[least] + weixiu()



                                                current_num[least] =
product_num


out.append([product_num,yingshe[least],t])  #真正的cnc
                                                #状态更新
                                                CNC_status[least] = 4
#等待上下料完成
                                                CNC_time[least] = 0
                                                RGV_work = 1
##
                                                RGV_time = delta_t + t
##

############################################################

                                            elif RGV_status == 3:

                                                #假设只有第二道结束后要清洗
                                                if CNC_status[least]
== 2:  #有第一道的成品也不一定有第二次下料，单独判断!

                                                    delta_t += clean
                                                    RGV_work = 1
                                                    RGV_time = t +
delta_t
                                                    #完成的工作台
                                        ##            print(least,end =
```

```
' : ')
                                ##              print("第二次下料开
始",end =',')
                                ##              print(t0,end =
',')
                                ##              #打印加工后的当前工
件编号
                                ##
print(current_num[least])


out[current_num[least]-1].append(t)

                                            if xialiao == 1: #有第
一道的成品
                                                xialiao = 0

                                                current_num[least]
= RGV_curnum


out[current_num[least]-1].append(yingshe[least])  #真正的cnc

out[current_num[least]-1].append(t)


                                ##              print(least,end =
' : ')
                                ##              print("第二次上料开
始",end=',')
                                ##              print(t0,end=',')
                                ##
print(current_num[least])
```

```
                                                    #状态更新
                                                    CNC_status[least] =
4 #等待上下料完成
                                                    CNC_time[least] = 0
                                                    RGV_work = 1
                                                    RGV_time = t +
delta_t

                                                    #第二道工序的故障初始
化,前提：有第一道的半成品
                                                    breakflag[least] =
guzhang()
                                                    if breakflag[least]
== 1:
                                                        breaktime[least]
= t + random.randint(1,process[1]) + change[1]
                                                        overtime[least]
=  breaktime[least] + weixiu()


                                                #重新找第一道工序
                                                else:
                                                    RGV_status = 0
                                            #current_num[least] =
product_num

##############################################################
################

                                            last_work  = least
```

```
                                    #如果完成
                                    for i in range(8):
                                        if i in CNC1:
                                            if CNC_time[i] >=
process[0] and CNC_status[i] == 1:
                                                CNC_status[i] = 2 #
呼叫
                                                CNC_time[i] =
process[0]
                                        #二道工序
                                        if i in CNC2:
                                            if CNC_time[i] >=
process[1] and CNC_status[i] == 1:
                                                CNC_status[i] = 2 #
呼叫
                                                CNC_time[i] =
process[1]

                                    # 时间流逝
                                    for i in range(8):
                                        if CNC_status[i] == 1:
                                            CNC_time[i] += 1
                                        if CNC_status[i] == 0 or
CNC_status[i] == 2:
                                            waittime[i] += 1
                                        if CNC_status[i] == 3:
                                            gz_time[i] += 1
```

```
    t += 1


for i in range(len(out)):
    if len(out[i]) == 7:
        bencijisuan += 1


if temp_dis == -1:
    temp_dis = bencijisuan
    a[0]=a1
    a[1]=a2
    a[2]=a3
    a[3]=a4
    a[4]=a5
    a[5]=a6
    a[6]=a7
    a[7]=a8
if temp_dis < bencijisuan:
    temp_dis = bencijisuan
    a[0]=a1
    a[1]=a2
    a[2]=a3
    a[3]=a4
    a[4]=a5
    a[5]=a6
    a[6]=a7
    a[7]=a8
print(bencijisuan)
t = [a1,a2,a3,a4,a5,a6,a7,a8]
print(t)
```

```
for i in range(len(a)):
    print(i+1,end = ':')
    if a[i]:
        print("二道工序2")
    else:
        print("一道工序1")
```

8. 双工序无障碍双层感知预判策略

```
import random
import csv
import math
import numpy as np
file = 'c1_1.csv'
fn = open(file,"w",newline = '')
writer= csv.writer(fn)
out = []
#参数
move = [20,33,46]
process = [400,378] #工序
change =[31,28] #上下料时间，偶，奇  （上排，下排）
clean = 25

yingshe = {0:2,1:4,2:6,3:8,4:1,5:3,6:5,7:7}

###故障发生函数：
##def guzhang():
##    temptemp = random.randint(1,100)
##    if temptemp == 1:
##        return 1
##    else:
##        return 0
#故障数量
```

```
gz_num = 0
def weixiu():
    tempp = random.randint(10*60,20*60)
    return tempp
#计算移动时间：
def movetime(i,j):
    if i == j:
        return 0
    else:
##        print(i)
##        print(j)
        return move[int(math.fabs(i-j)-1)]

#两轮消耗函数
def xiaohao(c1,c2,status,time):
    zuhe = {}
    for i in range(len(c1)):
        for j in range(len(c2)):
            temp = 0
            if status[c1[i]] == 0 or status[c1[i]] == 2:
                temp +=movetime(loc,c1[i]%4) +
change[int(c1[i]/4)]
            elif status[c1[i]] == 1:
                #一道工序 process
                temp += max(movetime(loc,c1[i]%4),process[0] -
time[c1[i]]) + change[int(c1[i]/4)]
            else:
                temp += 10000

#只有一不为空才考虑后续的消耗：
            if status[c1[i]] != 0:
                # && j对应的消耗
                if status[c2[j]] == 0 :#空
```

```
                temp +=movetime(loc,c2[j]%4) +
change[int(c2[j]/4)]
            elif status[c2[j]] == 2:
                temp +=movetime(loc,c2[j]%4) +
change[int(c2[j]/4)] + clean
            elif status[c2[j]] == 1:
                #二道工序 process
                temp += max(movetime(loc,c2[j]%4),process[1] -
time[c2[j]]) + change[int(c2[j]/4)]
            else:
                temp += 10000
        zuhe[c1[i],c2[j]] = temp
    return zuhe
def get_least(zidian):
    c = sorted(zidian.items(),key = lambda
item:item[1])[0][0][0]
    d = sorted(zidian.items(),key = lambda
item:item[1])[0][0][1]
    return [c,d]  #第一步机器，第二步机器

# 第二道故障，第一道重新寻路：
def ercichazhao(loc,c2,status,time):
    xiao = [0] * len(c2)
    for i in range(len(c2)):
        if status[c2[i]] == 0 : # 空
            xiao[i] = movetime(loc,c2[i]%4) +
change[int(c2[i]/4)]
        ####新更新！lastday
        elif status[c2[i]] == 2:
            xiao[i] = movetime(loc,c2[i]%4) +
change[int(c2[i]/4)] + clean
        elif status[c2[i]] == 1:
            #二道工序 process
```

```
            xiao[i] = max(movetime(loc,c2[i]%4),process[1] -
time[c2[i]]) + change[int(c2[i]/4)] + clean
        else:
            xiao[i] = 10000

    return c2[xiao.index(sorted(xiao)[0])]


#当前位置 0 1 2 3
loc = 0

#一二道工序机器的编号
CNC1 = [1,4,6,7]
CNC2 = [0,2,3,5]

# 八个机器的状态码，0表示空，1表示正在加工，2表示呼叫抓手,3表示故障，4表示
待上下料
CNC_status = [0]*8
# 八个机器的运行时间
CNC_time = [0] * 8
#八个机器的当前运行的工件编号
current_num = [0]*8
#总时间
alltime = 8*3600
#故障发生
breaktime = [0] *8
#回复时间
overtime = [0] *8
#等待时间
waittime = [0] *8

kong1 = 0
kong2 = 0
```

```
tiao1 = 0
tiao2 = 0
RGV_move = 0
RGV_time=0
RGV_work = 0
RGV_curnum = -1
xialiao = 0
#add,未修改
last_least = -1
last_least2 = -1
last_work = -1
last_work2 = -1
jiuyuan = -1
product_num = 0
t = 0
RGV_status = 0
while t <= alltime:
    if RGV_status == 0:
        #消耗最小的机器编号
        path = get_least(xiaohao(CNC1,CNC2,CNC_status,CNC_time))
        least1 = path[0]
        least2 = path[1]   #第二道的下家
        least = least1
        jiuyuan = least%4
        RGV_status = 1 #去执行1道工序

    #第一部分执行完毕
    elif RGV_status == 2:
        #二次重新查找
        least = ercichazhao(jiuyuan,CNC2,CNC_status,CNC_time)
        RGV_status += 1
    elif RGV_status ==  4:
        RGV_status = 0
```

```
##    print(least)
##    print(xiaohao(CNC_status,CNC_time))
#    print(loc)

    #如果需要移动,且未在工作和移动！！：
    if  loc != least % 4 and RGV_move == 0 and RGV_work == 0:
        #移动到对应位置
        target = least % 4
        delta_t = movetime(loc,target)
        RGV_time = t+delta_t
        RGV_move = 1
        last_least = least
        last_least2 = least2
    if RGV_move == 1 and t >= RGV_time:
        loc = last_least % 4
        RGV_move = 0  #停止移动

    #上下料结束
    if RGV_work ==1 and t>=RGV_time:
        RGV_work = 0
        CNC_status[last_work] = 1
        CNC_time[last_work] = 0
        RGV_status += 1
##    if tiao1 == 1:
##        tiao1 = 0
##        RGV_status = 0 #重新寻找一道
##    if tiao2 == 1:
##        tiao2 = 0
##        RGV_status = 0 #不清洗，重新找一道
    #执行更替,一开始或呼叫状态时执行,RGV_status = 1/3
    if (RGV_status ==1 or RGV_status == 3) and RGV_work == 0
and RGV_move == 0 and loc == least % 4 and (CNC_status[least]
== 2 or CNC_status[least] == 0):
```

```
        t0 = t
        #更替时间
        delta_t = change[int(least/4)]
#################################################################
###
        if RGV_status == 1:

            if CNC_status[least] == 2 :
                #完成的工作台
                print(least,end = ' : ')
                print("下料开始",end =',')
                print(t0,end = ',')
                #打印加工后的当前工件编号
                print(current_num[least])
                xialiao = 1
                RGV_curnum = current_num[least]

                out[current_num[least]-1].append(t)

            else:
                xiaoliao = 0

            RGV_work = 1                    ##
            RGV_time = delta_t + t            ##
        #放入新工件
            product_num += 1
            print(least,end = ' : ')
            print("上料开始",end=',')
            print(t0,end=',')
            print(product_num)
            current_num[least] = product_num
            out.append([product_num,yingshe[least],t])
            #状态更新
```

```
            CNC_status[least] = 4 #等待上下料完成
            CNC_time[least] = 0
############################################################

        elif RGV_status == 3:

            #假设只有第二道结束后要清洗
            if CNC_status[least] == 2:  #有第一道的成品也不一定有第二次下料，最晚！！

                delta_t += clean
                RGV_work = 1
                RGV_time = t + delta_t
                #完成的工作台
                print(least,end = ' : ')
                print("第二次下料开始",end =',')
                print(t0,end = ',')
                #打印加工后的当前工件编号
                print(current_num[least])

                out[current_num[least]-1].append(t)

            if xialiao == 1: #有第一道的成品
                xialiao = 0
                current_num[least] = RGV_curnum
                print(least,end = ' : ')
                print("第二次上料开始",end=',')
                print(t0,end=',')
                print(current_num[least])

                out[current_num[least]-1].append(yingshe[least])
                out[current_num[least]-1].append(t)
```

```
                #状态更新
                CNC_status[least] = 4 #等待上下料完成
                CNC_time[least] = 0
                RGV_work = 1
                RGV_time = t + delta_t
            else:
                RGV_status = 0
        #current_num[least] = product_num
##################################################################
#################

        last_work  = least




    #如果完成
    for i in range(8):
        if i in CNC1:
            if CNC_time[i] >= process[0] and CNC_status[i] == 1:
                CNC_status[i] = 2 #呼叫
                CNC_time[i] = process[0]
        #二道工序
        if i in CNC2:
            if CNC_time[i] >= process[1] and CNC_status[i] == 1:
                CNC_status[i] = 2 #呼叫
                CNC_time[i] = process[1]

    # 时间流逝
    for i in range(8):
        if CNC_status[i] == 1:
            CNC_time[i] += 1
```

```
        if CNC_status[i] == 0 or CNC_status[i] == 2:
            waittime[i] += 1

    t += 1

for i in range(len(out)):
    if len(out[i]) == 7:

writer.writerow([out[i][0],out[i][1],out[i][2],out[i][3],out[i
][4],out[i][5],out[i][6]])
fn.close()
```

9. 双工序有障碍感知预判策略

```
import random
import csv
import math
import numpy as np
file = 'c1_1.csv'
file2 = 'guzhang.csv'
fn = open(file,"w",newline = '')
fn2 = open(file2,'w',newline = '')
writer= csv.writer(fn)
writer2 = csv.writer(fn2)
out = []
out_guzhang = {}
#参数
move = [20,33,46]
process = [400,378] #工序
change =[31,28] #上下料时间，偶，奇  （上排，下排）
clean = 25

yingshe = {0:2,1:4,2:6,3:8,4:1,5:3,6:5,7:7}
```

```
#故障发生函数：
def guzhang():
    temptemp = random.randint(1,100)
    if temptemp == 1:
        return 1
    else:
        return 0
##    return 0

#故障数量
gz_num = 0
def weixiu():
    tempp = random.randint(10*60,20*60)
    return tempp
#计算移动时间：
def movetime(i,j):
    if i == j:
        return 0
    else:
##        print(i)
##        print(j)
        return move[int(math.fabs(i-j)-1)]

#两轮消耗函数
def xiaohao(c1,c2,status,time):
    zuhe = {}
    for i in range(len(c1)):
        for j in range(len(c2)):
            temp = 0
            if status[c1[i]] == 0 or status[c1[i]] == 2:
                temp +=movetime(loc,c1[i]%4) +
change[int(c1[i]/4)]
```

```
            elif status[c1[i]] == 1:
                #一道工序 process
                temp += max(movetime(loc,c1[i]%4),process[0] -
time[c1[i]]) + change[int(c1[i]/4)]
            else:
                temp += 10000

#只有一不为空才考虑后续的消耗：
            if status[c1[i]] != 0:
                # && j对应的消耗
                if status[c2[j]] == 0 :#空
                    temp +=movetime(loc,c2[j]%4) +
change[int(c2[j]/4)]
                elif status[c2[j]] == 2:
                    temp +=movetime(loc,c2[j]%4) +
change[int(c2[j]/4)] + clean
                elif status[c2[j]] == 1:
                    #二道工序 process
                    temp += max(movetime(loc,c2[j]%4),process[1] -
time[c2[j]]) + change[int(c2[j]/4)]
                else:
                    temp += 10000
            zuhe[c1[i],c2[j]] = temp
    return zuhe
def get_least(zidian):
    c = sorted(zidian.items(),key = lambda
item:item[1])[0][0][0]
    d = sorted(zidian.items(),key = lambda
item:item[1])[0][0][1]
    return [c,d]  #第一步机器，第二步机器

# 第二道故障，第一道重新寻路：
def ercichazhao(loc,c2,status,time):
```

```
    xiao = [0] * len(c2)
    for i in range(len(c2)):
        if status[c2[i]] == 0 : # 空
            xiao[i] = movetime(loc,c2[i]%4) +
change[int(c2[i]/4)]
        ####新更新! lastday
        elif status[c2[i]] == 2:
            xiao[i] = movetime(loc,c2[i]%4) +
change[int(c2[i]/4)] + clean
        elif status[c2[i]] == 1:
            #二道工序 process
            xiao[i] = max(movetime(loc,c2[i]%4),process[1] -
time[c2[i]]) + change[int(c2[i]/4)] + clean
        else:
            xiao[i] = 10000

    return c2[xiao.index(sorted(xiao)[0])]


#当前位置 0 1 2 3
loc = 0


ceshizongshu = 0
#一二道工序机器的编号
CNC1 = [1,4,6,7]
CNC2 = [0,2,3,5]

# 八个机器的状态码，0表示空，1表示正在加工，2表示呼叫抓手,3表示故障，4表示
待上下料
CNC_status = [0]*8
# 八个机器的运行时间
CNC_time = [0] * 8
```

```
#八个机器的当前运行的工件编号
current_num = [0]*8
#总时间
alltime = 8*3600
#故障发生
breaktime = [0] *8
#回复时间
overtime = [0] *8
#等待时间
waittime = [0] *8
#是否有故障
breakflag = [0] * 8

gz_time = [0] * 8

RGV_move = 0
RGV_time=0
RGV_work = 0
RGV_curnum = -1
xialiao = 0
#add,未修改
last_least = -1
last_least2 = -1
last_work = -1
last_work2 = -1
jiuyuan = -1
product_num = 0
t = 0
RGV_status = 0
while t <= alltime:
    #故障处理
    for i in range(8):
        if breakflag[i] == 1 and t >= breaktime[i] and t <
```

```
overtime[i] and CNC_status[i] == 1:    #加工状态故障
            CNC_status[i] = 3  #故障状态
            print("故障",end= ' :')
            print(i,end=',')
            print(current_num[i],end=',')
            print(t)
            out_guzhang[current_num[i]] =
[current_num[i],yingshe[i],t]
            if last_least == i:
                RGV_status = 0

        elif breakflag[i] == 1 and t >= overtime[i] and
CNC_status[i] == 3:
            CNC_status[i] = 0 #修复后，工件作废！！
            CNC_time[i] = 0
            breakflag[i] = 0
            print("故障修复",end=':')
            print(current_num[i],end=',')
            print(t)
            out_guzhang[current_num[i]].append(t)



    if RGV_status == 0 or RGV_status == 4:
        #消耗最小的机器编号
        path = get_least(xiaohao(CNC1,CNC2,CNC_status,CNC_time))
        least1 = path[0]
        least2 = path[1]   #第二道的下家
        least = least1
        jiuyuan = least%4
        RGV_status = 1 #去执行1道工序
##        print(xiaohao(CNC1,CNC2,CNC_status,CNC_time))
    #第一部分执行完毕
```

```
    elif RGV_status == 2:
        #二次重新查找
        least = ercichazhao(jiuyuan,CNC2,CNC_status,CNC_time)
        RGV_status += 1


#    print(loc)

    #如果需要移动,且未在工作和移动！！：
    if  loc != least % 4 and RGV_move == 0 and RGV_work == 0:
        #移动到对应位置
        target = least % 4
        delta_t = movetime(loc,target)
        RGV_time = t+delta_t
        RGV_move = 1
        last_least = least
        last_least2 = least2
    if RGV_move == 1 and t >= RGV_time:
        loc = last_least % 4
        RGV_move = 0  #停止移动

    #上下料结束
    if RGV_work ==1 and t>=RGV_time:
        RGV_work = 0
        CNC_status[last_work] = 1
        CNC_time[last_work] = 0
        RGV_status += 1
        if xialiao == 0:
            RGV_status = 0
```

```
    #执行更替,一开始或呼叫状态时执行,RGV_status = 1/3
    if (RGV_status ==1 or RGV_status == 3) and RGV_work == 0
and RGV_move == 0 and loc == least % 4 and (CNC_status[least]
== 2 or CNC_status[least] == 0):
        t0 = t
        #更替时间
        delta_t = change[int(least/4)]
#############################################################
###
        if RGV_status == 1:

            if CNC_status[least] == 2 :
                #完成的工作台
                print(least,end = ' : ')
                print("下料开始",end =',')
                print(t0,end = ',')
                #打印加工后的当前工件编号
                print(current_num[least])

                #是否拿到半成品
                xialiao = 1
                RGV_curnum = current_num[least]

                out[current_num[least]-1].append(t)

            #未拿到半成品
            else:
                xialiao = 0

        #放入新工件
            product_num += 1
```

```
            ceshizongshu += 1
            print(least,end = ' : ')
            print("上料开始",end=',')
            print(t0,end=',')
            print(product_num)

        #第一道工序的故障初始化
            breakflag[least] = guzhang()
            if breakflag[least] == 1:
                breaktime[least] = t +
random.randint(1,process[0]) + change[0]
                overtime[least] =  breaktime[least] + weixiu()



            current_num[least] = product_num

            out.append([product_num,yingshe[least],t])  #真正的cnc
            #状态更新
            CNC_status[least] = 4 #等待上下料完成
            CNC_time[least] = 0
            RGV_work = 1                    ##
            RGV_time = delta_t + t           ##
##############################################################

        elif RGV_status == 3:

            #假设只有第二道结束后要清洗
            if CNC_status[least] == 2:  #有第一道的成品也不一定有第二
次下料，单独判断!

                delta_t += clean
                RGV_work = 1
```

```
                RGV_time = t + delta_t
                #完成的工作台
                print(least,end = ' : ')
                print("第二次下料开始",end =',')
                print(t0,end = ',')
                #打印加工后的当前工件编号
                print(current_num[least])

                out[current_num[least]-1].append(t)

            if xialiao == 1: #有第一道的成品
                xialiao = 0

                current_num[least] = RGV_curnum

                out[current_num[least]-1].append(yingshe[least])
#真正的cnc
                out[current_num[least]-1].append(t)


                print(least,end = ' : ')
                print("第二次上料开始",end=',')
                print(t0,end=',')
                print(current_num[least])



                #状态更新
                CNC_status[least] = 4 #等待上下料完成
                CNC_time[least] = 0
                RGV_work = 1
                RGV_time = t + delta_t
```

```
                #第二道工序的故障初始化,前提：有第一道的半成品
                breakflag[least] = guzhang()
                if breakflag[least] == 1:
                    breaktime[least] = t +
random.randint(1,process[1]) + change[1]
                    overtime[least] =  breaktime[least] + weixiu()


            #重新找第一道工序
            else:
                RGV_status = 0
        #current_num[least] = product_num
##################################################################
#################

        last_work  = least




    #如果完成
    for i in range(8):
        if i in CNC1:
            if CNC_time[i] >= process[0] and CNC_status[i] == 1:
                CNC_status[i] = 2 #呼叫
                CNC_time[i] = process[0]
        #二道工序
        if i in CNC2:
            if CNC_time[i] >= process[1] and CNC_status[i] == 1:
                CNC_status[i] = 2 #呼叫
                CNC_time[i] = process[1]
```

```
    # 时间流逝
    for i in range(8):
        if CNC_status[i] == 1:
            CNC_time[i] += 1
        if CNC_status[i] == 0 or CNC_status[i] == 2:
            waittime[i] += 1
        if CNC_status[i] == 3:
            gz_time[i] += 1

    t += 1

for i in range(len(out)):
    if len(out[i]) == 7:

writer.writerow([out[i][0],out[i][1],out[i][2],out[i][3],out[i
][4],out[i][5],out[i][6]])
for i in range(len(out_guzhang.items())):
    writer2.writerow(list(out_guzhang.items())[i][1])
fn.close()
fn2.close()
```