# GA for JSP

## 一 、GA步骤

### 1.GA

Step1 初始化，设定种群规模、迭代次数等参数，并初始化种群

Step2 进行操作

Step2.1 复制

Step2.2 交叉

Step2.3 变异

Step 3 对种群进行评价，若得到满意解或达到了迭代次数，就输出，否则，继续进行步骤2

## 二、MATLAB实现遗传算法的JSP求解

### 1 引入FT10算例

0 29 1 78 2 9 3 36 4 49 5 11 6 62 7 56 8 44 9 21

0 43 2 90 4 75 9 11 3 69 1 28 6 46 5 46 7 72 8 30

1 91 0 85 3 39 2 74 8 90 5 10 7 12 6 89 9 45 4 33

1 81 2 95 0 71 4 99 6 9 8 52 7 85 3 98 9 22 5 43

2 14 0 6 1 22 5 61 3 26 4 69 8 21 7 49 9 72 6 53

2 84 1 2 5 52 3 95 8 48 9 72 0 47 6 65 4 6 7 25

1 46 0 37 3 61 2 13 6 32 5 21 9 32 8 89 7 30 4 55

2 31 0 86 1 46 5 74 4 32 6 88 8 19 9 48 7 36 3 79

0 76 1 69 3 76 5 51 2 85 9 11 6 40 7 89 4 26 8 74

1 85 0 13 2 61 6 7 8 64 9 76 5 47 3 52 4 90 7 45

奇数列机器号加1，data（：，odd）+=1；

### 2 随机生成初试种群

*~~初始种群的规模是initalPopu（工件数\*机器数） 这里工序数=机器数~~*

*~~这里每10个数是一个块，表示一个工件的安排，这10个数应在1-10之间，且不能重复，~~*

*~~1用temp=randperm（10）实现10个不重复的1-10数字~~*

*~~2 重复循环temp=（temp，ranperm（10）） 9次，即可实现~~*

*~~然后用initalPopus（popu，100）表示整个种群~~*

*~~调用上一个函数100次行扩展即可~~*

呃呃 解码快写完了又发现这里错了。。。

这里要的一个染色体上面的元素是工件号 这个工件出现几次，就是这个工件第几个工序

2 1 1 4 10 1.。。。这样的

每十个元素里面是可以有重复的 在每行里面 1-10各随机出现10次就行

那就这样

还是先生成a=1-10 1- 10 1-10.。。。这个序列

然后随机一个b=randperm（100）

再用这个随机数组做种子 成功随机数组

c=a(b) 先试试这样能行吗

好好好 可以

### 3 解码，插入工序安排 生成安排表

好，现在已经有了一个安排序列了 initialPopu了

规划的时间表是这样子的

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| schedule | | | | |
| 工件ID | 工件工序ID | 机器ID | 开始时间 | 结束时间 |
| workpieceId | W.K.processId | machId | startTime | endTime |

Schedule的行数是总工序数 100

每行表示一个工序的安排

这里还是像开学学的FJSP那样，用插入的方法

对一个initialPopu行，是一个染色体，考虑把这个函数分割为对一个染色体的解码操作，而不是对整个种群的解码操作

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *~~工序号~~* | *~~1~~* | *~~2~~* | *~~3~~* | *~~4~~* | *~~5~~* | *~~6~~* | *~~7~~* | *~~8~~* | *~~9~~* | *~~10~~* |
| *~~机器号~~* | *~~9~~* | *~~5~~* | *~~2~~* | *~~3~~* | *~~4~~* | *~~1~~* | *~~6~~* | *~~10~~* | *~~8~~* | *~~7~~* |

*~~取了一个染色体的前10列，观察一下~~*

*~~可以这样想：顺序的遍历这个一维数组，对每一个元素~~*

*~~记录要花费的时间speedTime，用的的机器号machId和这个工序用到的这个机器总共被用了多少次machIdUseNum，该工序的最早可开工时间woekpieceCanStartTime，约定开始时间workpieceStartTime和结束时间workpieceEndTime~~*

*~~记录过程：遍历到第i个工序的时候，通过读data数组，第~~*

*~~记录完之后，~~*

这里改完了上面生成初试种群的错误之后，就这样想

我要遍历每一个元素 也就是每一个工序，我要把它插入进去，我就得知道

1这个工序对应的这个工件是第几道工序work piece ProcesId（1，10），每遇到这个工件之后他都得自加，

2呢是是这个工序的可开工时间，或者说是这个工件的上道工序结束时间work pieceCanStartTime（1，10）

然后进入循环

For i=1：100

首先要知道是那个工件WorkpieceId，知道是工件的第几道工序work piece ProcesId（i），知道是要用哪个机器machId，知道花费的时间

然后开始安排：

找到对应的机器的已经安排的任务，单独拉出来newSchedule，

如果这个数组是空的，那就好办，第一个插入进去，然后 break；

如果不是空的，那就是有安排了，就找所有的空闲时间， 先看安排表的第一行，如果能插进去，就插，插不进去的 话，开始遍历每个间隔行，能插，就插进去然后设置bool 表示已经插了，然后break这个循环。循环结束之后，只 剩下了最后一行和空白的间隔了，这时候，检查bool类型 看插没插，插了的话就break，没插就继续直接插入最后一 行。

这里的插入不是真的插，因为是单拉出来的只包含这个机器的 时间表，循环结束就free了，所以设置个变量： workpieceStartTime，在这个循环里面，更新Schedule，结束

End

### 4 生成甘特图

这里函数的输入是一个序列的schedule

输出是一张图

加图例这一步不大会，那个legend怎么试都不行，最后GPT了，他写的我也每看懂是个什么意思

### 5 适应度评价

评价一个染色体的适应度值

* 适应度函数为最大完工时间，schedule中第5列的最大值
* 目标是追求其最小

输入一个种群，对每个染色体

生成schedule，然后找到schedule第五列的最大值，这个就是适应度值。

### 6 轮盘赌选择法对种群进行染色体的选择

好好好，终于搞完了，现在进行选择操作

selectChromo()

要使用轮盘赌选择法，需要评价种群中各个染色体的适应度fitness，这是第一列，评价完之后，需要知道fitness哪一个对应哪一个，这是第二列，然后依适应度值升序排列，乘一个系数1.2吧就，得到第三列变适应度，第四列进行归一化处理，第五列是累加列。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| roulette | | | | |
| 染色体编号 | 适应度 | 新的适应度 | 归一之后的新的适应度 | 轮盘区域 |

然后，对一个种群，行遍历：

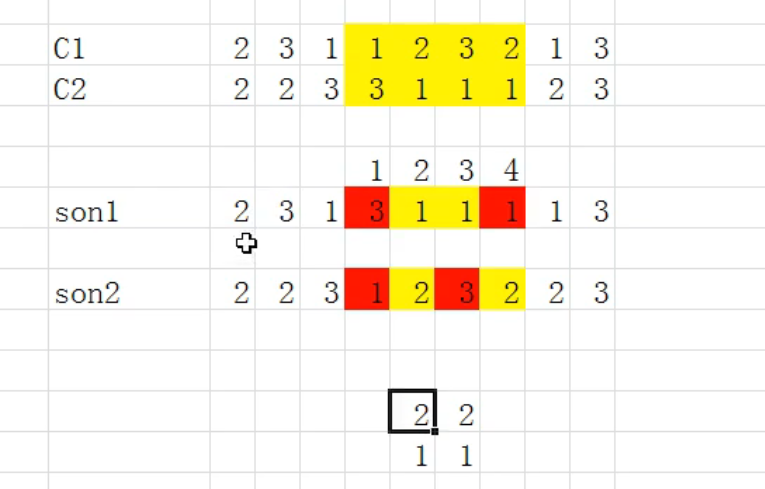
生成一个随机数0-1，命中哪个累加列，就选中对应的染色体遗产下去。

然后这样生成一个新的种群。

### 7 对种群染色体交叉

首先，设置交叉概率

对每两个染色体，生成随机数，如果小于随机数，就交叉



交叉用两点交叉；



1. 随机生成两个整数，按小大排列好，两父代这之间的完全互换，生成两个子代（pChromo1/2、cChromo1/2、subsegment1/2）
2. 对子代进行检查：

2.1对截取的subsegment进行对比，去掉相同的部分newSubsegment1/2=createNewSubsegment（）

2.2剩下的新字串，如果size大于0，就是有不匹配的地方，就要进行改进，这时候要检查，交换的位置，有三种情况1.交换的从前边开始2.交换的在最后3.交换的在中间。

片段newSubsegment1/2分别从subsegment1/2得到，subsegment1/2分别从pChromo1/2得到。所以，newSubsegment2里面的元素，代表了cChromo1里面多的步骤，要遍历cChromo1，对找到的NS2里面的元素用对应的NS1表示。

cC1用NS2查，用NS1填充

cC2用NS1查，用NS2填充

1. 依次遍历，得到交叉之后的种群。

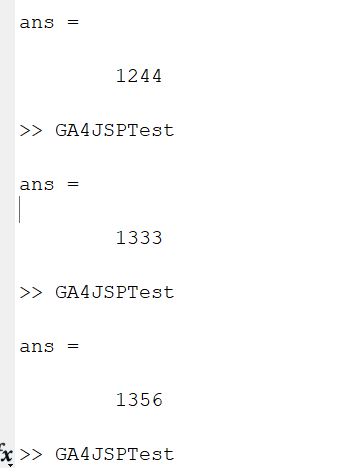
难受，改了两天bug了，还是不知道哪错了。

我超，乱改改莫名其妙又能运行了。难绷

### 8 染色体变异

简简单单逆序一下

### 9 设置参数并运行

 popu=200;%得是偶数，要不然交叉就错了

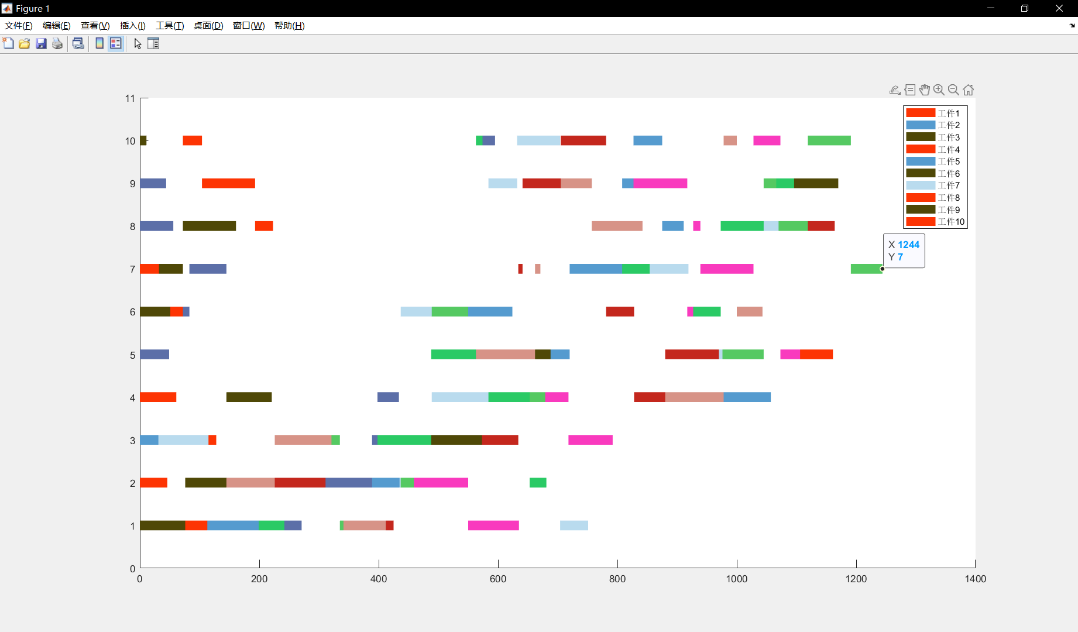
maxIterate=300;

nowIterate=0;

Pcross=0.6;

Pmuta=0.15;

在这个情况下，运行几次的结果在（1200，1400）区间内。



# 算法优化

## 混合遗传算法

### 1 总体思想

上面的运行结果再最优的情况下依然是1200+，现在考虑优化算法。

加入禁忌搜索。

禁忌搜索包括 1设置相关系数 2对每一个染色体进行适应度评价 3找到关键路径，找到染色体的关键块，4对关键块进行N5邻域解构操作，得到一批邻域解5对邻域解进行适应度评价，得到最佳解，5查找禁忌表，观察是否在禁忌表内及是否特赦该邻域解，7最终更新染色体和紧急表

依据上面的理论，再遗传算法的变异操作之后，设置以下操作：

%开始进行禁忌搜索

%对种群的每个染色体搜索进行禁忌搜索

for i=1:popu

%解码，得到schedule1，并进行适应度评价，找到该染色体的最大完工时间

schedule1 = createSchedule(changeData,chromos(i),workpieceNum,machNum);

Cmax=max(schedule1(:,5));

%对染色体进行逆解码，得到schedule1

schedule2=createFlipSchedule(changeData,chromos(i),workpieceNum,machNum,Cmax);

%找到关键路径，找到染色体关键块

keyPath = searchKeyPath();

keyBlock = searchKeyBlock();

%对关键块进行N5邻域结构操作，得到邻域解

neighborhoodChromos=creatneighborhoodChromos();

%对所有的邻域解进行评价，得到最佳解,查找禁忌表

[fitness,chromos] = calcFitness(chromos,popu,changeData);

%更新染色体和禁忌表

tabooTable=changeTabooTable();

end

### 2 顺解码

可以直接用原来的createSchedule() 函数。

得到schedule 和 Cmax 就是最大完工时间

### 3 逆解码

这里对比 createSchedule() 函数 构建createFlipSchedule 函数

倒着排，只是工序的加工时间的漂移，工件的工序还是和顺解码一样保持不变。

规划的时间表依然是这样子的

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| schedule | | | | |
| 工件ID | 工件工序ID | 机器ID | 开始时间 | 结束时间 |
| workpieceId | W.K.processId | machId | startTime | endTime |

先初始化schedule，然后读取

*~~1这个工序对应的这个工件是第几道工序work piece ProcesId（1，10），每遇到这个工件之后他都得自加，~~*

*~~2呢是是这个工序的可开工时间，或者说是这个工件的上道工序结束时间work pieceCanStartTime（1，10）~~*

*~~然后进入循环~~*

*~~For i=1：100~~*

*~~首先要知道是那个工件WorkpieceId，知道是第几次work piece ProcesId（i）++，知道是要用哪个机器machId，知道花费的时间~~*

*~~然后开始安排：~~*

*~~找到要用的机器的所有安排，单独拉出来newSchedule，~~*

*~~插入时往后顶，就是让工件的加工结束时间尽量紧挨着机器的 空闲结束时间~~*

*~~如果这个数组是空的，那就好办，第一个插入进去从Cmax插进去，然后break；~~*

*~~如果不是空的，那就是有安排了，就找所有的空闲时间，先看安排表的第一行，如果能插进去，就插。这里记得是从Cmax 开始插，从结束的地方开始插，但是呢，他又有工序先后的约束。这时候还要对比工件的上道工序的结束时间。那就是Cmax-max（上道工序结束时间，机器最后一段的空闲开始时间）>=这道工序的花费时间 才能插入。~~*

*~~插不进去的话，开始遍历每个间隔行，能插，就是说这段机器的空闲结束时间-max（工件的上道工序的结束时间，机器的空闲开始时间）>=这道工序的花费时间~~*

*~~插进去然后设置bool表示已经插了，然后break这个循环。~~*

*~~循环结束之后，只剩下了最后一行和空白的间隔了，这时候，检查bool类型看插没插，插了的话就break，没插就继续直接插入最后一行。~~*

全错咯 浪费一个晚上 直接猪脑过载

为了保持工序的顺序不变 应该逆序的遍历染色体 ，并且

1这个工序对应的这个工件是第几道工序work piece ProcesId（1，工件数），应该初始化为工序数，每加工一次就自减一次

2是这个工序的可开工时间应该以他的可结束时间为要求！！！或者说是这个工件的下道工序开始时间work pieceCanEndTime（1，工件数），在没下道工序的时候，也就是这是最后一道工序的时候应该是Cmax时间，所以初始化为Cmax

从后到前遍历这个染色体

For i=length：-1：1

首先要知道是那个工件WorkpieceId，知道是第几次work piece ProcesId（i）--，知道是要用哪个机器machId，知道花费的时间

然后开始安排：

找到要用的机器的所有安排，单独拉出来newSchedule，

插入时往后顶，就是让工件的加工结束时间尽量紧挨着机器的空闲结束时间

如果这个数组是空的，那就好办，第一个插入进去从Cmax插进去，然后break；

如果不是空的，那就是有安排了，就找所有的空闲时间，先看安排表的第一行，如果能插进去，就插。这里记得是从Cmax 方向插，从结束的地方开始插，但是呢，他又有工序先后的约束。这时候还要对比工件的下道工序的开始时间。这时候，下道工序的开始时间>=机器的空闲的开始时间+这道工序耗费的时间

插不进去的话（else），开始遍历每个间隔行，就是从第二行遍历到最后一行，每行与上一行的差，就是间隔行，能插，就插进去，插进去的意思是 机器的空闲开始时间+花费的时间<=min(机器空闲的结束时间，机器下一道工序的开始时间)

插进去就让 这道工序的结束时间= min(机器空闲的结束时间，机器下一道工序的开始时间)。插进去，就设置bool表示已经插了，然后break这个循环。

先检查bool，如果插了，就break这个大循环，如果没插，说明间隔行里也插不进去，那就放在最后一行，直接让工件的结束时间=min（空闲的结束时间，工件下道工序的开工时间）

唉 又错了 打了一上午的断点

应该这样：

在检查完NS不空之后，应该从倒数第一行开始检查，检查完之后入伙能插，就插，差不了，开始逆序的遍历NS，遍历能插，则插，差不了，直接插入第一行，

好，现在终于得到了另一个方向的排序

### 4 找到关键路径和关键块

关键路径好找，只要对比两个解码方式生成的时间表的异同，相同的就是关键路径。

*~~目前对关键块的定义还比较模糊，暂且认为他是关键路径上连续的、大小大于等于二的块。~~*

*~~这不操作也不难，主要是MATLAB好像是没有指针的，只能用i指向数组的某一个元素当作伪指针~~*。

找到了他的定义了，发现之前的是错的了，关键路径上在同一台机器上加工的相邻的工序序列构成了关键块。关键块中至少 包含两个工序。

这里学到了一个语法糖：

eyBlock = keyBlock(~cellfun('isempty', keyBlock)); 这行代码执行了一个操作，用于移除 eyBlock = keyBlock(~cellfun('isempty', keyBlock)); 这行代码执行了一个操作，用于移除 keyBlock 中的空单元格keyBlock 中的空单元格。

1. cellfun('isempty', keyBlock)：cellfun 函数用于应用一个函数（这里是 'isempty' 函数，用于检查单元格是否为空）到一个单元格数组 keyBlock 的每个元素上。结果是一个逻辑数组，其中每个元素对应于 keyBlock 中的一个单元格，如果单元格为空，则相应的逻辑值为 true，否则为 false。
2. ~ 运算符：这是 MATLAB 中的逻辑否运算符，用于翻转逻辑数组中的 true 和 false 值。~ 将 true 转换为 false，将 false 转换为 true。
3. keyBlock(~cellfun('isempty', keyBlock))：这一步使用逻辑数组来筛选 keyBlock 中不为空的单元格。只有在逻辑数组中相应位置为 false（表示单元格不为空）的元素才会被保留，而 true（表示单元格为空）的元素将被排除。

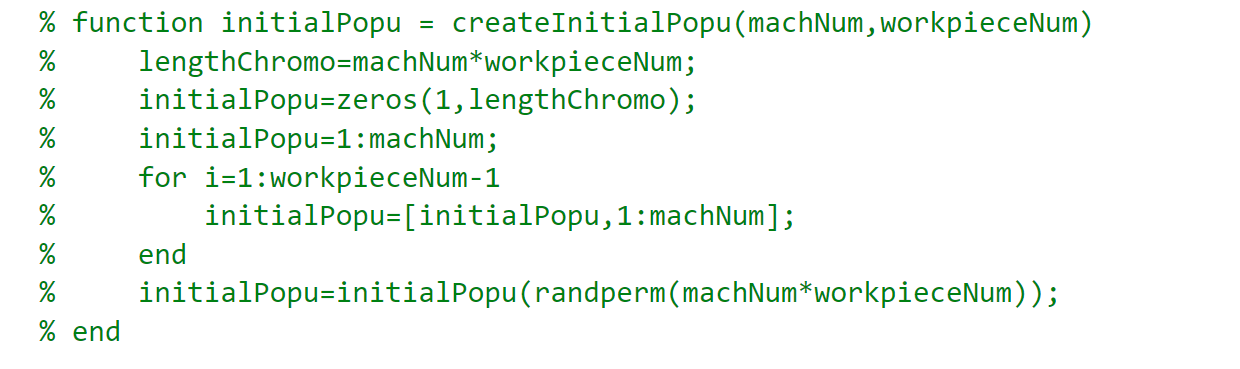
### 5 对关键块进行N5邻域解构操作

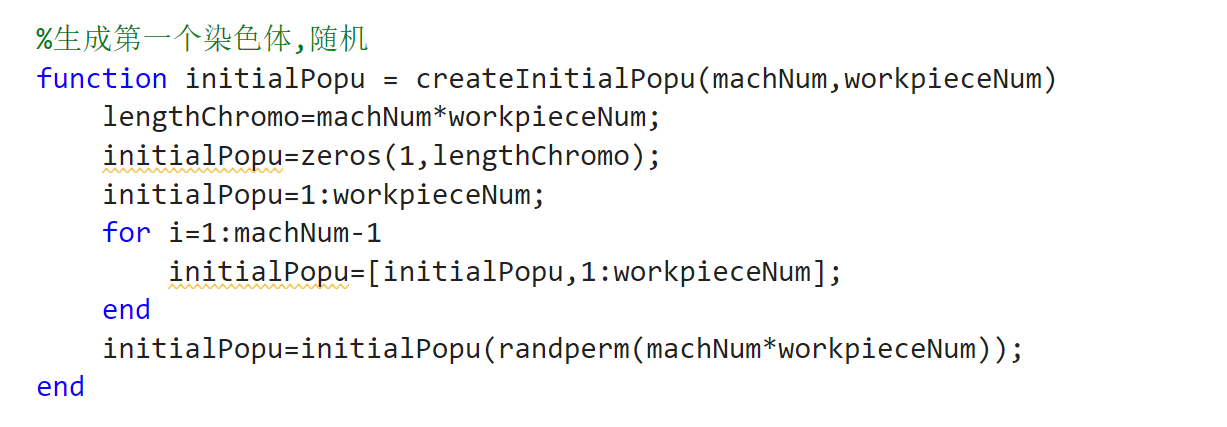
现在找到了几个关键块，输入到这个创造邻域的函数中，首先检查输入的块的个数。

在这些邻域结构中，N5 是最简单、最容易实现的邻域结构。N5 邻域结构只交换关键块最前面的两个工序以及关键块最后面的两个工序，并且不考虑关键路径上第一个关键块最前面的两个工序，也不考虑关键路径上最后一个关键块最后面的两个工序。

## 二、改错

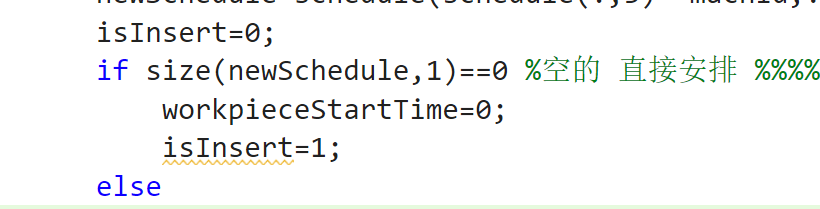
### 1 错误一

这个是周一上的那个新生研讨课发现的BUG，画GANT图本来是想让程序画出来的，结果把算例输入进去竟然出错了。原来一直用的是FT10的算例，工件数和工序（机器）数是相等的，所有machNum和workpieceNum设置的比较随意，然后上课用了这个2\*3和3\*3的例子，发现了几处错误，以生成随机种群的函数为例：

改成了：

这样就对了

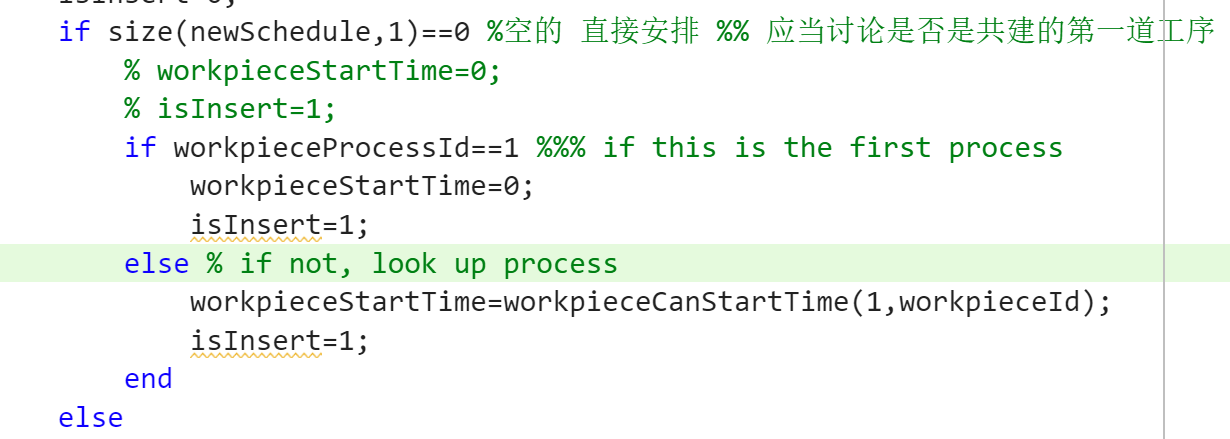
### 2 错误二

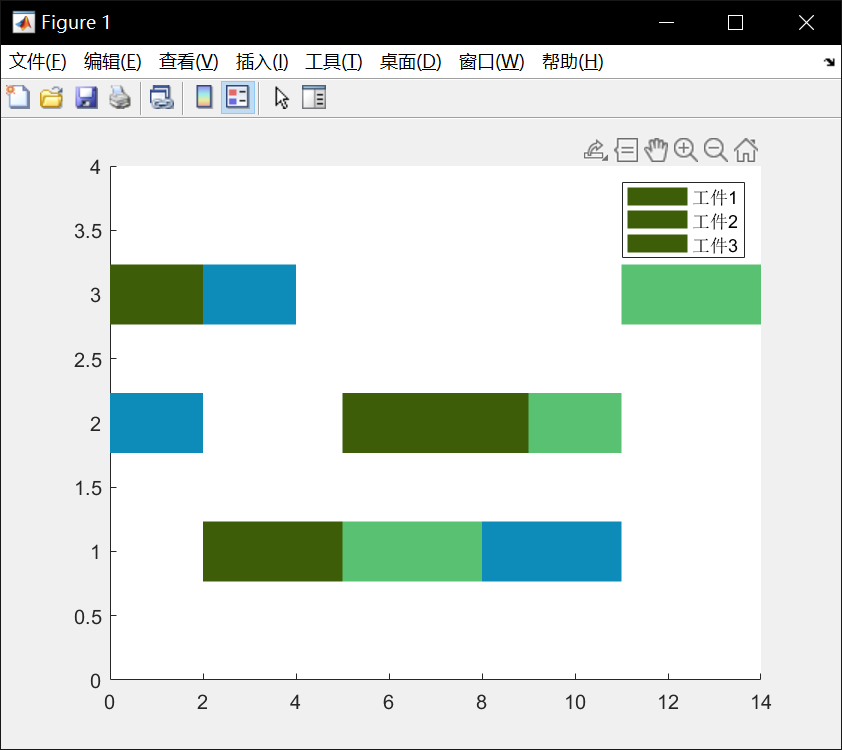
这个也是在算例较少的情况下发现的，有一部分的工件的工序没有受到先后的约束，因为都是在机器第一道工序的时候出现的错误，应该是在解码的时候出现的错误，经过检查是这里错了：

这里应该不止检查这道工序是不是机器的第一道工序，还应检查是不是工件的第一道工序。

改正为：

如果schedule表为空，就是机器还在空闲 此时检查工件的状态，

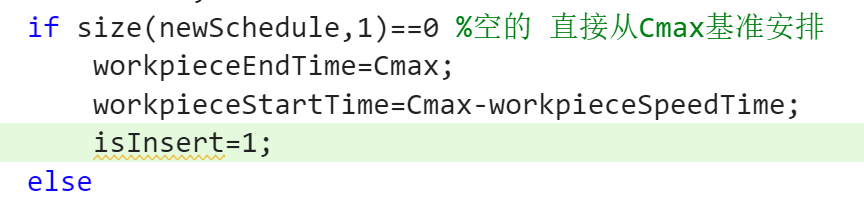
1. if这也是工件的第一道工序，直接安排，
2. else 不是第一到工序，安排到上一道工序结束后

改成这样就正常了：

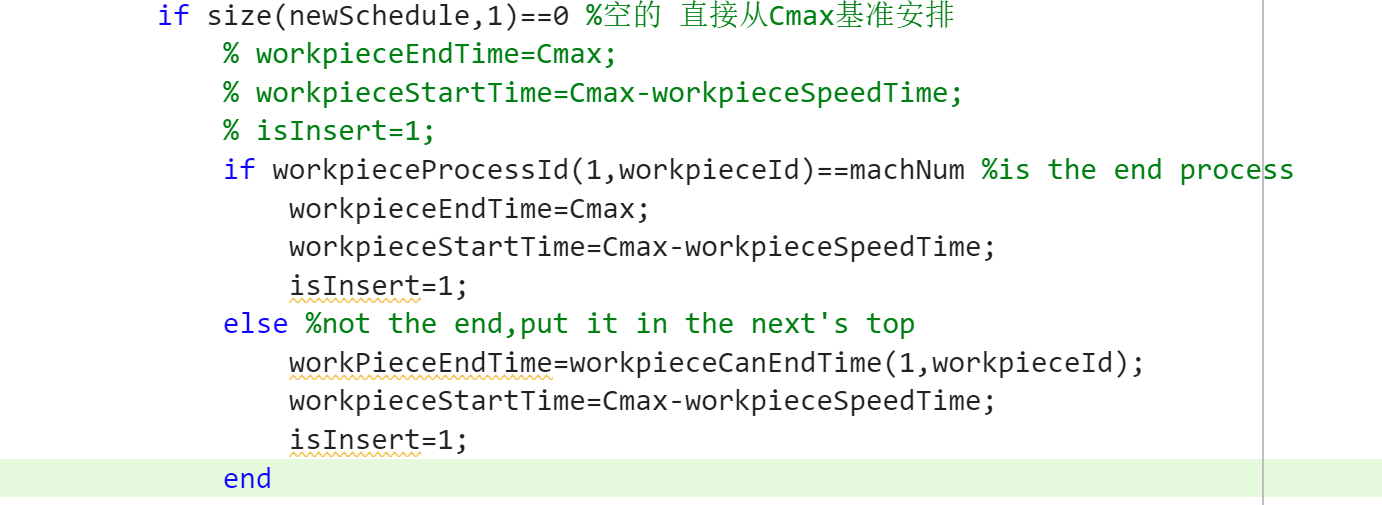
但是不知道为什么他这个图例又出现问题了。。。

这个先不改了

以后有空再看

然后这个是和后边查关键路径的一样，也应该变了；

这里边是从Cmax方向开始的，不只检查目前的机器是否空闲，还要见检查工件的状态：  
 1 if工件是最后一道工序，直接安排

 2 else 不是最后一道工序，安排到工序的下一道工序的开始时间

## 引入禁忌搜索

### 1 禁忌搜索步骤

禁忌搜索是对每个解多次搜索

步骤如下：

0.初始化禁忌表。  
1.解码，再逆解码，对比，生成关键路径和关键块。

2.对每个关键块进行N5邻域结构操作，生成所有的邻域解。

3.对每个邻域解进行评价，检查邻域解最优解是否是历史最优解，如是，记忆。

~~4.检查邻域最优解是否在禁忌表内，如否，用此最优解替换本代解，如是：检查是否满足特赦原则，如不满足，则不操作，如满足，则用该解替换本代解，并将该解加入禁忌表。~~

4.对邻域解进行排序，从上到下一次遍历每一个邻域解：

4.1 检查该解是否在禁忌表内，若在，继续遍历。若不在，使用改解替换本代解，并跳出遍历循环。并将该解加入禁忌表中。

5. 将所有的禁忌长度-1，之后将禁忌长度=0的行删去。

6.检查是否满足终止规则，如满足，终止，如不满足，继续从步骤1开始。

7.从最终解和历代最优解选择适应度更好的替换全局解并输出。