答卷编号（竞赛组委会填写）：

答卷编号（竞赛组委会填写）：

论文题目：柔性车间生产调度问题的排班优化

（C 题）

参赛队员：

1.姓名：刘彦莹　　学院：交通运输工程学院　学号：1109160512

电话：15773270362

2.姓名：牛旭　　学院：机电工程学院　学号：0801160918

电话：18774063035

3. 姓名：曹鸣佩　　学院：机电工程学院　　学号：0801161205

电话：18131162762

柔性车间生产调度问题的排班优化

摘要

车间调度生产问题(Job-shop Scheduling Problem)的排班优化属于典型的NP-hard 问题，即在确定产品加工工序、产品工序对应的加工机床和加工时间的情况下，找出机床利用效率最高、从而使生产周期最短的排版方案。而本文研究的柔性车间生产调度问题(Flexible Job-shop Scheduling Problem)是在 JSSP 的基础上，放宽约束条件，允许产品加工工序互换顺序，即其加工工序存在并列情况，且一组产品工序可对应多台加工机床，从而使问题复杂化。

本文就柔性车间生产调度问题的排班优化方案进行分析研究，针对如何缩短生产周期、最大化机床的利用率，分别建立基于蒙特卡罗法和遗传算法的数学模型，并运用数学软件 MATLAB 进行求解。

对于问题一，首先随机确定产品工序所对应的加工机床，再将车间中产品的加工过程转化为矩形的排列问题，其中机床代表放置矩形的管道，高度代表产品工序加工所对应的时间，矩形代表产品的工序。根据产品各个工艺过程与车床的对应关系、产品加工时间等建立约束条件，运用“贪心算法”使矩形在满足调度约束条件的情况下，尽可能的放置在接近底部的位置，以达到最大高度最小化的目标。为实现上述过程，设定机床选择的加工产品的评判标准，即优先级，以确定不同产品工序的加工顺序。

对于问题二：由于需要考虑某一产品提前完工的情况，可修改优先级的评定方式，使在满足所有约束条件及“贪心沉底”的情况下，需要提前完工的产品的优先级高于其余产品。

对于问题三：本文运用遗传算法来寻找近似最优解。首先根据柔性车间生产调度的已知条件设定遗传算法中的编码策略、适应度函数、交叉因子、变异因子等遗传过程中的构造因素，而初始种群则选取问题一中的较优解，以保证父代基因的优良性，使种群在较短的繁殖代数内找到满意的近似最优解。当考虑产品数量及机床数量增多的情况时，即相当于染色体上基因数目增多，而其余条件并不发生改变。根据模型得出最大完工时间为 59358 工时。

关键词：柔性车间生产调度　NP-hard 问题　蒙特卡洛法　遗传算法

一、问题重述

在机械制造业中，一个车间需要负责制造的不同类型的产品由若干道加工工序构成，而每道加工工序需要在特定的数控机床上完成。如何运用有限的数控机床，在一定的时间内完成生产任务，并尽可能地缩短生产周期，提高生产效率，是影响企业在该行业中竞争力的重要因素。

围绕如何制定合理的车间生产计划，以有效地提高设备利用率，进而缩短生产周期，本题要求用数学建模的方法依次研究以下问题：

（1）设计并求解该优化排班问题的算法。

（2）在问题（1）建立的模型的基础上，依次讨论每件类产品需要提前交货的情况下，生产计划的更改安排。

（3）建立具有普适性的数学模型，考虑产品需求量和机床总数增加的情况，设计求解不同条件下的生产方案。

二、问题分析

本题要求设计合理的车间生产方案，约束条件有产品-工序与机床间的对应关系、产品的需求量、产品加工的工序顺序，目标函数为最小化生产周期。由于产品的实际加工时间以及机床的加工效率为已知固定值，目标函数可在模型中转换为最小化一个生产周期内机器的空闲时间。

对于问题（1）：直接考虑机床上的加工安排过于抽象，可将车间中产品的加工过程具象化为矩形的排列问题，并根据产品各个工艺过程与车床的对应关系、产品加工时间等建立约束条件。然后设定不同情况下机床选择的加工产品的评判标准，即优先级，以确定不同产品工序的加工顺序。

对于问题（2）：由于需要考虑某一产品提前完工的情况，可修改优先级的评定，使在满足所有约束条件及满足第一优先级条件的情况下，需要提前完工的产品的第二优先级高于其余产品。

对于问题（3）：运用遗传算法寻找近似最优解，利用柔性车间生产调度的已知条件设定遗传算法中的编码策略、适应度函数、交叉因子、变异因子等。当考虑产品数量及机床数量增多的情况时，及相当于染色体上基因数目增多，而其余条件并不发生改变。

三、模型假设

（1）当某一产品的某道加工工序可以在不同机床上完成时，各个机床的加工时间、切换时间等不存在差异。

（2）某一类产品的某一工序在机床上的实际加工时间仅由单品加工时间和切换时间决定，而不受其他因素，如产品在转换工序时在不同机床上的中途转移时间等的影响。

（3）每台机床一次只能加工一个产品。

（4）同种类型的产品在某一机床上加工一道工序时，其加工数量必须达到需求量才能停止加工，不允许出现将一类产品拆分成两组或多组分开进行加工。

（5）每一台机床的加工效率均为同一固定常数。

四、符号说明

符号	意义
Q	需求量
t_C	试切工时
t_M	加工工时
η	加工效率
d	加工天数
i	产品种类号
j	工序序号
T	实际加工时间
ES	最早开始时间
EF	最早结束时间
k	机床序号
P	产品
M	机床
PRI_i	优先级
R	剩余实际加工时间
N	剩余工序数
D	当前加工时间
K	产品工序对应的可用机床数
H	矩形上边距底部高度

五、模型的建立与求解

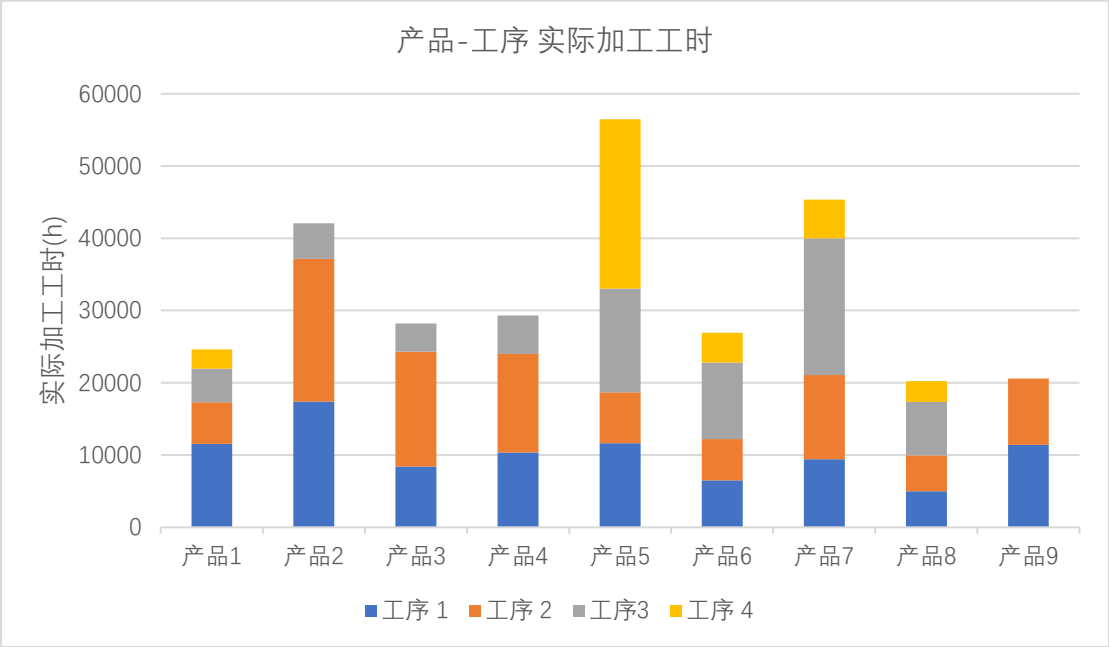
5.1 模型准备

5.1.1 实际加工工时

实际加工工时表示一类产品的一道工序从开始试切到全部加工完成所需工时。根据假设（2），可得出在产品恰好满足需求量的情况下，第 i 类产品的第 j 道工序所需的实际加工工时：

$$T(i,j)=t_C(i,j)+[Q(i)-1]\times t_M(i,j)$$

由上述公式计算得产品及工序对应的所需加工时间如下表：



5.1.2 机床加工效率

由附件中所给 2017 年 7 月生产计划，根据卓科 JKC-13090 机头转台 1（100）的加工时间为 9 天，由上表知卓科 JKC-13090 机头工序 1 的实际加工工时为 14335h，根据假设（5），由公式

$$\eta=\frac{T}{\mathrm{d}}$$

计算出每台机床的加工效率为 1592.78 工时/天。

5.2 基于蒙特卡洛法的柔性车间生产调度方案

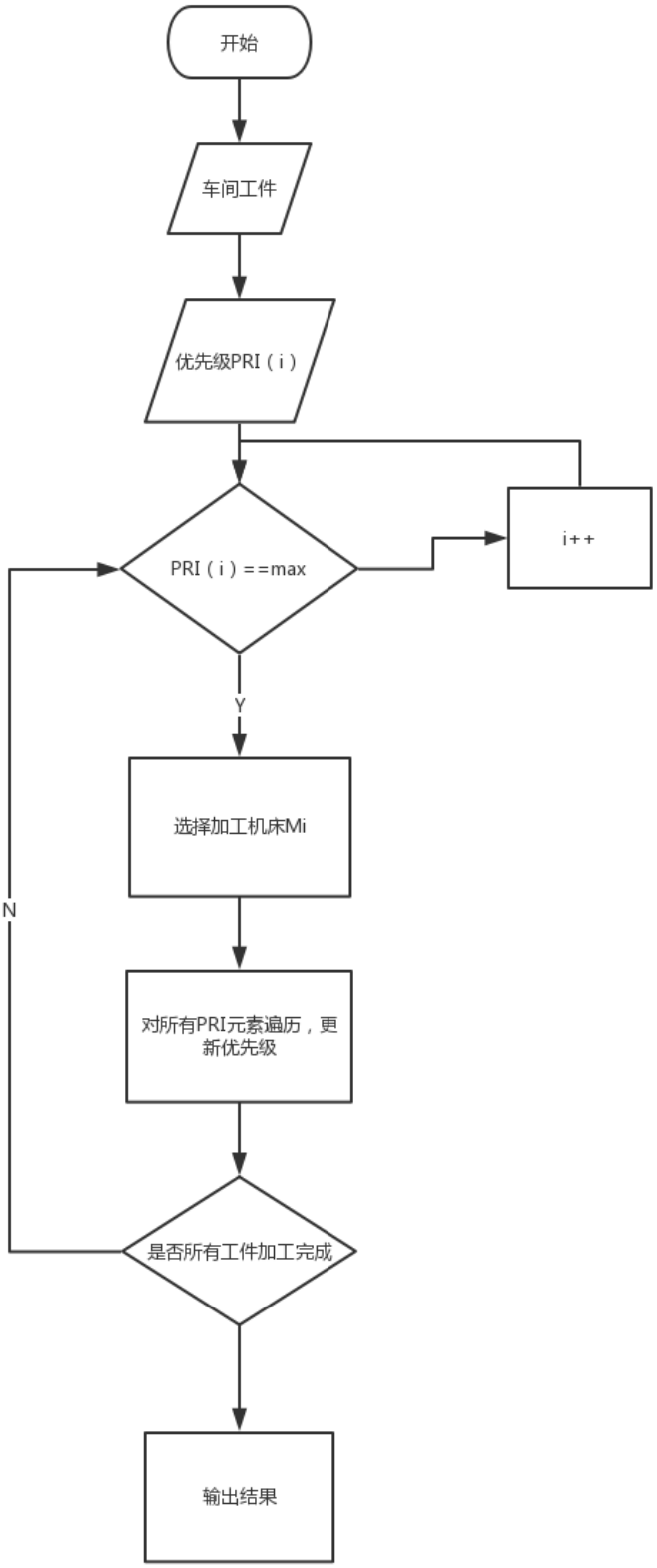
5.2.1 模型概述

运用蒙特卡洛法的思想，使产品工序在满足生产约束关系的情况下，随机选取对应的可加工机床，即先确定产品工序与加工机床的一一对应关系，再依据设定的优先级判断每台机床上的产品工序的先后加工顺序，即可得出相应的生产计划方案。

从所得的生产计划方案中找出满足最后一件产品工序的加工时间小于额定

生产周期的解即可。

流程图如下所示：



伪代码如下：
for（循环 1000000 次）

alist=random %生成随机数表

while(工序数<5)
t=优先级+工件制造时间；
for(对于每个工件循环)
update(PRI)； %更新 PRI 的值
工序数+1

If（max（t）<max（reserve））
reserve=t   %记录最小值

5.2.2 蒙特卡洛方法

（1）原理：

蒙特卡罗方法是用随机试验的方法计算积分，即将所要计算的积分看作服从某种分布密度函数 $f(r)$ 的随机变量 $g(r)$ 的数学期望

$$< g >= \int_0^{\infty} g(r) f(r) dr$$

通过某种试验，得到 N 个观察值 r_1，r_2，…，r_N（用概率语言来说，从分布密度函数 $f(r)$ 中抽取 N 个子样 $r_1, r_2, \cdots, r_N$，），将相应的 N 个随机变量的值 $g(r_1)$，$g(r_2)$，…，$g(r_N)$ 的算术平均值

$$\bar{g}_N = \frac{1}{N} \sum_{i=1}^{N} g(r_i)$$

作为积分的估计值（近似值）。

（2）步骤：

一、构造或描述概率过程

对于本身就具有随机性质的问题，如粒子输运问题，主要是正确描述和模拟这个概率过　程，对于本来不是随机性质的确定性问题，比如计算定积分，就必须事先构造一个人为的概率过程，它的某些参量正好是所要求问题的解。即要将不具有随机性质的问题转化为随机性质的问题。对于本题，将 FJSP 问题转化为随机路径的 JSP 问题，将最大完工时间的最小值作为求解参量，使得非随机问题随机化。

二、实现从已知概率分布抽样

构造了概率模型以后，由于各种概率模型都可以看作是由各种各样的概率分布构成的，　因此产生已知概率分布的随机变量（或随机向量），就成为实现蒙特卡罗方法模拟实验的基本手段。本题采用计算机调用的伪随机数列。三、建立各种估计量

对于本题，由于 JSP 解空间解集个数过多，且边界不便于明确，因而在此不讨论其数理统计量，但对于本文测试次数 1000000 次，其得到接近最优解的概率超过 95%。

5.2.3 生产约束关系

（1）“产品工序-设备”关系约束

根据不同产品的每道工序所需的加工机床不同，建立对应关系矩阵来描述其对应关系。

（2）工序顺序约束

将每一类产品的加工工序均视为 4 道，当其实际加工工序为 3 道时，取其加工时间为 0 即可。

第 i 类产品的第 j 道加工工序时间满足如下条件：

$$EF(i,j)=ES(i,j)+T(i,j)$$

当产品的第 j 道加工工序与第 j+1 道加工工序存在先后顺序关系时，其第 j+1 道工序的最早开始时间必须满足如下条件：

$$ES(i,j+1)\geq EF(i,j)$$

当产品的第 j 道加工工序与第 j+1 道加工工序存在并列关系时，令其随机选取第 j 道或 j+1 道工序。则有：

$$ES(i,j+2)\geq \max\{EF(i,j),EF(i,j+1)\}$$

（3）需求量约束

由于产品工序的加工时间是根据不同产品的需求量计算而来的，无需再额外考虑需求量的约束条件。

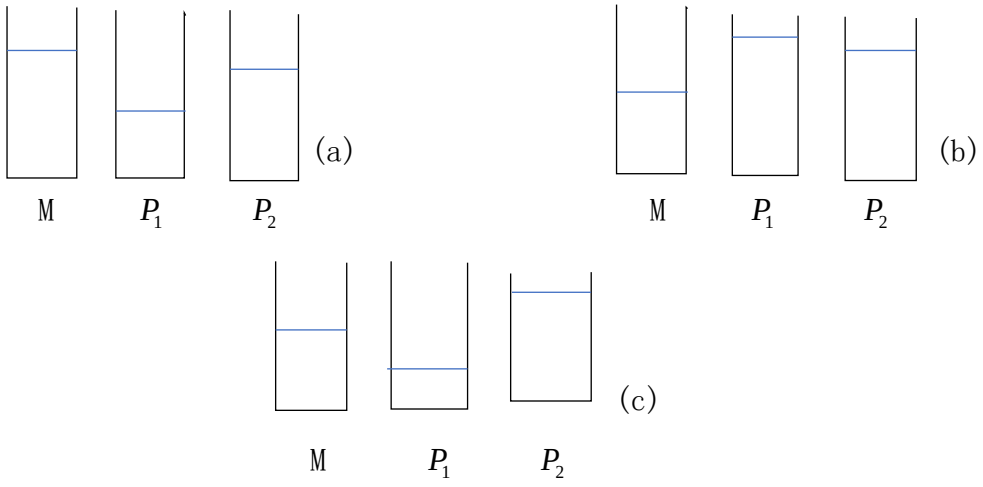
5.2.4 优先级的设定

对于如何确定机床加工固定产品工序的先后顺序，需先建立优先级的评判标准。

本文根据是否生产过程中存在需要提前加工完成的产品，对优先级的设定进行以下分类讨论：

（1）所有产品处于平等地位，无需提前完成每类产品的加工。

假设目前有 $P_1,P_2,\cdots,P_n$ 这 n 件产品（图中仅表示两件产品）的下一道工序都需要在机床 M 上进行加工，则可将情况分为以下三种类型：



上图表示产品工序在机床上的加工过程。用 H 表示图形中线段至底面的高度，其实际意义为产品的某道工序在机床上的结束时间。

（a）产品 P_1, P_2 的工序结束时间均早于机床 M 上正在加工的产品工序的结束时间。

（b）产品 P_1, P_2 的工序结束时间均晚于机床 M 上正在加工的产品工序的结束时间。

（c）产品 P_1 的工序结束时间早于机床 M 上正在加工的产品工序的结束时间，产品 P_2 的工序结束时间晚于机床 M 上正在加工的产品工序的结束时间。

设定第一优先级：

$$PRI_1(i) = -\max\{H_M, H_i\}$$

当 $PRI_1(i)$ 的数值越大时，其代表的优先级越高。

该式可以解决（b）、（c）两种情形下的优先次序判定：当所有产品的工序的结束时间均晚于机床 M 上正在加工的产品工序的结束时间时，越早结束当前加工过程的产品优先级越高；当只有一个产品的工序的结束时间早于机床 M 上正在加工的产品工序的结束时间时，其优先级最高。

而在（a）情况下，存在两类或两类以上产品的工序结束时间早于机床 M 上正在加工的产品工序的结束时间，需设定第二优先级[1] 来判断产品工序的加工优先次序：

$$PRI_2(\mathrm{i}) = \frac{N_i \cdot \sum_{j=1}^{4} T(i,j)}{D_i}$$

当 $PRI_2(i)$ 的数值越小时，其代表的优先级越高。

该式表示第二优先级与剩余工序数及加工总时长成正比，而与当前加工工时成反比。

（2）某一类产品应尽早提前完成加工。

此类情况下依旧存在上述（a）、（b）、（c）所述情形。

第一优先级的选取不发生改变：

$$PRI_1(i) = \max\{H_M, H_i\}$$

第二优先级由产品是否为需要提前完工的产品类型决定：

$$PRI_2 = \begin{cases} 1 & P(i) = *(i) \\ 0 & P(i) \neq *(i) \end{cases}$$

其中*(i)代表需要提前完工的特殊产品类型。

第三优先级依旧选取情况（1）中的第二优先级：

$$PRI_3(\mathrm{i})=\frac{N_i\cdot\sum_{j=1}^{4}T(i,j)}{D_i}$$

5.2.5 模型的求解

表一-“贪心算法”得出的机床加工安排时间表

时间\机器	A	B	C.	D	E	F	G	H
7月1日	7		1	5		4	2	3
7月2日	7		1	5		4	2	3
7月3日	7		1	5		4	2	3
7月4日	7		1	5		4	2	3
7月5日	7		1	5		4	2	
7月6日	7		1	5		4	2	
7月7日	9	7	1	5			2	
7月8日	9	7	3	5			2	
7月9日	9	7	3	5			2	
7月10日	9	7	3	5			2	
7月11日	9	7	3	5			6	
7月12日	9	7	3	2			6	
7月13日	9	7	3	2			6	
7月14日	9	5	3	2			6	
7月15日	9	5	3	2	6		6	
7月16日	9	5	3	2	6		8	
7月17日	9	5	3	2	6		8	
7月18日	9	5	7	2	3		8	
7月19日		5	7	2	3			8
7月20日	7	5	7	2	3			8
7月21日	7	5		2				8
7月22日	7	8	5	2				
7月23日	7	8	5	2				
7月24日	7	1	5	2		2		
7月25日	7	1	5	6		2		
7月26日	7	1	5	6		2		
7月27日	7	4	5	6		1		
7月28日	7	4	5	6		1		
7月29日	7	4	5	6		1		
7月30日	7	4	5	6		1		
7月31日	7	4	5	4		1		
8月1日	7	4	5	4				
8月2日	8	4	5	4				

8月3日	8	4	5	4				
8月4日	8	6	5	4				
8月5日	8	6	5	4				
8月6日		6		4				
8月7日		6		4				

表二

	最大完工时间	最大提前交工时间
1	63553	18.79
2	61630	9.8
3	63518	21.55
4	61631	15.49
5	59640	1.99
6	64071	18.27
7	61075	4.4
8	62548	19.6
9	61806	25.8

5.3 基于改进型遗传算法的柔性车间工作调度方案

5.3.1 模型概述

改进型遗传算法以基本遗传算法为基础，结合柔性生产工作的具体情况，对遗传算法中的编码、交叉、变异进行改进，将编码作为工序与机器选择方案，利用自然界中优胜劣汰的基本原则，对较差结果进行淘汰；引入交叉和变异机制，避免陷入局部最优解的情况[2]。

5.3.2 目标函数与约束条件

本文的调度目标由单一目标构成——最小化生产周期，也即最小化最大完工时间：

$$z=\min\{\max_{1\le j\le q_i}\{EF(i,j)\}\}$$

而在生产调度过程中满足的约束条件与模型 5.2 中的约束条件相同。

5.3.3 编码设计与解码

在传统遗传算法中解决 Job-shop 问题时，常采用一串编码方式同时记录工序与零件的选择方式。但对于柔性车间调度问题，其工件不仅仅可以在单一车床上加工，这也就造成了问题的复杂化。因此，为解决此项问题，本文采用两套子串进行编码[3]。

第一子串表示各个产品、工序所对应的机器编号。

第二子串表示调度次序。

具体表述如下：

(1)第一子串

工件\工序	①	②	③	④
J1	B/C/D	B/C/D	F/G	F/G
J2	F/G	B/C/D	F/G	A
J3	E/F/G/H	B/C/D	E/F/G/H	A
J4	E/F/G	B/C/D	B/C/D	A
J5	A/B/C/D	A/B/C/D	A/B/C/D	B/C/D
J6	E/F/G/H	E/F/G/H	B/C/D	A/B/C/D
J7	A/B/C/D	A/B/C/D	B/C/D	A/B/C/D
J8	E/F/G/H	E/F/G/H	B/C/D	A/B/C/D
J9	A/B/C/D	A/B/C/D	A	A

第一子串和解码子串存储了相关信息。对于本题，解码子串由1X36 的元胞数组组成，其每部分包含的向量元素表述不同工件不同工序的组成情况。第一子串保留可使用的机器数的随机值，通过第一子串与解码子串可以等可能选择机器进行加工。

（2）第二子串

第二子串由随机数进行初始化，得到 1X36 维且每工件数均出现工序数次，通过比较从子串起始到所求位置的元素格数,即可得到所处工序顺序情况。例如：[1，2，1，3，3，2，1，2，3]其中第 3 个元素 1 表示工件 1 进行到第 2 道工序。

5.3.4 初始种群

将模型 5.2 中所求得的较优解作为初始种群中的个体,以保证父代的基因较为优良，能在较少的繁殖代数内找到符合条件的近似最优解。同时引入部分完全随机解，避免死锁在局部优化问题中，而导致无法向更优解集靠近。

5.3.5 选择策略

选择过程表示采用一定的选择策略,将父代中适应度较好的个体保留进入子代中，子代再在父代的基础上进行交叉和变异，以搜素父代附近的局部最优解，以保证子代的总体适应度优于父代的总体适应度。

由于目标函数为最小化生产周期，即个体所对应的目标函数值越小，其所对应的适应度函数值应当越高。故选取目标函数的倒数作为适应度函数[4] ：

$$f(x)=\frac{1}{z(x)}$$

其中 x 代表生产计划方案。

对于本题情况而言，由于目标函数是最小化生产周期，在求解问题时,FJSP 问题转化成为 JSP 问题，JSP 问题是一个 NP-hard 问题，采用的方法将决定收敛速度的快慢，因此在本题中，选用第一、二问“贪心算法”的方式，利用贪心算法快速获得较优的解。

5.3.6 交叉操作

交叉过程需要如下满足条件：

（1）交叉后所得的子代染色体为一可行解；

（2）交叉后所得子代染色体满足调度的约束条件。

对两列子串的具体交叉过程如下：

（1）第一子串交叉

第一子串由各个产品、工序所对应的加工机床编号进行编码构成，而不同产品工序对应的可加工机床编号在染色体上的位置固定，即在理论上保证了交叉后染色体所对应的调度方案的可行性，在保证随机性的条件下，必然满足调度约束条件。

对于第一子串的交叉方式，本文采用多点交叉法。先随机生成一组由二进制数组成的随机码，父代中位置与该随机码中数值为“0”的位置相同时，父代中的基因进行互换，而其余位置的基因保持不变，据此得到子代染色体。

（2）第二子串交叉

本文的第二子串交叉选用 POX 交叉，具体操作过程如下：

首先随机将所有需加工工件分为两部分 J_1, J_2 ，当子代 C_1，C_2 分别继承父代 P_1，P_2 中 J_1 部分的所有基因和对应位置，而子代 C_1 中剩余基因由父代 P_2 中 J_2 部分的基因按顺序插入，同理，子代 C_2 中剩余基因由父代 P_1 中 J_2 部分的基因按顺序插入。

5.3.7 变异操作

变异是改善算法的局部搜索功能，在全局范围内寻找近似最优解的关键。

（1）第一子串变异

第一子串由可加工机床的编码构成，故每个基因都可根据变异概率从当前数值变异为 0 到 $K(i, j)-1$ 之间的数值。显然变异后的染色体依旧满足调度约束条件。

（2）第二子串变异

为了保证变异后染色体依旧满足调度约束条件，使染色体代表一可行的生产调度方案，本文优化染色体变异的方式，不采取单个基因独自变异的情形，而采用两个父代染色体交换两位置的基因的方式，从而有效避免了复杂的由于单个基因改变而造成的后续基因变动的修复和重建工作。

5.3.8 模型的求解

运用 matlab 进行求解，改进遗传算法得出的机床加工安排时间表如下：

时间\机器	A	B	C.	D	E	F	G	H
7 月 1 日	5	1	7		5	2		3
7 月 2 日	5	1	7		5	2		3
7 月 3 日	5	1	7		5	2		3
7 月 4 日	5	1	7		5	2		3
7 月 5 日	5	1	7		5	2		3

7月6日	5	1	7	3	5	2		3
7月7日	5	1	5	3	6	2		3
7月8日	7	9	5	3	6	2		3
7月9日	7	9	5	3	6	2		3
7月10日	7	9	5	3	6	2		3
7月11日	7	9	5	3		8		3
7月12日	7	9	5	3		8		3
7月13日	7	9	5	3		8		3
7月14日	7	9	5	3		8		3
7月15日	9	8	5	3		6		3
7月16日	9	8	5	4		6		3
7月17日	9	8	7	4		6		3
7月18日	9	5	7	4		8	1	
7月19日	9	5	7	4		8	1	
7月20日	9	5	6	4		8	1	
7月21日	7	5	6	4		1	2	
7月22日	7	2	6	4		1	2	
7月23日	7	2		4			2	
7月24日	7	2		8			2	
7月25日	7	2					2	
7月26日	7	2					2	
7月27日	7	2					2	
7月28日	7	2					2	
7月29日	7	2					2	
7月30日	7	2					2	
7月31日	7	2					2	
8月1日	8	2					2	
8月2日	8	2					2	
8月3日	8	2					2	
8月4日							2	
8月5日							2	
8月6日							2	

六、模型评价与推广

6.1 模型评价：

本题应用主要有两个模型：蒙特卡罗方法的贪心算法模型与改进遗传算法。从总体上看，两个模型取得了不错的成果，同时也展示出不同模型的优缺点。下面对优缺点进行一般分析：

1.蒙特卡罗方法的贪心算法

优点：

（1） 对于特定 JSP 问题（即固定加工路线）拥有较高的精确度以及快速

的收敛时间，适合对于精度要求不高且要求快速响应的问题。

（2） 采用蒙特卡罗方法，通过对最大完工时间的比较，能够保留较优的路线，减少由于其局部最优方式造成的影响。

（3） 较为简单的处理方式以及良好的适应性。

缺点：

（1） 对于 FJSP 问题响应速度过慢，由于其局部最优解与路线完全随机的特点，造成需要比对大量数据以得到较优结果，影响其速度。因此在得到本题结果的情况下，所需时间为 1367s。

（2） 贪心算法具有局限性，容易进入局部最优状态而无法获得更优解，因此，贪心算法所得结果较难进一步获得更优解。

2.改进遗传算法

优点：

（1）遗传算法对解的选择大大减少了复杂度，使得迭代次数快速减少，本题达到稳定情况所需代数约为 100 代，进行 10 次对遗传算法的计算，所用时间为 1173s。

（2）具有优秀的全局搜索能力，通过变异、交叉等方式，保证了在一定情况下对全局的搜索。

（3）具有并行性，可以多个个体同时比对，缩短了迭代时间。

缺点：

（1）对初始种群依赖程度较高，初始种群的质量决定了遗传算法迭代代数的多少。

（2）需要较多的参数，变异率、交叉率，其参数品质严重影响解的品质，然而其选择目前多依靠经验，不具有较为成熟的可靠性指标。

（3）编程实现较为复制，首先对问题分析并进行解码和提出相关方式确定参数时对使用者有较高要求。

（4）对于本题，由于其评价函数采用的是最大完工时间的倒数，因而其评价函数的处理和编写也较为复杂。

6.2 模型推广

下面考虑建立具有更广普适性的数学模型：对于具有m台设备，n个零件，p道工序的情况，利用改进遗传算法依然能够有效解决。其算法复杂度为多项式。对于子串，需要设定其维度为1X（n*p）,其余方法与上述求解基本相同。

参考文献

[1]黄文奇,邓泽林. Job-Shop Scheduling问题的一个快速算法[J]. 株洲工学院学报,2003,(02):38-40. [2017-08-07].

[2]朱文凡. 基于遗传算法的多柔性作业车间调度问题研究[D].合肥工业大学,2014.

[3]方剑,席裕庚. 基于遗传算法的 Job Shop 静态调度算法[J]. 上海交通大学学报,1997,(03):51-54. [2017-08-07].

[4]张超勇,董星,王晓娟,李新宇,刘琼. 基于改进非支配排序遗传算法的多目标

柔性作业车间调度[J]. 机械工程学报,2010,46(11):156-164. [2017-08-07].

附录

“贪心算法”程序代码：

```
clc,clear
load 'connection.mat'
load 'mj-con.mat'
load 'time.mat'
for i = 1:10000
x_ran = randi([0,1],1,8);            %Ëæ»ú½»»»±äÁ¿
mat = zeros(36,8);
alist = [pro_list(W1),pro_list(W2),pro_list(W3),pro_list(W4)];
alist(alist==0)=1;
bar = [4 3 3 3 4 4 4 4 2];
eqa = zeros(3,9);                    %ÓÅïÈ¼¶Ë³Ðò±í
t = zeros(36,8);                     %Ê±¼ä°²Å±í
%ÖØÒª¬ÇëÖ¸¨³õÊ¼Öµ


input_eqa2=4;
eqa(2,input_eqa2)=1;%êÇ·ñÓÐÐèÒªìáÇ°½»¹¤


%ÖØÒª¬ÇëÖ¸¨³õÊ¼Öµ
seq = ones(1,9);            %¹¤Ðò²½ÖèÊý


sheng = zeros(1,9);
for char = 1:9
var = eval(['t',int2str(char)]);
sheng(char) = (product(char)-1)*sum(var(2,:))+sum(var(1,:));
end                                  %Ê£Óà¹¤Ê±³õÊ¼»¯
eqa(3,:)=sheng*4;
all_pro = sheng;


for m = [1,4,5]
if isequal(x_ran(m),1)
    ran_var = eval(['t',int2str(m)]);
    eval(['t',int2str(m),'=[ran_var(:,[1,3,2,4])];']);
end
end
```

```
for n = [6 7 8]
if isequal(x_ran(n),1)
    ran_var = eval(['t',int2str(n)]);
    eval(['t',int2str(n),'=[ran_var(:,[1,2,4,3])];']);
end
end
%Ëæ»ú½»»»


while(min(eqa(1,:))<300000)


%
index1=find(eqa(1,:)==min(eqa(1,:)));
[~,index3]=max(eqa(3,index1));


if(length(index1)~=1)
    if(any(eqa(2,:))&&length(find(eqa(2,:)~=0))==1&&eqa(1,input_eqa2)==min(eqa(1,:)))
        rechar=input_eqa2;
    else
        rechar=index1(index3);
    end
else
rechar=index1;
end
%ÓÅïÈ¼¶±È½ï·½·¨,recharîªÑ¡Ôñ¶ÔÏó


machine = alist(rechar,seq(rechar));%ËùÓÃ»úÆ÷
var = eval(['t',int2str(rechar)]);      %×îÓÅïÈÁã¼þ¼Ó¹¤¾ØÕó
var_sp = var(:,seq(rechar));            %×îÓÅïÈÇé¿ö¼Ó¹¤¾ØÕó£¨ÊÔÇÐ£¬¼Ó¹¤£©
list_time=find(t(:,machine)==0);
new_list = list_time(2);                %Ñ°ÕÒµ½µÚ¶þ¸öîªÁãµÄË÷Òý
pro=var_sp(1)+var_sp(2)*(product(rechar)-1);
t(new_list,machine)=eqa(1,rechar)+pro;%t±í¸üÐÂ
no = find(mat(:,machine)==0);
suo = no(1);                            %Ñ°ÕÒµ½µÚÒ»¸öîªÁãµÄË÷Òý
mat(suo,machine) = rechar;      %¼ÇÂ¼°²ÅÅË³Ðò
sheng(rechar)=sheng(rechar)-pro;    %¸üÐÂÊ£Óà¹¤Ê±
seq(rechar)=seq(rechar)+1;              %¸üÐÂ¹¤Ðò
%
for j = 1:9
evar = eval(['t',int2str(j)]);
```

```
if seq(j)>4
eqa(1,j)=inf;                                %íê³ÉËÄ²½¹¤ÐòÓÅÏÈ¼¶±äî³inf
elseif seq(j)==1
will_use = alist(j,seq(j));                  %¼´½«Ê¹ÓÃµÄ»ú´²
eqa(1,j)=max(t(:,will_use));
elseif eqa(3,j)==0
eqa(1,j)=inf;
seq(j)=4;
else
used = alist(j,seq(j)-1);                    %Ñ°ÕÒµ½Ö®Ç°Ò»´ÎµÄÊ¹ÓÃ»ú´²
elist=find(mat(:,used)==j);                  %ÕÒµ½¸Ã»ú´²
eindex=elist(end)+1;                         %ÕÒµ½¸Ã»ú´²×îºóÒ»¸öÔªËØ
evar_sp = evar(:,seq(j)-1);                  %Ö®Ç°¹¤ÐòµÄÊ±¼ä
will_use = alist(j,seq(j));                  %¼´½«Ê¹ÓÃµÄ»ú´²
show = evar_sp(1)+evar_sp(2)*(product(j)-1);      %¹¤×÷Ê±¼ä
eqa(1,j)=max(max(max(t(:,will_use))),t(eindex,used)); %T1ÓÅÏÈ¼¶¸üÐÂÍê³É
eqa(3,j)=all_pro(j)*(bar(j)-seq(j)+1)/show;                %¼ÇµÃshengÒªÔÚ¸üÐÂºó±»¯
end


end
end
%eqaÓÅÏÈ¼¶¸üÐÂ·½·¨


if max(max(t)) < max(max(final_t))
    final_t = t;
    final_mat = mat;
    final_alist=alist;
end
end
final_t
final_mat
```

“遗传算法”程序代码：

```
%GAÖ÷³ÌÐò
clc,clear
data_save; %¼ÓÔØÊý¾Ý
accept=zeros(1,50);
Point=zeros(1,50);
P_c=0.4;   %½»²æ¸ÅÂÊ
P_m=0.1; %±äÒì¸ÅÂÊ
gen=1;      %µü´ú´ÎÊý
answer_min = 0;
```

```
for chr=1:50
eval(['save_random',int2str(chr),'=zeros(1,36);']);        %±£Áôëæ»úÊý¾Ý    µÚÒ»×Ó´®
eval(['save_machine',int2str(chr),'=zeros(1,36);']);        %±£Áô»úÆ÷Ê¹ÓÃÊý¾Ý
eval(['judge_machine',int2str(chr),'=zeros(2,36);']);     %±£ÁôÅÐ¶ïÊý¾Ý
for i = 1:36
    eval(['save_random',int2str(chr),'(i)','=randi([0,num_mac(i)-1]);']);
end
for i =1:36
    p = mac{i};
    h = eval(['save_random',int2str(chr),'(i)'])+1;
    eval(['save_machine',int2str(chr),'(i)','=p(h);']);
end
save_line =
process();                                                    %±£ÁôµÚ¶þ×Ó´®±àÂë
eval(['judge_machine',int2str(chr),'(1,:)','=save_line;']);          %µÚÒ»ÐÐ±£Áô¼Ó¹¤¹¤Ðò
judge_machine(1,:)µÚ¶þ×Ó´®
for i = 1:9
    index = find(save_line == i);
    cool=eval(['save_machine',int2str(chr),'(1,4*i-3:4*i)']);
    eval(['judge_machine',int2str(chr),'(2,index)','=cool;']);
end


end


for mar = 1:50
    p = eval(['save_random',int2str(mar)]);
    q = eval(['judge_machine',int2str(mar),'(1,:)']);
    eval(['Ulist',int2str(mar),'=p;']);
    eval(['Vlist',int2str(mar),'=q;']);
end


while(gen<70)    %µü´ú´îÊý¿É¸Ä±ä
for die = 1:50
accept(die)=targetfun(eval(['judge_machine',int2str(die)]));
if accept(die)> answer_min
    answer_min = accept(die);
    judge_machine = eval(['judge_machine',int2str(die)]);
end
end
```

```
aldie=sum(accept);
for str=1:50
Point(str)=accept(str)/aldie;
end                          %µÃµ½Ã¿¸öÏà¶Ô¸ÅÂÊ
Q=cumsum(Point); %ÂÖÅÌ¸ÅÂÊ
for kar = 1:50
eval(['Aex',int2str(kar),'=Ulist',int2str(kar)]);
eval(['Bex',int2str(kar),'=Vlist',int2str(kar)]);
end                                  %½»»»ÐòÁÐ
for dish=1:50
    lun = rand(1,1);
    lun_index=find(lun<Q);
    lun_need = lun_index(1);
    eval(['Ulist',int2str(dish),'=Aex',int2str(lun_need)]);
    eval(['Vlist',int2str(dish),'=Bex',int2str(lun_need)]);
end
%ÐÂÖÖÈº¸´ÖÆÍê³É


num_first = floor(P_c*50);   %½»²æÌõÊý
yan=randi(2,1,36)-1;
yan_in = find(yan==0);
make = round(rand(1,num_first)*49)+1;
for i = 1:10
father1=make(2*i-1);
father2=make(2*i);
for j = yan_in
father1_ex=eval(['Ulist',int2str(father1),'(j)']);
father2_ex=eval(['Ulist',int2str(father2),'(j)']);
eval(['Ulist',int2str(father1),'(j)','=father2_ex;']);
eval(['Ulist',int2str(father2),'(j)','=father1_ex;']);
end
end
%µÚÒ»×Ó´®½»²æ·½Ê½


sui = randi(2,1,36)-1;
base = unidrnd(9);
make = round(rand(1,num_first)*49)+1;
for i = 1:10
all_ex1=1:36;
all_ex2=1:36;
father1=make(2*i-1);
father2=make(2*i);
```

```
Vlist_f1=eval(['Vlist',int2str(father1)]);
Vlist_f2=eval(['Vlist',int2str(father2)]);
base_index1=find(Vlist_f1==base);
base_index2=find(Vlist_f2==base);
all_ex1(base_index1)=[];
all_ex2(base_index2)=[];
for i = 1:32
eval(['Vlist',int2str(father1),'(all_ex1(i))','=Vlist_f2(all_ex2(i))']);
eval(['Vlist',int2str(father2),'(all_ex2(i))','=Vlist_f1(all_ex1(i))']);
end
end
%µÚ¶þ×Ó´®½»²æ·½Ê½




for count =1:50
for inner = 1:36
if rand(1,1)<P_m
    eval(['Ulist',int2str(count),'(inner)','=unidrnd(num_mac(inner))-1;']);
end
end
end
%µÚÒ»×Ó´®
for count =1:50
for inner = 1:36
if rand(1,1)<P_m
    a =unidrnd(36);
    b =unidrnd(36);
    ta = eval(['Vlist',int2str(count),'(a)']);
    tb = eval(['Vlist',int2str(count),'(b)']);
    eval(['Vlist',int2str(count),'(a)','=tb;']);
    eval(['Vlist',int2str(count),'(b)','=ta;']);
end
end
end
%±äÒì


for last = 1:50
    eval(['judge_machine',int2str(last),'(1,:)','=Vlist',int2str(last)]);
    for i =1:36
    p = mac{i};
    h=eval(['Ulist',int2str(last),'(i)'])+1;
```

```
        eval(['save_machine',int2str(last),'(i)','=p(h);']);
        end
        for j=1:9
            list_index=find(eval(['judge_machine',int2str(last),'(1,:)'])==j);
            lop = eval(['save_machine',int2str(last),'(4*j-3:4*j)']);
            eval(['judge_machine',int2str(last),'(2,list_index)','=lop']);
        end
    end


    %Éú³ÉÅÐ¶Ï¾ØÕó
    gen=gen+1;
    end
```