基于改进的遗传算法求解 P-FJSP 线性动态规划

摘要

本文针对机械公司部分柔性作业生产调度问题，基于改进的遗传算法，条件约束，建立线性动态规划模型，对不同产品、不同工序在不同机床的调度方案进行了优化，分别给出在给定约束时间条件下单批次和多批次调度的最短生产周期规划方案。在日渐复杂的市场背景下，进一步提高了企业生产高质量产品的效率和质量。

针对问题一，本文建立基于 P-FJSP 线性动态规划模型，运用遗传算法进行求解最短生产周期规划方案。首先，以最短生产周期为目标函数，并以产品各工序可使用机床的信息和同产品工序之间的关系为约束条件建立**线性动态规划模型**。之后，采用**遗传算法**进行求解。最后在所有求解中找出最优解，**最短生产周期为 62733 分钟，约为 44 天**。

针对问题二，可以在问题一的基础上增加保证规定产品提前交货的目标函数，建立多目标线性规划模型，采用**改进的遗传算法**进行求解。首先，我们利用 σ **约束法**将多目标规划问题转化为单目标规划问题，利用引入**惩罚项**的遗传算法进行求解，得到最优生产计划方案。以**产品 1、2、3 交货时间为 30 天**、其余产品交货时间为 60 天为例，代入模型进行求解得到生产完成时间分别为 **24653 分钟**、**43021 分钟**、**31217 分钟**。并且由于产品存在生产时间下限，在单批次情况下某些产品无法在 30 天内规定完成，比如产品 4.

针对问题三，可以将单批次车间调度改进为多批次车间调度，从而提高机床利用率，并在提高利用率的同时，求解最短生产周期规划方案。首先，将每个产品进行**分批生产**，将每个批次作为一个新的产品处理，并利用新的数据信息带入问题一和问题二中的模型中进行求解。以每个产品进行双批次处理，并平均分配需求量，在问题一和问题二的模型中运行结果。最终得到的**最优生产周期为 66241 分钟，利用率为 0.6476**，产品 4 也能在 30 天内完成加工。

关键字：柔性车间调度　线性动态规划　改进遗传算法　分批调度

目录

1 问题重述

1.1 问题背景

制造业是我国国民经济的支柱产业，主导着我国经济的发展。近年来，制造业发展迅速，如何在最短的制造周期内生产出高质量的产品，并在最短的时间内交付给客户，是制造企业正在面临的重大考验。

面对日渐复杂的市场背景，为解决当前难题，公司应以客户的需求为根本依据，科学安排生产，改善优化车间的生产计划方案，减少停工时间，使生产设备利用率达到最大，提高产品的生产效率及交付的准时性。

本问题是一个经典的车间作业生产调度问题，又根据题目信息，产品的部分工序处于并列关系且同一工序可使用多个机床，属于部分柔性作业生产调度问题 $(P-FJSP)$.

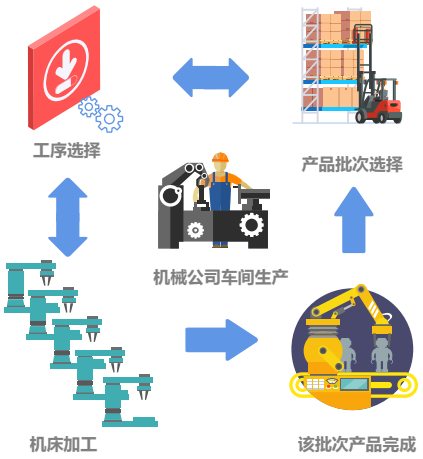

图 1: 机械公司车间生产安排背景图

1.2 问题提出

某机械公司需在 30 天的生产周期内，运用现有的 A-H 编号的 8 台机床，完成其负责的 9 中机械零件的加工。某一类产品的同道加工工序可用的机床及所需的时间相同，不同工序则不完全相同。在实际生产时，某种产品的需求量与月份以及产品的种类有关。车间需要建立合理的机床—产品加工安排表，在 30 天生产周期内，按时对给定需求量的这 9 种产品进行加工。

基于上述问题和附件信息，我们需要建立数学模型，研究并解决以下问题：

问题一：针对车床-产品的车间生产安排建立优化模型，并对该优化模型的算法进行构造求解。

问题二：在某产品需提前交付的情况下，应如何合理安排生产计划，以保证交货的准时性。

问题三：在保证机床充足的前提下缩短停工时间并提高利用率，在更短的生产周期中完成产品加工。建立合理的数学模型，使其满足产品或机床数目增加的情况，并设计求解，得出具有一般适用性的生产计划方案。

2 模型假设

1. 生产出的每个产品都满足质量要求，不存在次品及产品消耗；
2. 某一类产品的每道加工工序在不同机床进行时的单品加工时间和切换时间全部相同；
3. 每道工序之间可能存在先后顺序关系，即该产品在完成某一道工序后才能开始下一道工序，也可能存在并列关系，即该产品在某一道工序完成后可以同时进行两道不同的工序或先开始两道工序中的其中一道；
4. 每类产品都作为同一批次进行各道工序的加工，直至这批产品加工完成；
5. 产品加工时间只考虑单品加工时间和切换时间的影响。

3 符号说明

符号	说明
n	产品类型数量
m	工序数
w	现有数控机床台数
t_{ij}^0	第 i 个产品的第 j 个工序的切换时间
t_{ij}^1	第 i 个产品的第 j 个工序的加工时间
S_{ijk}	第 i 个产品的第 j 个工序在第 k 个机床上加工开始的时间
F_{ijk}	第 i 个产品的第 j 个工序在第 k 个机床上加工完成的时间
N_i	产品 i 的需求量
T_{ij}	加工 N_i 个第 i 个产品的第 j 个工序的需要加工的总时间
Q_{ijk}	第 i 个产品的第 j 个工序是否可以在第 k 台机床上进行
x_{ij}	第 i 个产品的第 j 个工序与其他工序的关系
M	无穷大
ε	无穷小
TL_i	产品的交货时间
λ_{i}	第 i 个产品的批次
T	单批次加工的生产周期
T'	多批次加工的生产周期
β	分批前后产品总加工时间变化
$\bar{\alpha}$	设备利用率

注：未列出的以及重复的符号均以首次出现处为准.

4 问题一的模型建立与求解

4.1 问题一的描述与分析

本问题要求根据给定的各类产品的加工需求量，为一个机械公司的车间建立一个优化模型，以在 30 天的生产周期内，合理安排 8 台数控机床（A-H）的生产计划，以满足 9 种不同类型产品的生产需求。产品的加工工序，加工时间，以及工序间的先后顺序和并列关系都已经给出。在这个问题中，主要的难点在于如何建立一个能够满足所有约束条件的优化模型。这些约束包括但不限于：每种产品的加工需求量，每种产品需要经过的加工工序，每道工序可以在哪些机床上进行，以及工序之间的先后顺序或并列关系。此外，由于每台机床一天内只能加工一种产品，因此也需要考虑到切换产品所需的时间。最后，根据问题条件和附件信息，对生产过程进行建模，将生产过程抽象为产品工序在机床的实时分配问题，将多目标规划问题转化为单目标动态整数规划问题，以保证规定的产品提前交货期和产品的总生产周期最短。最后，采用遗传算法对模型进行拓扑，包括编码策略、初始种群、适应度函数、选择、交叉和变异操作等步骤，得到最优的生产调度方案。

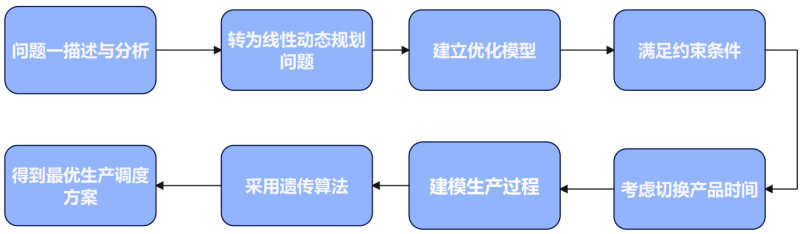

图 2: 问题一思维导图

4.2 基于 P-FJSP 问题的线性动态规划模型

本问题要保证实现生产周期最短，就要合理安排每一个产品工序的加工时间，是一个多阶段决策过程，本文针对该问题建立线性动态规划模型。

4.2.1 决策变量

(1)t_{ij}^{0} 代表第 i 个产品的第 j 个工序的切换时间;

(2)t_{ij}^{1} 代表第 i 个产品的第 j 个工序的加工时间;

(3)S_{ijk} 代表第 i 个产品的第 j 个工序在第 k 个机床上加工开始的时间;

(4)F_{ijk} 代表第 i 个产品的第 j 个工序在第 k 个机床上加工完成的时间;

(5)N_i 代表产品 i 的需求量;

(6)M 代表无穷大;

(7)ϵ 代表无穷小;

(8)Q_{ijk} 代表第 i 个产品的第 j 个工序是否可以在第 k 台机床上进行;

$$Q_{ijk} = \begin{cases} 0 \text{ 第 i 个产品的第 j 个工序不能在第 k 台机床上进行 ;} \\ 1 \text{ 第 i 个产品的第 j 个工序能在第 k 台机床上进行 .} \end{cases} \tag{1}$$

(9) $\mathcal{C}_k$ 代表第 k 台机床产品工序加工的实时完成时间;

(10) 代表第 i 个产品的第 j 个工序是否已完成;

$$\mathrm{d_{ij}} = \begin{cases} 0 \text{ 第 i 个产品的第 j 个工序未开始 ,} \\ 1 \text{ 第 i 个产品的第 j 个工序已完成 .} \end{cases} \tag{2}$$

(11)T_{ij} 代表加工 N_i 个第 i 个产品的第 j 个工序的需要加工的总时间

表 1: 产品工序使用时间表

	工序 1	工序 2	工序 3	工序 4
产品 1	11553	5769	4657	2616
产品 2	17380	19770	4905	0
产品 3	8410	15885	3902	0
产品 4	10370	38778	9386	0
产品 5	11655	7043	14355	23410
产品 6	6480	5735	10585	4125
产品 7	9405	11665	18910	5375
产品 8	4975	4975	7395	2875
产品 9	11430	9155	0	0

(12)x_{ij} 代表第 i 个产品的第 j 个工序与其他工序的关系

$$x_{ij} = \begin{cases} 0 \text{ 产品的该工序不存在 ,} \\ 1 \text{ 与其他工序都是顺序关系 ,} \\ 2 \text{ 与其他工序存在并列关系 .} \end{cases} \tag{3}$$

表 2: 产品工序关系表

	工序 1	工序 2	工序 3	工序 4
产品 1	1	2	2	1
产品 2	1	1	1	0
产品 3	1	1	1	0
产品 4	1	2	2	0
产品 5	1	2	2	1
产品 6	1	1	2	2
产品 7	1	1	2	2
产品 8	1	1	2	2
产品 9	1	1	0	0

4.2.2 目标函数

本文将优化方案定义为设备利用率高、机床停工时间短的生产计划方案。将目标函数理解为最后完成所有既定产品所用生产总时间级生产周期的最短。

$$T = \min(\max(F_{ij})); i = 1, 2 \ldots \mathrm{n}; j = 1, 2 \ldots \mathrm{m}. \tag{4}$$

4.2.3 约束条件

（1）一个工件的一个工序一旦开始加工就不能中断，产品工序的完成时间必须等于工序的开始时间加上工序的加工时间。

（2）各类产品在进行某个工序时是作为同一批次，在同一个机器上加工。全部完成后在进入下一个工序。切换时间内含生产第一个产品所需时间，则加工时间：

$$T_{ij} = t_{ij}^{0} + t_{ij}^{1} \left(N_i - 1\right). \tag{5}$$

$$(i = 1, 2 \ldots \mathrm{n}; \mathrm{j} = 1, 2 \ldots \mathrm{m})$$

（3）各个产品的工序必须按照可使用机床信息来加工；

$$F_{ijk} = S_{ijk} + T_{ij} \times Q_{ijk}. \tag{6}$$

$$(i = 1, 2 \ldots \mathrm{n}; \mathrm{j} = 1, 2 \ldots \mathrm{m}; \mathrm{k} = 1, 2 \ldots \mathrm{w})$$

当 Q_{ijk}=1 时，说明第 i 个产品的第 j 个工序可以在第 k 台机床上进行，此时产品工序的完成时间等于工序的开始时间加上工序的加工时间；当 Q_{ijk}=0 时，说明第 i 个产品的第 j 个工序不可以在第 k 台机床上进行，此时可记作产品工序的完成时间等于工序的开始时间。

（4）记录第机床加工产品的实时完成时间

$$\text{if } d_{ij} = 1, C_k = \max\left(F_{ijk}\right) \tag{7}$$

（5）相同产品的部分工序之间存在顺序关系；

$$\begin{cases} S_{i(j+1)k'} + \frac{1}{M|\mathrm{x_{i(j+1)}}-\mathrm{x_{ij}}|+\varepsilon} \geq S_{ijk} + T_{ij}, \\ F_{i(j+1)k'} + \frac{1}{M|\mathrm{x_{i(j+1)}}-\mathrm{x_{ij}}|+\varepsilon} \geq F_{ijk} + T_{ij}, \\ i = 1, 2 \ldots \mathrm{n}; \mathrm{j} = 1, 2 \ldots \mathrm{m}; \mathrm{k}, \mathrm{k}' = 1, 2 \ldots \mathrm{w}. \end{cases} \tag{8}$$

当 $|\mathrm{x_{i(j+1)}} - \mathrm{x_{ij}}| \neq 0$ 时，说明此产品的这一步工序与下一步工序是顺序关系，必须前一步工序结束后才能进行下一步的工序。当 $|\mathrm{x_{i(j+1)}} - \mathrm{x_{ij}}| = 0$，说明此产品的这一步工序与下一步工序是并列关系，两者可同时进行，也可不同进行。

（6）每一时刻每一台机床只能加工一个工件的一个工序。

$$\begin{cases} \text{if } \left|x_{i(j+1)k} - x_{i(j+1)k''}\right| = 0, S_{i(j+1)k''} \geq \max\left\{\mathcal{C}_k, F_{i(j-1)k'}\right\}, \\ \text{if } \left|x_{i(j+1)k} - x_{i(j+1)k''}\right| \neq 0, S_{i(j+1)k''} \geq \max\left\{\mathcal{C}_k, F_{i(j-1)k'}\right\}. \end{cases} \tag{9}$$

4.2.4 模型总述

综上，产品调度最优短的模型如下

$$T = \min\left(\max(F_{ij})\right). \tag{10}$$

$$st. \begin{cases} T_{ij} = t_{ij}^0 + t_{ij}^1\left(N_i - 1\right), \\ F_{ijk} = S_{ijk} + T_{ij} \times Q_{ijk}, \\ \text{if } d_{ij} = 1, C_k = \max\left(F_{ijk}\right), \\ S_{i(j+1)k'} + \frac{1}{M|\mathrm{x_{i(j+1)}}-\mathrm{x_{ij}}|+\varepsilon} \geq S_{ijk} + T_{ij}, \\ F_{i(j+1)k'} + \frac{1}{M|\mathrm{x_{i(j+1)}}-\mathrm{x_{ij}}|+\varepsilon} \geq F_{ijk} + T_{ij}, \\ \text{if } \left|x_{i(j+1)k} - x_{i(j+1)k''}\right| = 0, S_{i(j+1)k''} \geq \max\left\{\mathcal{C}_k, F_{i(j-1)k'}\right\}, \\ \text{if } \left|x_{i(j+1)k} - x_{i(j+1)k''}\right| \neq 0, S_{i(j+1)k''} \geq \max\left\{\mathcal{C}_k, F_{i(j-1)k'}\right\}, \\ i, p = 1, 2 \ldots \mathrm{n}; \mathrm{j}, \mathrm{p} = 1, 2 \ldots \mathrm{m}; \mathrm{k}, \mathrm{k}', \mathrm{k}'' = 1, 2 \ldots \mathrm{w}. \end{cases} \tag{11}$$

4.3 遗传算法求解

遗传算法是一种基于自然选择原理和自然遗传机制的寻优算法，它是模拟自然界中“适者生存”这一生命进化机制，将问题求解表示为根据适者生存的原则逐代进化的过程，实质是通过群体搜索在人工系统中实现问题目标的优化，最终求得最优解或准最优解。

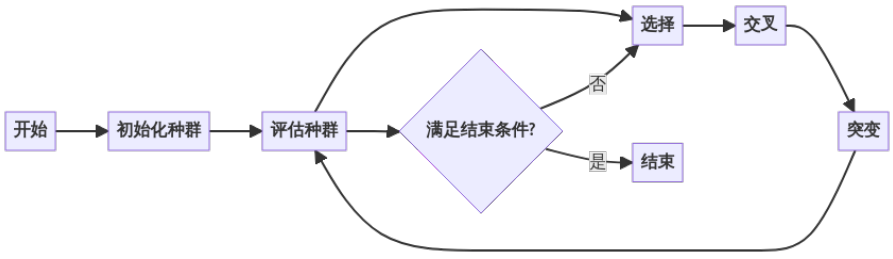

图 3: 遗传算法流程图

4.3.1 编码策略

对于遗传算法而言, 基因编码就是将问题的状态空间与算法的码空间相对应, 而编码的方式直接影响遗传算法的性能和效率以及能否找到全局最优解，具有至关重要的作用。遗传算法的编码方式目前有浮点数编码、二进制编码、整数编码、符号编码、矩阵编码等多种方式。

本文针对 JSP, 调度的最优解里要包括每个产品的各个工序分别在哪一台机床上进行加工，以及每个产品的工序加工时间节点的顺序 [1,2]。本文将染色体分为两个长度都为 n×m 的染色体片段。其中，前一个片段的每一个位置的基因表示每一个产品工序的编码，后一个片段的每一个位置的基因表示前面产品工序对应使用的机床，整个染色体片段代表产品工序-机床对，基因的排列顺序代表加工的优先顺序，每个产品的各个工序将根据优先顺序并结合产品可使用机床信息来选择机床加工。当对每个染色体样本的总时间进行计算时，会将采用百分位数编码进行数据存储，百位数代表产品，个位数代表工序，如 “101” 表示第 1 类产品的第 1 个工序。

4.3.2 初始种群

由于遗传算法操作的对象是群体，所以要有一个有若干个个体组成的初始种群。鉴于保优遗传算法的概率 1 收敛特性和提高算法的搜索效率，本文确定初始种群采用随机生成每个工序-机器对的值的方式。一方面，依据附件产品可使用机床信息，将初始种群设定整个问题空间的最优解的空间分布范围内；另一方面，随机生成若干个体，在其中挑选最优加入到初始种群中。

4.3.3 适应度函数

遗传算法判断个体优劣的依据是适应度高低。根据 “适者生存，不适者被淘汰” 原则 [4]，一般适应度越高，被遗传到子代的概率就越高。本文将目标函数作为适应度函数，本文将所有产品都按需求量完成的总时间即为最后一件的最后一道工序的加工完成的时间作为目标函数。要求

$$T = \min(\max(F_{ij})); i = 1, 2 \ldots \mathrm{n}; j = 1, 2 \ldots \mathrm{m}. \tag{12}$$

调用 cal 函数计算种群中每个个体的目标函数值，根据目标函数值对种群进行排序，计算适应度值，不断循环迭代来寻找最优解。

4.3.4 选择操作

对种群进行不断筛选, 使用轮盘赌选择方法从种群中选择个体用来进行交叉和变异操作。

4.3.5 交叉和变异操作

交叉是父代个体染色体两两进行基因重组，变异是小概率改变父代个体的染色体的部分基因，两者都是模拟生物的遗传方式生成新的子代个体的一种形式，本文采用单点交叉算子对选定的个体进行交叉操作，使用随机变异算子对选定的个体进行变异操作。通过交叉和变

异操作 [3]，可以维护群体中个体的多样性，是遗传算法在搜索时朝向最优解空间的方向进行，加速最优解的收敛性，并降低对有效模式的破坏概率。

4.4 结果分析

通过遗传算法求解得到随着迭代次数的增加种群均值和目标值的变化如下图 4所示。可以观察到随着迭代次数的增多，目标值越来越小，逐渐趋向于最优解空间，算法收敛。在求解过程中，经过多次迭代求得的最短的生产周期为 62733 分钟约为 44 天，具体生产调度方案如下图 5所示。

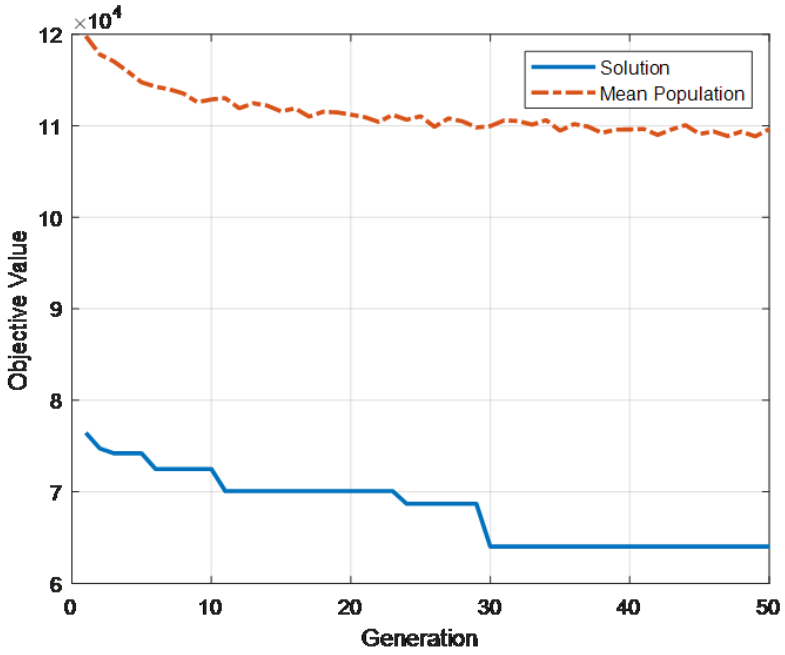

图 4: 种群均值和目标值变化图

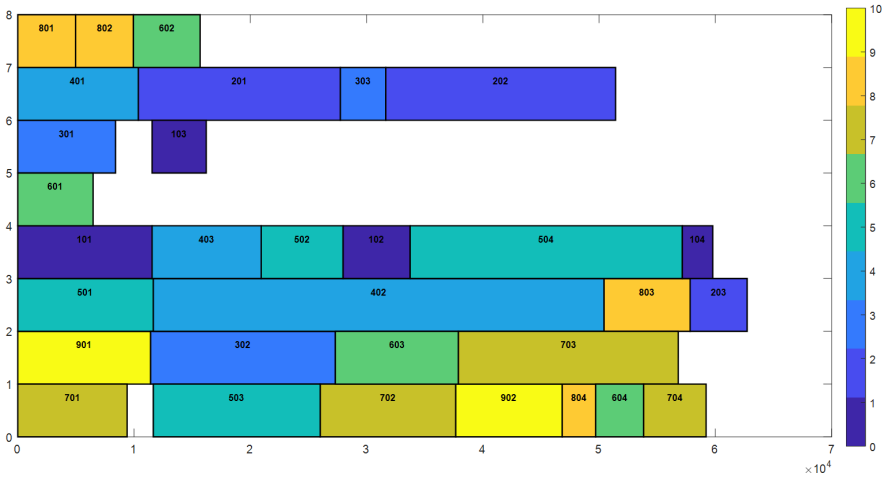

图 5: 生产调度甘特图

通过多次遗传算法测试，可以获得多个局部最优解，能够得出生产周期的最优结果区间为 [61000,63000]。

5 问题二的模型建立与求解

5.1 问题 2 的描述与分析

问题二在问题一的基础上引入一个新的约束条件：某些产品可能需要提前交货，代表必须在规定的日期之前完成这些产品的生产。因此，我们需要修改我们的优化模型，以便在满足所有已有约束的同时，还能满足这些产品的提前交货需求。本文中问题一的基础上，引入交货期指标，将其作为 ϵ 值进行约束，将多目标规划问题转化为单目标规划问题进行求解。最终，采用改进的遗传算法对模型进行仿真，得到最优化的生产调度方案。

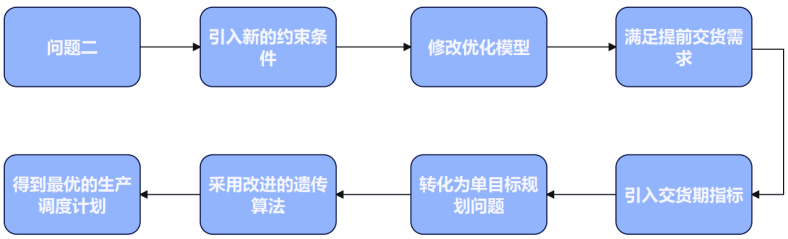

图 6: 问题二思维导图

5.2 基于 P-FJSP 的多目标整数规划模型建立

问题二在问题一的基础上新增了保证规定的产品提前交货的目标，是多目标规划问题。本文针对规定的产品所有工序的完成时间要先于该产品的交货时间和产品的总生产周期最短两个目标建立多目标整数规划模型。

求解多目标规划问题一般分为直接法和间接法两种 [5]。其中，直接法是直接对问题进行转化求解；间接法是将多目标规划问题转化为单目标规划问题进行求解。一般间接法是采用对目标函数进行线性赋权从而转化为综合目标函数进行求解，但基于本问题很大程度上会受主观因素的影响。

本文采用间接法中的 ϵ-约束法 [10]，原理是选取最重要的目标进行优化，将其他目标作为约束条件来进行求解。本文将生产周期最短作为主要目标，将规定的产品所有工序的完成时间要先于该产品的交货时间作为约束条件。在问题 1 模型的基础上，问题 2 引入交货时间变量 TL_i，将其作为 ϵ_i 的值进行约束，新增的约束条件为

$$\max(F_{ijk}) \leq TL_i (j = 1, 2 \ldots \mathrm{m}; k = 1, 2 \ldots \mathrm{w}). \tag{13}$$

其中，i 是需要提前交货的产品的序号。

5.3 改进遗传算法求解

基于问题一，本文采用的遗传算法引入了惩罚项 [7]，当规定的提前交货的产品超过交货日期时，给予生产周期 T 一个惩罚值，从而淘汰掉该个体，迫使生成的解满足完成时间先于交货时间这一约束条件。惩罚项的引入限制了解空间，使得算法能够更快地收敛到最优解，可以帮助算法更快地搜索到合适的解，并提高算法的鲁棒性和适应性。

5.4 数值仿真结果分析

假设产品 1,2,3 需要提前交货，提前交货时间 TP 都是 30 天（43200 分钟），同时保证其余产品要在 60 天之内完成（86400 分钟），利用上述修改模型求解得到的生产调度方案如图8所示。

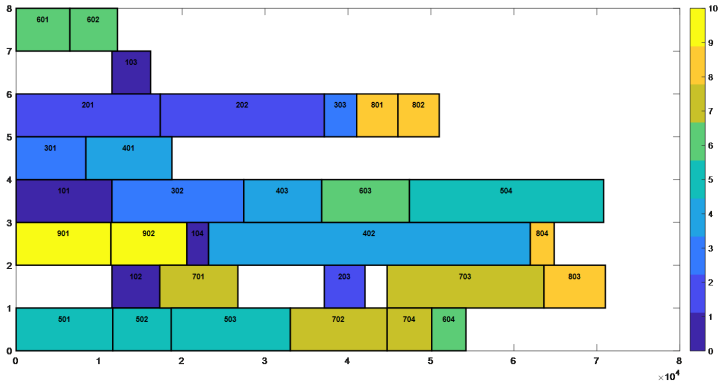

图 7: 数值仿真生产调度甘特图

根据结果可知产品 1 的生产完成时间为 24653 分钟，产品 2 的生产完成时间为 43021 分钟，产品 3 的生产完成时间为 31217 分钟，生产周期是 69560 分钟。三个产品的生产完成时间都小于交货时间，其余产品也在规定的 60 天之内完成，说明该优化模型有效。

但并不是所有设定的生产规定时间都能满足，在单批次的情况下，对于每个产品而言，由于工序的顺序关系，其加工完成的时间都会有下界，并且在考虑到多个产品有生产规定时间时，也可能出现无法满足要求的情况。比如将产品 4 规定在一个月之内完成，它的产品工序时间下限为 58534 分钟，在单批次情况下无法实现在三十天的期限内完成产品的加工。每个产品的工序时间下限如下表所示。

表 3: 产品工序时间下限表

产品 1	产品 2	产品 3	产品 4	产品 5	产品 6	产品 7	产品 8	产品 9
24595	42055	28197	58534	56463	26925	45355	20220	2058

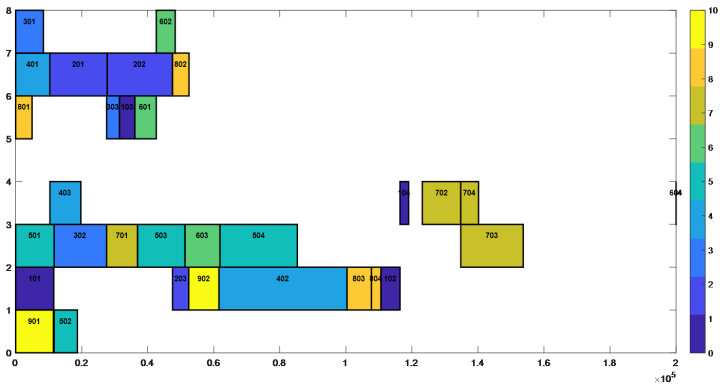

图 8: 单批次产品 4 无法实现产生的结果图

6 问题三的模型建立与求解

6.1 问题三的描述与分析

问题三要求减少机床的停工时间以实现高的设备利用率，同时能尽快完成产品加工以实现短的生产周期。考虑到在单批次情况下，所有产品完成每道工序所需时间是不变的，问题一构建的模型得出所求的生产周期为一个较稳定的范围值，利用率和生产周期成反比，提升空间较少。将产品分为多批次进行调度 [8]，从而实现产品分配的灵活性和机床使用的充分性。最后考虑问题二中需要提前交货的目标，通过分批调度实现原本单批次无法实现的产品加工。

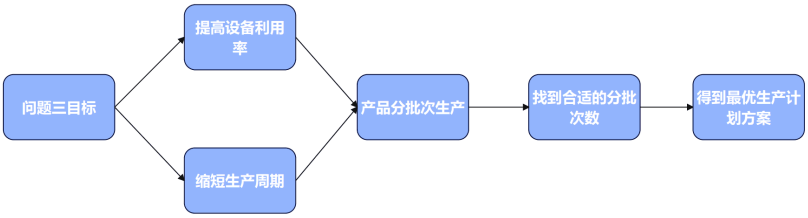

图 9: 问题三思维导图

6.2 多批次柔性车间调度模型

根据题目，本题既要求设备利用率高又要求的生产周期短。本文将所有机床的利用率的平均值作为设备利用率的度量值.

$$\bar{\alpha} = \frac{\sum_{k=1}^{w} \alpha_k}{w}, \tag{14}$$

其中，$\bar{\alpha}$ 表示设备利用率，α_k 表示第 k 台机床的利用率，w 表示现有数控机床台数. 并规定各机床的利用率是该机床工作时间与总的生产周期之比.

$$\alpha_k = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} (F_{ijk} - S_{ijk})}{T}, \tag{15}$$

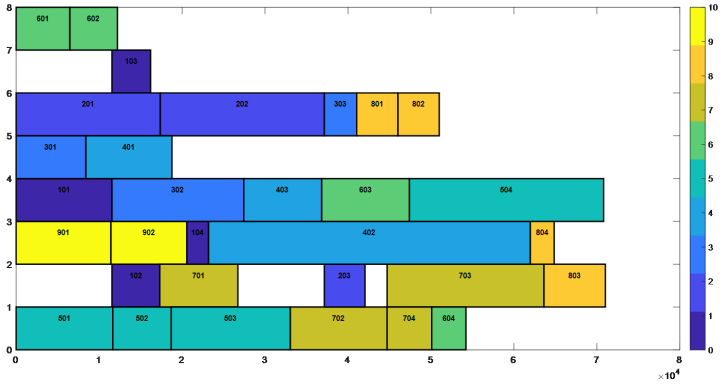

图 10: 数值仿真生产调度甘特图

其中 α_k 表示第 k 台机床的利用率，T 表示生产周期。

又因为所有产品的总加工时间是一个定值

$$z = \sum_{k=1}^{w}\sum_{i=1}^{n}\sum_{j=1}^{m}(F_{ijk} - S_{ijk}) = \sum_{i=1}^{n}\sum_{j=1}^{m}(t_{ij}^{0} + t_{ij}^{1}(N_i - 1)), \tag{16}$$

因此，设备利用率最终表现形式是

$$\bar{\alpha} = \frac{z}{w \times T}. \tag{17}$$

从式子中可以看出设备利用率与生产周期成反比关系。针对此情况，本文采取将各个产品分批调度生产的方式来寻找设备利用率与生产周期的平衡方案，各个产品分批生产可以在一定程度上避免加工时间较长的工序一旦开始加工就不能中断从而造成后续工序无法加工、机床出现空缺的情况，让各个机床设备使用时间更加紧密。

将第 i 个产品分为 λ_i 个批次来进行生产，则第 i 件产品一个批次要生产的产品数量为

$$n' = \sum_{i=1}^{n}\lambda_i. \tag{18}$$

在进行求解时，为方便计算可以将分成的产品看作新的产品，从而产品数量变为

$$n' = \sum_{i=1}^{n}\lambda_i. \tag{19}$$

当单批次加工转为多批次加工后，所有产品的总加工时间变为

$$z' = \sum_{i=1}^{n'}\sum_{j=1}^{m} t_{ij}^{0} + \frac{t_{ij}^{1}}{\lambda_i}(\sigma_i - 1). \tag{20}$$

由此可知，分批前后所有产品的总加工时间变化为

$$\beta = z' - z = \sum_{i=1}^{n}\sum_{j=1}^{m}(n' - n)(t_{ij}^{0} - t_{ij}^{1}). \tag{21}$$

由附件信息可知 $t_{ij}^{0} > t_{ij}^{1}$, 分批后总的加工时间变长，一定程度会造成生产周期变长。将分批前后的设备利用率相比得到

$$\frac{\bar{\alpha}'}{\bar{\alpha}} = \frac{z'}{z} \times \frac{T}{T'} \tag{22}$$

为保证设备的利用率提高，且保证利用率有效 ($\bar{\alpha}' \leq 1$)，我们可以得到分批处理后的总时间的范围

$$\frac{z+\beta}{w} \leq T' \leq T(\frac{z+\beta}{z}). \tag{23}$$

综上可以分析，β 随着批次的增加而不断增加，同时会影响 T' 的所在区间，故可以判断当分批次过多时，会导致总生产周期变长，在保证利用率提升的条件下可能不能满足对某些产品在某一时间段内的需求量；分批次过少时，设备利用率较小，不能实现机床的灵活分配。为此，需要选择一个合适的分批方案，在保证提高利用率的同时，也要尽量控制生产周期。

根据题目，产品数目和机床数目有可能增多，意味着 n 和 m 值将会发生变化，需要重新安排生产调度批次，每一个 n 和 m 值都会有合适的 λ_{i} 值与之对应使设备利用率和生产周期都达到最优。

6.3 结果分析

为了简化模型，我们将每个产品都平均分配成 2 批次，考虑到同一个产品的每个批次不再需要同时在一个机床进行，可以将其看作 2 个具有相同工序关系、具有相同可使用机床信息的不同产品。因此，我们将 9 个产品进行分配并生成了 18 个新的产品进行测试，得到最短生产周期为 66241 分钟，机床利用率约为 0.6476，相较于单批次调度利用率得到一定提升，测试结果如下图 11所示。

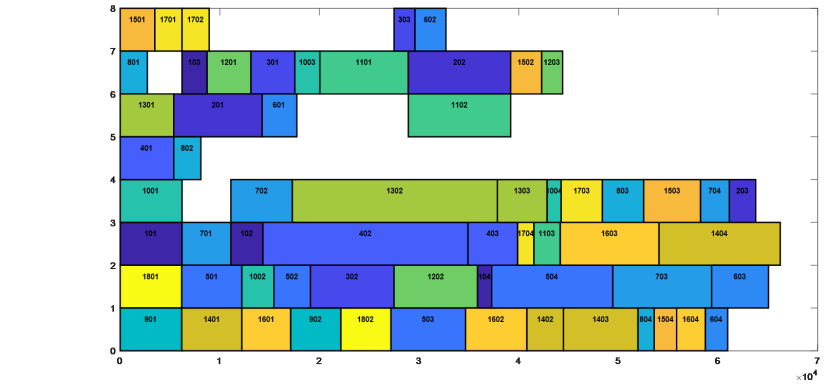

图 11: 产品分批调度甘特图

将分批次调度的方法与问题 2 中提前完成加工的要求相结合，可以发现当分批次进行加工时，可以使得原本单批次情况下不能提前完成加工的产品尽可能的同时调用多个机床分批加工，从而加快加工效率。问题二中产品 4 无法提前完成加工的问题，在通过分批调度后可以得到解决，产品 4 在 26608 分钟完成了加工，如图 12所示，其中 4 号和 13 号分别代表着产品 4 的两个不同批次。

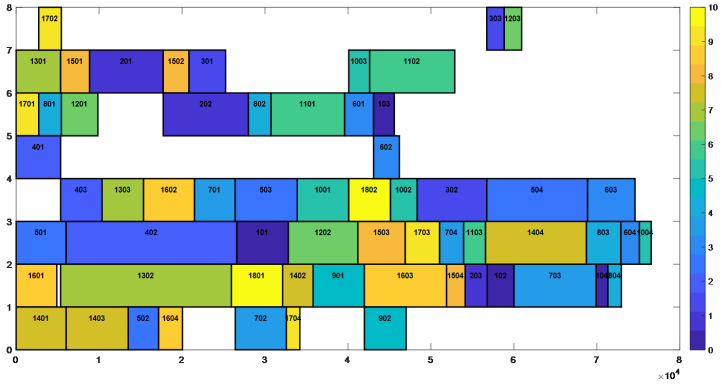

图 12: 多批次产品 4 实现产生的结果图

7 模型评价

7.1 模型总结

本文通过建立动态规划模型利用改进的遗传算法解决部分柔性生产调度作业问题；建立多目标规划模型，在保证产品提前交货的条件下也要保证生产周期最短；建立多批次柔性车间调度模型，寻找设备利用率与生产周期平衡的最优生产调度安排。

7.2 模型优点

（1）采用动态规划模型和改进的遗传算法求解，能够有效地解决部分柔性生产调度作业问题，提高了求解效率和准确性；

（2）建立了多目标规划模型，能够在保证产品提前交货的条件下，同时保证生产周期最短，实现了多目标优化；

（3）建立了多批次柔性车间调度模型，能够灵活地分配产品，提高设备利用率，同时控制生产周期，实现了生产调度的灵活性和高效性；

（4）在模型建立过程中考虑了产品数目和机床数目可能增多的情况，能够应对实际生产中的变化，具有一定的适用性和可扩展性。

7.3 模型缺点

（1）问题一的求解中，为减少计算量，只考虑单批次柔性调度的情况，且模型忽略机器故障、工序中断、机床负荷等情况，在实际应用中可能会出现偏差；

（2）由于调度规模较大，遗传算法具有随机性，并不能在所有样例中快速的选取出最优解。

7.4 模型改进与推广

模型改进：

（1）遗传算法可以与禁忌搜索相结合，既能保持种群多样性的条也能加快收敛速度 [9]；

（2）在建立模型时加入对实际不确定性因素的考虑，使模型更贴合实际。

模型推广：

本文采用遗传算法求解，可以动态调整模型，从而可以对实际情况中发生的各种事件及时作出调整，给出最优方案。本文主要是针对 P-FJSP 建立的模型，也可以解决其他领域的相似问题，如服装生产线调度安排、办公室排班问题。

参考文献

[1] An Improved Genetic Algorithm for Solving the Multi-AGV Flexible Job Shop Scheduling Problem by Leilei Meng, Weiyao Cheng, Biao Zhang, Wenqiang Zou, Weikang Fang, Peng Duan. Published on 2023-04-01.

[2] Li B, Xia X. A Self-Adjusting Search Domain Method-Based Genetic Algorithm for Solving Flexible Job Shop Scheduling Problem[J]. Computational Intelligence and Neuroscience, 2022, 2022.

[3] Li, Xuewen Xia. Published on 2022-10-10. Liu X L, Xu H Y, Chen J M, et al. Neighborhood Combination Search for Single-Machine Scheduling with Sequence-Dependent Setup Time[J]. Journal of Computer Science and Technology, 2022.

[4] Solving Job-Shop Scheduling Problem Using Genetic Algorithm Approach by Wathiq N. Abdullah. Published on 2022-12-22.

[5] Super Heuristic Genetic Algorithm for Fuzzy Flexible Job Shop Scheduling by Yu-yan Han. Published on 2022-10-10.

[6] 黄宇, 顾智勇, 张中印, 等. 基于差分量子粒子群优化算法的作业车间调度 [J]. Science Technology Engineering, 2022, 22(29).

[7] 孙娴静, 唐秋华, 邓明星. 基于改进遗传算法的异步并行拆卸序列规划 [J]. 工业工程, 2022, 25(4): 151.

[8] 孙宇婷. 求解 JSP 问题的邻域结构设计 [D]. 大连理工大学, 2021.

[9] 王展青, 魏毅峰. 求解 JSP 问题的改进遗传算法 [J]. 武汉理工大学学报: 信息与管理工程版, 2006, 28(2): 40-43.

[10] 赵诗奎, 黄林, 吕杰. Job shop 强化多工序联动邻域结构与近似评价研究 [J]. 机械工程学报, 2023, 59(4): 318-331.

附录 A: 问题一 Matlab 源代码

```
%%%%main.m
clc;
clear;

load data.mat Jc T N J x status;

%% GA
NIND = 1000;    % Number of individuals
MAXGEN = 500;   % Maximum generations
GGAP = 0.9;     % Generation gap
XOVR = 0.8;     % Crossover rate
MUTR = 0.6;     % Mutation rate
gen = 0;        % Generation counter

[N1, N2] = size(Jc);
trace = zeros(2, MAXGEN); % Optimization results
N_total = N1 * N2;        % Total number of operations

%% Initialization
Number = N2 * ones(1, N1); % Number of operations for each product

Chrom = zeros(NIND, 2 * N_total);
for j = 1:NIND
    WNTemp = Number;
    for i = 1:N_total
        % Randomly generate operations
        val = randi(N1);
        while WNTemp(val) == 0
            val = randi(N1);
        end

        % First layer represents operations
        Chrom(j, i) = val;
        WNTemp(val) = WNTemp(val) - 1;

        % Second layer represents machines
        Temp = Jc{val, N2 - WNTemp(val)};
```

```
        SizeTemp = length(Temp);
        % Randomly generate machines for operations
          Chrom(j,i+N_total)= unidrnd(SizeTemp);
    end
end

% Calculate objective function value
[~, ObjV, ~, ~] = Cal(Chrom, N, T, Jc,J,status);

%% Evolution loop
while gen < MAXGEN
    % Assign fitness values
    FitnV = ranking(ObjV);
    % Selection operation
    SelCh = select('rws', Chrom, FitnV, GGAP);
    % Crossover operation
    SelCh = across(SelCh, XOVR);
    % Mutation operation
    SelCh = aberranceJc(SelCh, MUTR, Jc, T);

    % Calculate objective fitness values
    [PVal, ObjVSel, P, S] = Cal(SelCh, N, T, Jc,J,status);
    % Reinsert the new population
    [Chrom, ObjV] = reins(Chrom, SelCh, 1, 1, ObjV, ObjVSel);
    % Increment generation counter
    gen = gen + 1;

    % Save the best value
    trace(1, gen) = min(ObjV);
    trace(2, gen) = mean(ObjV);

    % Record the best solution
    if gen == 1 || MinVal > trace(1, gen)
        Val1 = PVal;
        Val2 = P;
        MinVal = trace(1, gen);
        STemp = S;
    end
end
```

```
% Current best value
PVal = Val1;  % Operation time
P = Val2;     % Operations
S = STemp;    % Scheduling genes with machine genes

%% Plotting the solution changes
figure(1);
plot(trace(1, :),'LineWidth',2);
hold on;
plot(trace(2, :), '-.','LineWidth',2);
grid;
legend('Solution', 'Mean Population');
xlabel('Generation');
ylabel('Objective Value');

%% Displaying the best solution
figure(2);
grid on;
MP = S(1, N1 * N2 + 1:N1 * N2 * 2);

colors = parula(9);
for i = 1:N_total
    val = P(1, i);
    a = mod(val, 100);   % Operation
    b = (val - a) / 100; % Product
    Temp = Jc{b, a};

    if x(b) < a
        continue;
    else
        mText = Temp(MP(1, i));
        x1 = PVal(1, i);
        x2 = PVal(2, i);
        y1 = mText - 1;
        y2 = mText;

        Plot(x1, x2, mText);
        Plot(PVal(1, i), PVal(2, i), mText);
```

```
        hold on;

        color = colors(b, :);

        fill([x1, x2, x2, x1], [y1, y1, y2, y2], color, 'EdgeColor', 'black',
            'LineWidth', 1.5);
        text((x1 + x2) / 2, mText - 0.25, num2str(P(i)), 'HorizontalAlignment'
            , 'center', 'VerticalAlignment', 'middle', 'FontWeight', 'bold', '
            FontSize', 8);
    end
end

colormap(colors);

%%%%Cal.m
function [PVal, ObjV, P, S] = Cal(Chrom, N, T, Jc,J,status)
% Calculate the scheduling operation time for each individual in the
    population

% Initialize
NIND = size(Chrom, 1);
ObjV = zeros(NIND, 1);

% Number of products and operations
[N1, ~] = size(Jc);

for i = 1:NIND
    % Take an individual
    S = Chrom(i, :);

    % Calculate scheduling operations based on genes
    P = Calp(S, N1);

    % Calculate scheduling operation time based on scheduling operations
    PVal = Caltime(S, P, N, T, Jc,J,status);

    % Calculate completion time
    MT = max(PVal);
    TVal = max(MT);
```

```
    % Save time
    ObjV(i, 1) = TVal;

    % Initialize
    if i == 1
        Val1 = PVal;
        Val2 = P;
        MinVal = ObjV(i, 1);
        STemp = S;
    end

    % Record the smallest scheduling operation time, best scheduling
        operation time, and best scheduling operations
    if MinVal > ObjV(i, 1)
        Val1 = PVal;
        Val2 = P;
        MinVal = ObjV(i, 1);
        STemp = S;
    end
end

% Set best scheduling operation time and best scheduling operations
PVal = Val1;
P = Val2;
S = STemp;
end

function P = Calp(S, N1)
% Calculate the scheduling operations based on the gene S

N_total = length(S) / 2; % Total number of operations

% Take the operation genes (half of the genes)
S = S(1:N_total);

% Initialize
temp = zeros(1, N1);
P = zeros(1, N_total);
```

```
% Decode and generate the scheduling operations
for i = 1:N_total
    % Increment the operation count for the corresponding product
    temp(S(i)) = temp(S(i)) + 1;

    % Construct the scheduling operation
    P(i) = S(i) * 100 + temp(S(i));
end
end

function PVal = Caltime(S, P, N, T, Jc,J,status)
% Calculate the scheduling operation time based on the scheduling operations

Sta = status;
[N1, N2] = size(Jc);               % Number of products and operations

M = S(1, N1 * N2 + 1 : N1 * N2 * 2); % Machine genes (half of the genes)
N_total = length(P);                % Total number of operations

TM = zeros(1, N);
TP = zeros(1, N1);
T_total=zeros(N1,N2);
PVal = zeros(2, N_total);

% Calculate the scheduling operation time
for i = 1:N_total
    val = P(1, i);
    a = mod(val, 100);   % Operation
    b = (val - a) / 100; % Product
    Temp = Jc{b, a};

    %The operation is unused
    if  Sta(b,a)==0
        m = Temp(M(1, i));

        % Get processing time
        t = T{b, a}(M(1, i));
```

```
    % Get the start time of the machine for this operation and the
       completion time of the previous operation
    TMval = TM(1, m);
    TPval = TP(1, b);

    %find the right time for this operation
    val = max(TMval, TPval);

    % Calculate time
    PVal(:, i) = [val; val + t];
    T_total(b,a)= val+t;
    % Record the machine time and operation time for this operation
    TM(1, m) = PVal(2, i);
    TP(1, b) = PVal(2, i);
    Sta(b,a)=1;

%The first of Link operation
if J(b,a)==2
   %Calculate the operation time for next Link operation
   a = a+1;
   val = b*100+a;
   [~,index]=find(P(1,:)==val);
    Temp = Jc{b, a};
    m = Temp(M(1, index));
    % Get processing time
    t = T{b, a}(M(1, index));

    %Choose the best time
    TMval = TM(1, m);
    TPval=max(T_total(b));
    val=max(TMval,TPval);

    % Calculate time
    PVal(:, index) = [val; val + t];
    T_total(b,a) = val + t;
    % Record the machine time and operation time for this operation
    TM(1, m) = PVal(2, index);
    TP(1, b) = max(PVal(2, index),max(T_total(b,:)));
    Sta(b,a)=1;
```

```
    end
    end
end
end


%%%%aberranceJc.m
function Chrom_New = aberranceJc(Chrom, MUTR, Jc, T)
[NIND, N_total] = size(Chrom);
N_total = N_total / 2;

Chrom_New = Chrom;
N1 = size(Jc, 1);
Number = ones(1, N1);

for i = 1:NIND
    S = Chrom(i, :);  % Take an individual

    WN1Temp = Number;

    for j = 1:N_total
        JcTemp = Jc{S(j), WN1Temp(S(j))};
        SizeTemp = length(JcTemp);

        % Check if mutation should occur
        if MUTR > rand
            % Select machine (random selection)
            % S(j + N_total) = unidrnd(SizeTemp);

            % Select machine (higher chance for machines with shorter
               processing time)
            if SizeTemp == 1
                S(j + N_total) = 1;
            else
                S(j + N_total) = selectJc(S(j + N_total), T{S(j), WN1Temp(S(j))
                   });
            end
        end
```

```
        WN1Temp(S(j)) = WN1Temp(S(j)) + 1;
    end

    % Put the data into the new population
    Chrom_New(i, :) = S;
end
end

function SelS=selectJc(P,S_T)

Num=length(S_T);

MaxVal=0;
for i=1:Num
  if S_T(i)>MaxVal
   MaxVal=S_T(i);
  end
end
MaxVal=2*MaxVal;

for i=1:Num
 S_T(i)=MaxVal-S_T(i);
end

eVal=0;
for i=1:Num
 eVal=eVal+S_T(i);
end

for i=1:Num
 P(i)=S_T(i)/eVal;
end

for i=2:Num
  P(i)=P(i)+P(i-1);
end

 num=rand;
 SelS=1;
```

```
 while(num>P(SelS))
     SelS=SelS+1;
 end
end

%%%%across.m
function NewChrom = across(Chrom, XOVR)
[NIND, N_total] = size(Chrom);
N_total = N_total / 2;
NewChrom = Chrom; % Initialize new population


% Randomly select crossover individuals (shuffle crossover)
SelNum = randperm(NIND);

Num = floor(NIND / 2); % Number of paired crossover individuals
for i = 1:2:Num
    if XOVR > rand
        Pos = unidrnd(N_total); % Crossover position
        while Pos == 1
            Pos = unidrnd(N_total);
        end
        % Take the two individuals for crossover
        S1 = Chrom(SelNum(i), 1:N_total);
        S2 = Chrom(SelNum(i+1), 1:N_total);
        S11 = S2;
        S22 = S1;  % Initialize new individuals
        % Copy the middle segment to the new individuals
        S11(1:Pos) = S1(1:Pos);
        S22(1:Pos) = S2(1:Pos);
        % Compare the excess and missing genes in S11 relative to S1, and S22
            relative to S2
        S3 = S11;
        S4 = S1;
        S5 = S22;
        S6 = S2;
        for j = 1:N_total
            Pos1 = find(S4 == S3(j), 1);
            Pos2 = find(S6 == S5(j), 1);
```

```
    if Pos1 > 0
        S3(j) = 0;
        S4(Pos1) = 0;
    end
    if Pos2 > 0
        S5(j) = 0;
        S6(Pos2) = 0;
    end
end
for j = 1:N_total
    if S3(j) ~= 0 % Excess genes
        Pos1 = find(S11 == S3(j), 1);
        Pos2 = find(S4, 1); % Find missing genes
        S11(Pos1) = S4(Pos2); % Repair excess genes with missing genes
        S4(Pos2) = 0;
    end
    if S5(j) ~= 0
        Pos1 = find(S22 == S5(j), 1);
        Pos2 = find(S6, 1);
        S22(Pos1) = S6(Pos2);
        S6(Pos2) = 0;
    end
end

% Save the machine genes before crossover
S1 = Chrom(SelNum(i), :);
S2 = Chrom(SelNum(i+1), :);

for k = 1:N_total
    Pos1 = Find(S11(k), S1);
    S11(N_total+k) = S1(N_total+Pos1);
    S1(Pos1) = 0;

    Pos1 = Find(S22(k), S2);
    S22(N_total+k) = S2(N_total+Pos1);
    S2(Pos1) = 0;
end

% Generate the new population
```

```
        NewChrom(SelNum(i), :) = S11;
        NewChrom(SelNum(i+1), :) = S22;
    end
end
end

function Pos=Find(FindVal,S)

[~,n]=size(S);

Pos=-1;
for i=1:n
    if FindVal==S(i)
      Pos=i;
      break;
    end
end
end



%%%%Plot.m
function Plot(mPoint1,mPoint2,mText)

vPoint =zeros(4,2);
vPoint(1,:)=[mPoint1,mText-0.5];
vPoint(2,:)=[mPoint2,mText-0.5];
vPoint(3,:)=[mPoint1,mText];
vPoint(4,:)=[mPoint2,mText];

plot([vPoint(1,1),vPoint(2,1)],[vPoint(1,2),vPoint(2,2)])
hold on;
plot([vPoint(1,1),vPoint(3,1)],[vPoint(1,2),vPoint(3,2)])
plot([vPoint(2,1),vPoint(4,1)],[vPoint(2,2),vPoint(4,2)])
plot([vPoint(3,1),vPoint(4,1)],[vPoint(3,2),vPoint(4,2)])

end
```

附录 B: 问题 2 Matlab 部分源代码

```
%%%%Cal.m
function [PVal, ObjV, P, S] = Cal(Chrom, N, T, T_limit, Jc, J, status)
% Calculate the scheduling operation time for each individual in the
    population

% Initialize
NIND = size(Chrom, 1);
ObjV = zeros(NIND, 1);

% Number of products and operations
[N1, ~] = size(Jc);

for i = 1:NIND
    % Take an individual
    S = Chrom(i, :);

    % Calculate scheduling operations based on genes
    P = Calp(S, N1);

    % Calculate scheduling operation time based on scheduling operations
    PVal = Caltime(S, P, N, T, T_limit, Jc, J, status);

    % Calculate completion time
    MT = max(PVal);
    TVal = max(MT);

    % Save time
    ObjV(i, 1) = TVal;

    % Initialize
    if i == 1
        Val1 = PVal;
        Val2 = P;
        MinVal = ObjV(i, 1);
        STemp = S;
    end
```

```
    % Record the smallest scheduling operation time, best scheduling
        operation time, and best scheduling operations
    if MinVal > ObjV(i, 1)
        Val1 = PVal;
        Val2 = P;
        MinVal = ObjV(i, 1);
        STemp = S;
    end
end

% Set best scheduling operation time and best scheduling operations
PVal = Val1;
P = Val2;
S = STemp;
end

function P = Calp(S, N1)
% Calculate the scheduling operations based on the gene S

N_total = length(S) / 2; % Total number of operations

% Take the operation genes (half of the genes)
S = S(1:N_total);

% Initialize
temp = zeros(1, N1);
P = zeros(1, N_total);

% Decode and generate the scheduling operations
for i = 1:N_total
    % Increment the operation count for the corresponding product
    temp(S(i)) = temp(S(i)) + 1;

    % Construct the scheduling operation
    P(i) = S(i) * 100 + temp(S(i));
end
end
```

```
function PVal = Caltime(S, P, N, T, T_limit, Jc, J, status)
% Calculate the scheduling operation time based on the scheduling operations

Sta = status;
[N1, N2] = size(Jc);                % Number of products and operations

M = S(1, N1 * N2 + 1 : N1 * N2 * 2); % Machine genes (half of the genes)
N_total = length(P);                 % Total number of operations

TM = zeros(1, N);
TP = zeros(1, N1);
T_total=zeros(N1,N2);
PVal = zeros(2, N_total);

% Calculate the scheduling operation time
for i = 1:N_total
    val = P(1, i);
    a = mod(val, 100);    % Operation
    b = (val - a) / 100; % Product
    Temp = Jc{b, a};

    %The operation is unused
    if  Sta(b,a)==0
        m = Temp(M(1, i));

        % Get processing time
        t = T{b, a}(M(1, i));

        % Get the start time of the machine for this operation and the
            completion time of the previous operation
        TMval = TM(1, m);
        TPval = TP(1, b);

        % Find the right time for this operation
        val = max(TMval, TPval);

        % Calculate time
        PVal(:, i) = [val; val + t];
        T_total(b,a)= val+t;
```

```
        % Record the machine time and operation time for this operation
        TM(1, m) = PVal(2, i);
        TP(1, b) = PVal(2, i);
        Sta(b,a)=1;

    %The first of Link operation
    if J(b,a)==2
       %Calculate the operation time for next Link operation
       a = a+1;
       val = b*100+a;
       [~,index]=find(P(1,:)==val);
        Temp = Jc{b, a};
        m = Temp(M(1, index));
        % Get processing time
        t = T{b, a}(M(1, index));

        % Choose the best time
        TMval = TM(1, m);
        TPval=max(T_total(b));
        val=max(TMval,TPval);

        % Calculate time
        PVal(:, index) = [val; val + t];
        T_total(b,a) = val + t;
        % Record the machine time and operation time for this operation
        TM(1, m) = PVal(2, index);
        TP(1, b) = max(PVal(2, index),max(T_total(b,:)));
        Sta(b,a)=1;
    end
    end
end
% Finish in Specified Date
for i=1:N1
    if max(T_total(i,:))>T_limit(i)
       PVal(i,end)=200000;
    end
end
end
```