# [应用模拟退火算法解决置换流水线车间调度问题]

name (学号: number)

**摘　要**:　置换流水线车间调度问题。是指在多条流水线车间中完成一组作业的最优调度问题。这个问题涉及到在多个工作站之间安排任务，以便最小化总体生产时间或最大化生产效率。在这个问题中，而个作业都有一个处理顺序。目标是找到一个最佳处理顺序，使得能够最小化完成所有作业所需的总时间。本文使用模拟退火算法，它是一种模拟物理世界中退火后降温的过程，用于通过温度的控制，在解空间中进行随机游走来搜索最优解。模拟退火算法的基本思路是从一个随机的解开始，在解空间中搜索其周围的解，并按一定概率接受较差的解，进而能够实现不陷于局部最小值能找到全局最小值的结果。本文使用模拟退火算法解决置换流水线车间调度问题，在可接受的时间内得出了结果，并测试了不同的实验条件下对于实验结果的影响，得出了最优化解法。

**关键词**:　置换流水线车间调度问题 最优化方法 模拟退火算法

## 1　引言

a) 置换流水车间调度问题（Permutation Flow Shop Scheduling Problem，PFSP）是指在一条具有多个工作站的流水线上，对一组作业进行调度，使得所有作业在流水线上按照指定的加工顺序进行加工，每个作业必须在前一个工作站上完成后才能移动到下一个工作站上进行加工。在该问题中，每个作业的加工时间在不同的工作站上可能不同，目标能够最小化完成所有作业所需的总时间[1]。

PFSP 是一个经典的组合优化问题，其应用广泛，如生产调度、作业排程、制造流程优化等领域。PFSP 的目标是最小化完成所有作业所需的总时间。在实际生产中，PFSP 的解决可以帮助企业优化制造流程，提高生产效率，降低制造成本。

b) 置换流水车间调度问题的数学语言描述

| | |
|---|---|
| $m$ | 机器编号 |
| $n$ | 工件编号 |
| $m_{max}$ | 机器的数量 |
| $n_{max}$ | 工件的数量 |
| $N_{n_{max}}$ | 工件的加工顺序 |
| $T^{m}_{N_n\,start}$ | 第 m 个机器加工顺序中第 n 个需要加工的工件的开始时间 |
| $T^{m}_{N_n\,end}$ | 第 m 个机器加工顺序中第 n 个需要加工的工件的结束时间 |
| $T_{m_{max}\,n_{max}}$ | 第 m 个机器加工编号为 n 的工件所需要的时间 |

$$\min \quad T^{m}_{N_n end} - T^{0}_{0\ start}$$

$$s.t. \quad T^{m}_{N_n end} - T^{m}_{N_n start} = T_{m\,N_n} \tag{1}$$

$$T^{0}_{0\ start} = 0 \tag{2}$$

$$\forall n,\ T^{m}_{N_n\ start} > T^{m}_{N_{n+1}\ end} \tag{3}$$

$$\forall m,\ T^{m}_{N_n\ start} > T^{m+1}_{N_n\ end} \tag{4}$$

$$m \in [0,\ m_{max}) \tag{5}$$

$$n \in [0,\ n_{max}) \tag{6}$$

$$\text{var} \quad N_{n_{max}}$$

对于每一个问题来说$m_{max}$，$n_{max}$为已知常数，$\mathrm{T}_{m_{max}\,n_{max}}$为已知的$m_{max}$，$n_{max}$维矩阵。$N_{n_{max}}$是决策变量，为$n_{max}$维向量。有六个约束条件，(1)式表示机器一次加工的结束时间减去起始时间严格等于已知的加

工时间，(2)式代表起始时间为 0，(3)式代表工件在同一时刻只能由一台机器加工[2]，(4)式代表一台机器在同一时刻只能加工一个工件，(5)(6)式代表机器与工件编号的取值范围为 0 到各自的数量。

c) 本文使用 Java 模拟退火算法来解决 PFSP 问题，模拟退火算法模拟了金属退火过程，从而实现找到一个接近最优解的解的结果，关键步骤包括初始化解，选择邻域解，判断是否接受新解，降低温度，终止判断等步骤。并使用 Python 实现甘特图的绘制。能够实现对于 PFSP 问题在可接受的时间内给出确定解并且快速生成可视图表。值得注意的是由于模拟退火算法的随机性，可能会出现对于相同的问题有不同的结果的情况。

本文后续部分组织如下。第 2 节详细陈述使用的方法，第 3 节报告实验结果，第 4 节对讨论本文的方法并总结全文。

## 2 算法设计

本文使用模拟退火算法（Simulated Annealing）对 PFSP 进行计算，该算法用于在解空间中搜索最优解或接近最优解。它最初是由 S. Kirkpatrick, C. D. Gelatt Jr. 和 M. P. Vecchi 在 1983 年提出的，灵感来自于金属退火过程的物理原理。

金属退火过程是指将材料加热至高温然后缓慢冷却的过程，通过这个过程可以改善材料的物理性能。在模拟退火算法中，解空间被看作是一个能量空间，通过温度的控制，在解空间中进行随机游走来搜索最优解。温度参数控制着在搜索过程中接受较差解的概率，并逐渐降低温度来减少解的波动，直到找到一个接近最优解的解为止[3]。

使用模拟退火算法解决 PFSP 的关键操作：

1. 初始化解：随机选择一个初始解，通常可以从一个已知的解开始，或者随机生成一个解。本文使用生成的一组随机序列来作为初始解。
2. 选择邻域解：在当前解的邻域内随机选择一个新解。本文将该 PFSP 问题中的邻域定义为与当前处理序列只存在一组元素交换的序列。
3. 判断是否接受新解：如果新解更优，则直接接受新解，反之根据一定的概率规则决定是否接受新解，本文概率 p 的计算方法我定义为：

$$p = e^{(newTime - lastTime) * 100 / newTime / temperature}$$

4. 降低温度：通过逐渐降低温度来控制接受劣解的概率，使得算法在搜索的初期较为随机，而在搜索的后期逐渐趋于确定性。降低温度的计算公式如下：

$$temperature = temperature * \theta$$

$$\theta = 0.9999$$

5. 终止条件：当满足一定终止条件时，算法停止搜索并返回最优解或近似最优解。在本文中，终止条件被设置为出现同样的解超过 150 次。

加工完成时间计算方法主要步骤：

1. 传入待计算的时间序列并进行初始化操作：首先我们需要确定两个重要的性质：第一个机器一定是无等待的按照加工序列进行加工。第一个工件一定是无等待的在进行所有加工。已知这两条性质之后即可确定第一台机器和第一个工件的所有加工时间。相当于确定了动态规划的起始状态。
2. 我们关注第 n + 1 个工件：对于第一台机器来说，当第 n 个工件加工完成之后就会立即开始第 n + 1 个工件的加工，此时我们记录第 n + 1 个工件在第一台机器加工完成的时间为$t_1$。对于第二台机器来说，当第 n 个工件加工完成后就会进入可加工状态，此时我们记录第 n 个工件在第二台机器加工完成的时间为$t_2$，那么第二台机器进行第 n + 1 个工件加工的时间就是$\max(t_1, t_2)$。对于之后的机器或者工件的时间计算就以此类推。其主要的思想与动态规划相似。

复杂度分析：

**空间复杂度：**本文在使用模拟退火算法解决 PFSP 问题时，主要需要存储的数据是当前解和上一个解，以及各个作业在每个机器上的开始和结束时间。因此，空间复杂度的主要来源是存储这些数据的空

间。

对于当前解和上一个解，由于每次只需要存储一组解，因此其空间复杂度为 O(n)。其中，n 为作业的数量。

对于各个作业在每个机器上的开始和结束时间，本文使用一个三维数组来进行存储，其中第一维表示机器的编号，第二维表示作业的编号，第三维表示开始时间和结束时间。因此，其空间复杂度为 O(mn)，其中 m 为机器的数量，n 为作业的数量。

综上所述，使用模拟退火算法解决置换 PFSP 问题的空间复杂度为 O(mn)。

**时间复杂度**：本文使用模拟退火算法来解决 PFSP 问题的时间复杂度取决于两个主要的操作：

1. 产生随机解：该操作的时间复杂度为 O(n)，其中 n 是车间调度问题中工作数量的总和。
2. 计算新解的完成时间：该操作的时间复杂度为 O(m * n)，其中 m 是车间数，n 是工作数。

模拟退火算法通过迭代来优化解决方案，每次迭代都会随机产生一个新的解并计算该解的完成时间，因此总的时间复杂度是 O(iterations * (m * n))，其中 iterations 是迭代次数。下面分析 iterations 的计算。

整个算法的停止有两种情况：

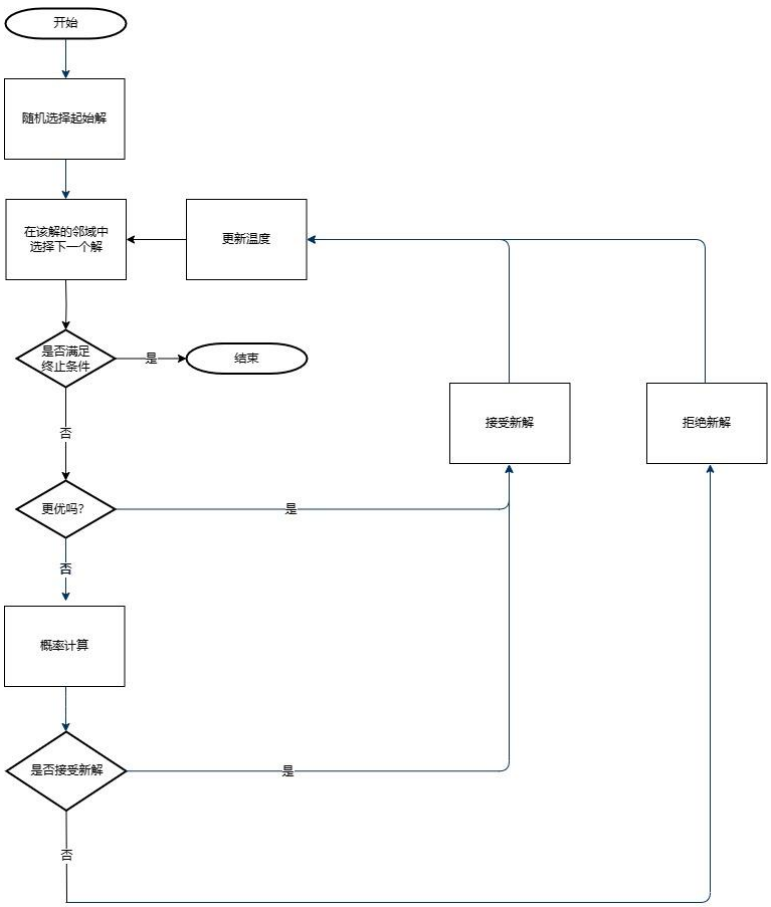
1. 温度小于所设置的最小温度阈值。
2. 未选择新解次数达到阈值。

对于第一种情况：$\text{iterations} = \log_{theta} T\_freeze / T\_start$。

对于第二种情况：实际迭代次数要由实际问题决定，但是迭代次数一定小于第一种情况的迭代次数。

综上所述：使用模拟退火算法解决置换 PFSP 问题的时间复杂度为

$$O(\log_{theta} T\_freeze / T\_start * mn)$$



# 3 实验

## 3.1 实验设置

**实验平台与环境：**

AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz
Windows 11 家庭中文版 21H2
IDEA 2023.1
openjdk 11.0.16.1 2022-08-12 LTS
PyCharm 2023.1
Python 3.9.12
numpy 1.21.5
matplotlib 3.5.1

**实验默认参数设置：**

初始温度为 1000
温度下降系数为 0.999
结束温度为 0.01
超时上限为 150 次

**执行方法：**

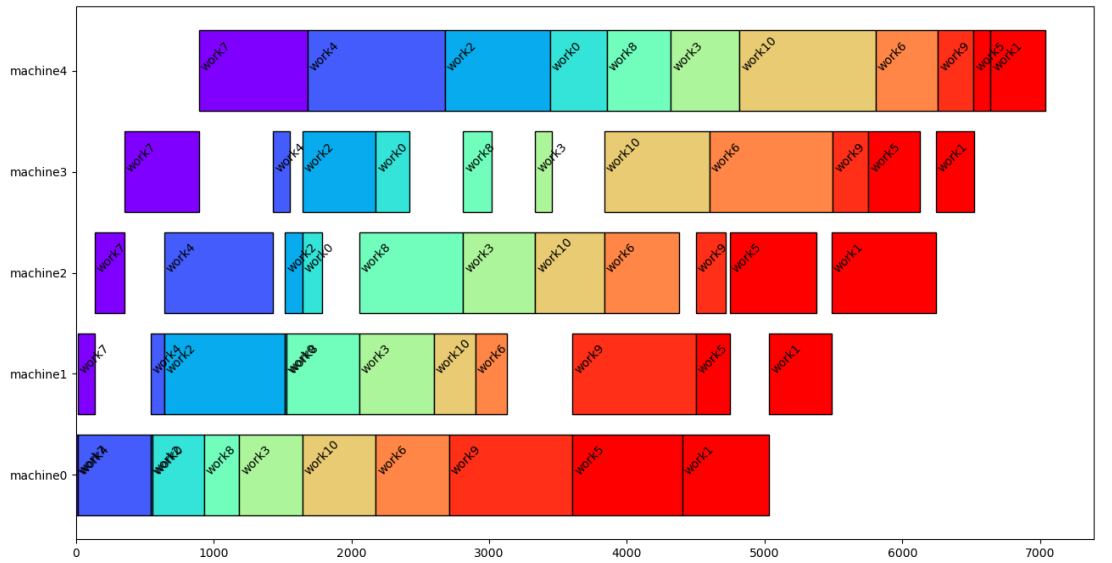
1. 将 java 执行目录中放入待测试的 test.txt 文件，文件格式为第一行写入用空格隔开的两个数字表示机器的数量 m 与工件的数量 n，之后 m 行表示每个工件在每个机器中所加工的时间，仅支持单次数据测试。
2. 算法执行完毕之后会生成 timeTable.txt 文件，文件格式为第一行表示工件执行序列，其余 m 行表示每台机器每一次加工的起始时间和终止时间。
3. 执行 python 代码，将 timeTable.txt 文件的绝对路径正确的填入，执行即可生成甘特图。

## 3.2 实验结果

### 3.2.1 实验结果展示

#### 3.2.1.1 初始参数设置的情况下测试案例执行结果

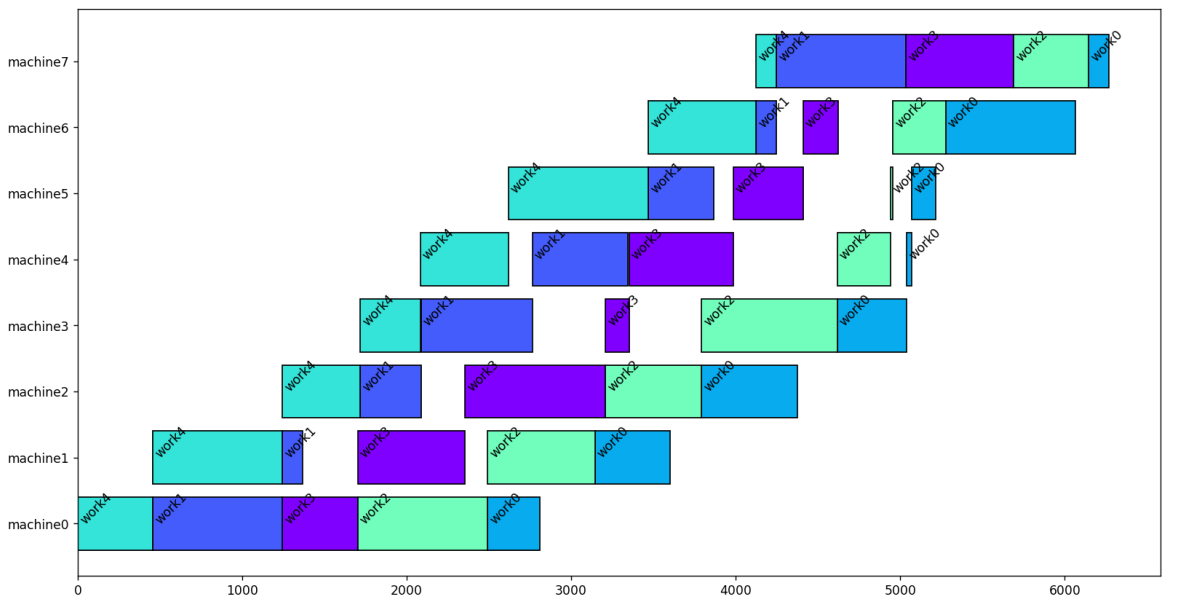
Instance 0
最短时间：7038
加工次序：7 0 2 10 8 4 9 6 5 1 3




Instance 1

最短时间：6269

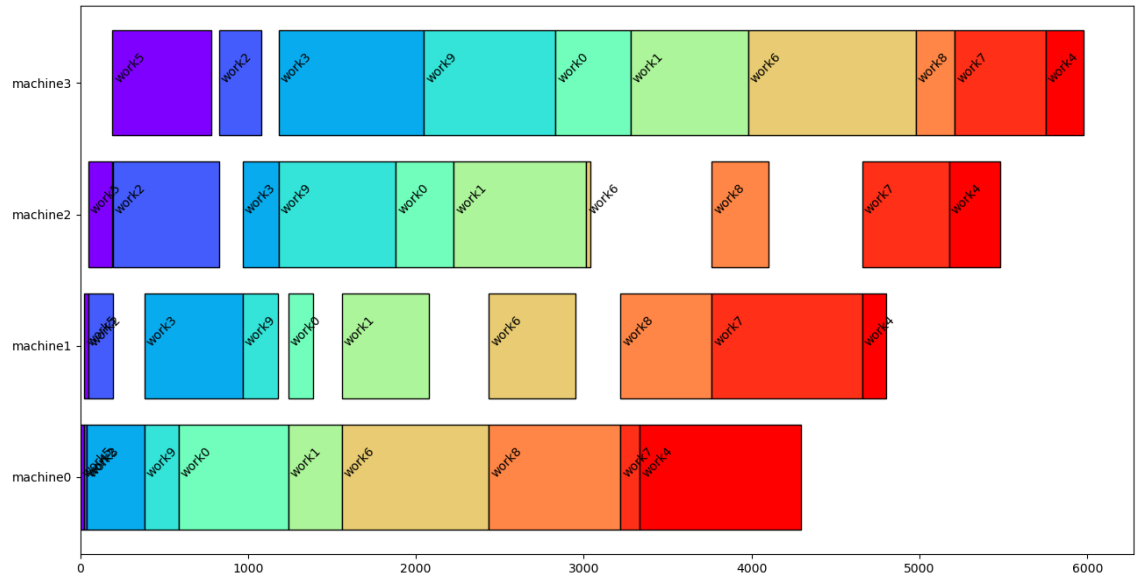
加工次序：4 1 3 2 0




Instance 2

最短时间：5977

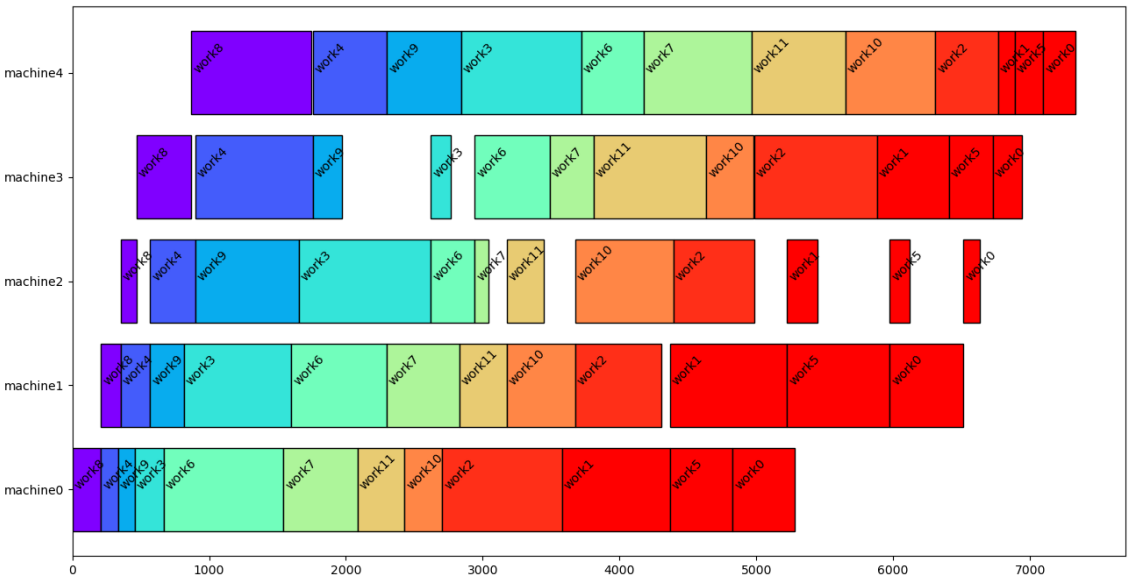
加工次序：5 2 3 9 0 1 6 8 7 4




Instance 3

最短时间：7332

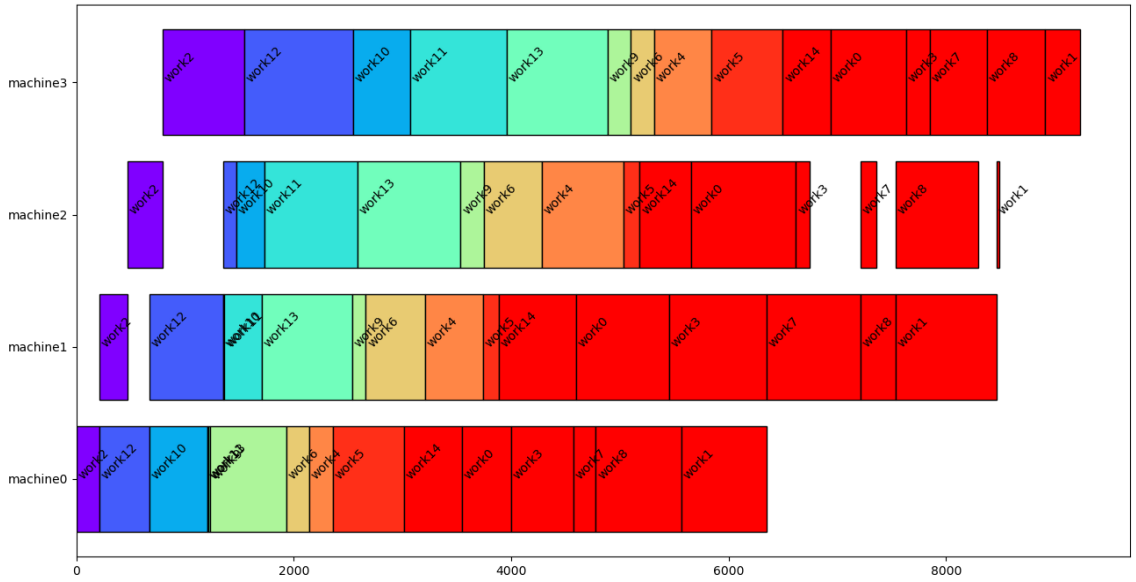
加工次序：8 4 9 3 6 7 11 10 2 1 5 0




Instance 4

最短时间：9231

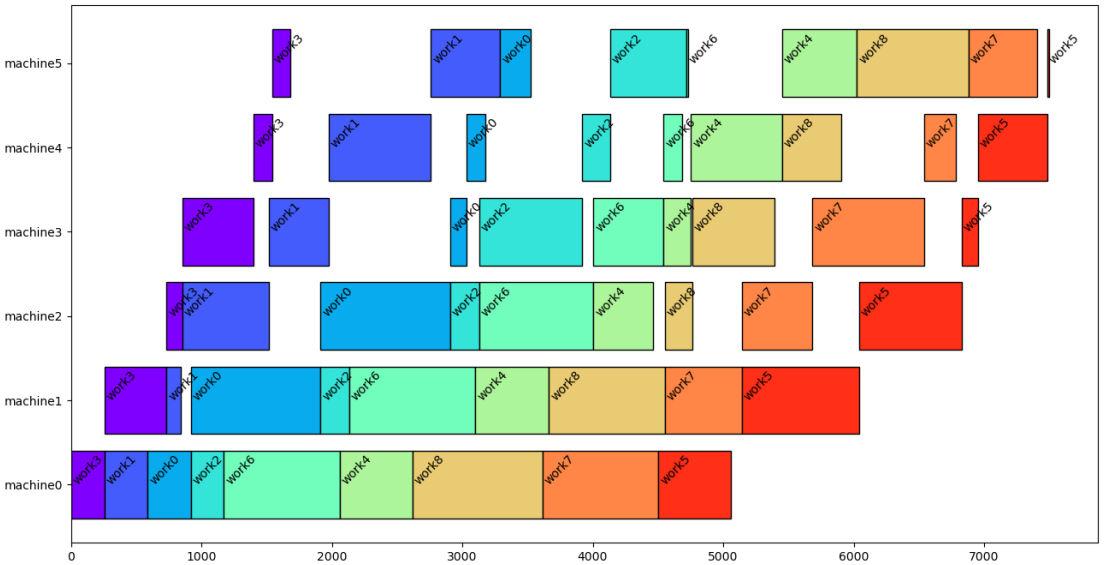
加工次序：2 12 10 11 13 9 6 4 5 14 0 3 7 8 1




Instance 5

最短时间：7528

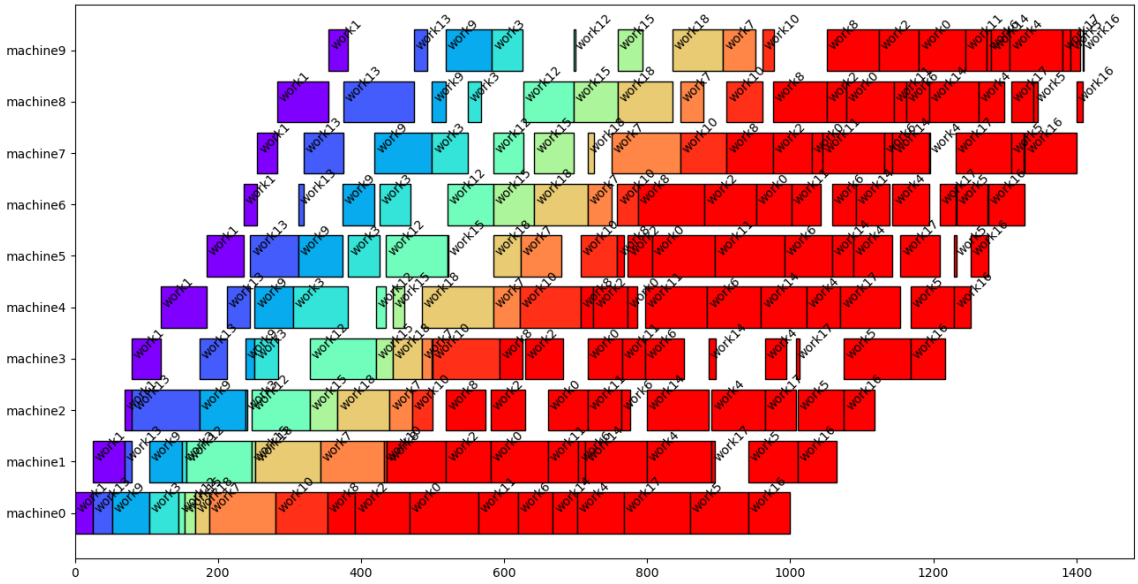
加工次序：3 1 0 2 6 4 8 7 5




Instance 6

最短时间：1410

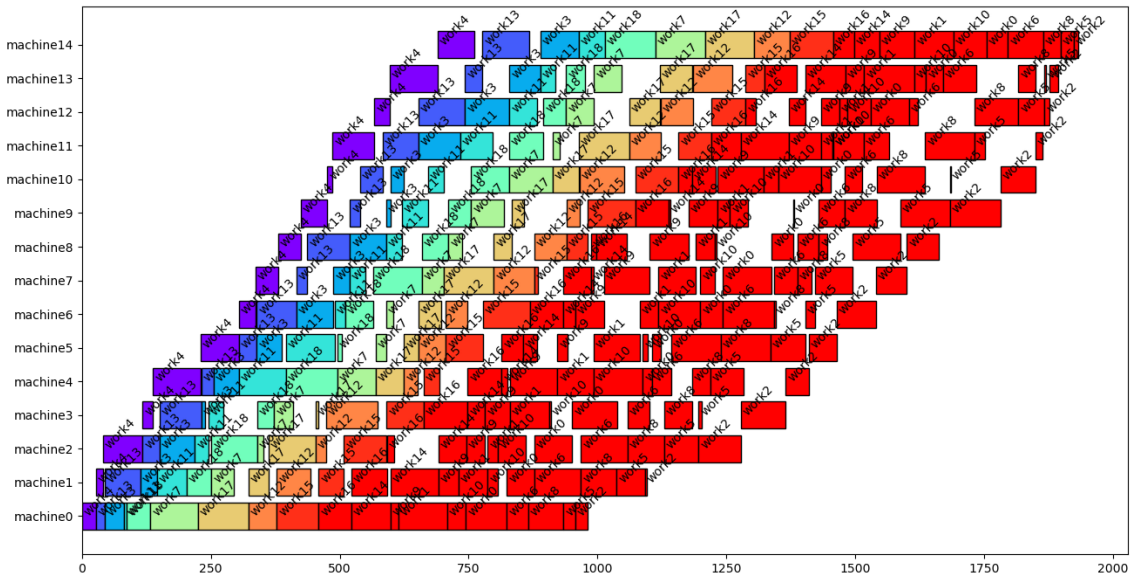
加工次序：1 13 9 3 12 15 18 7 10 8 2 0 11 6 14 4 17 5 16




Instance 7

最短时间：1932

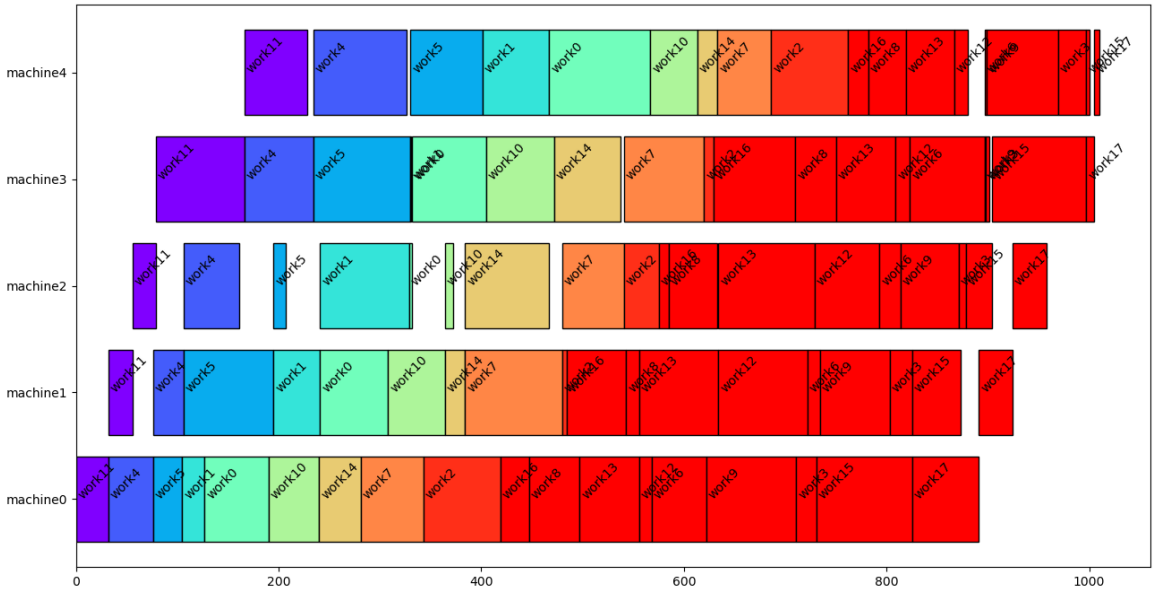
加工次序：4 13 3 11 18 7 17 12 15 16 14 9 1 10 0 6 8 5 2




Instance 8

最短时间：1010

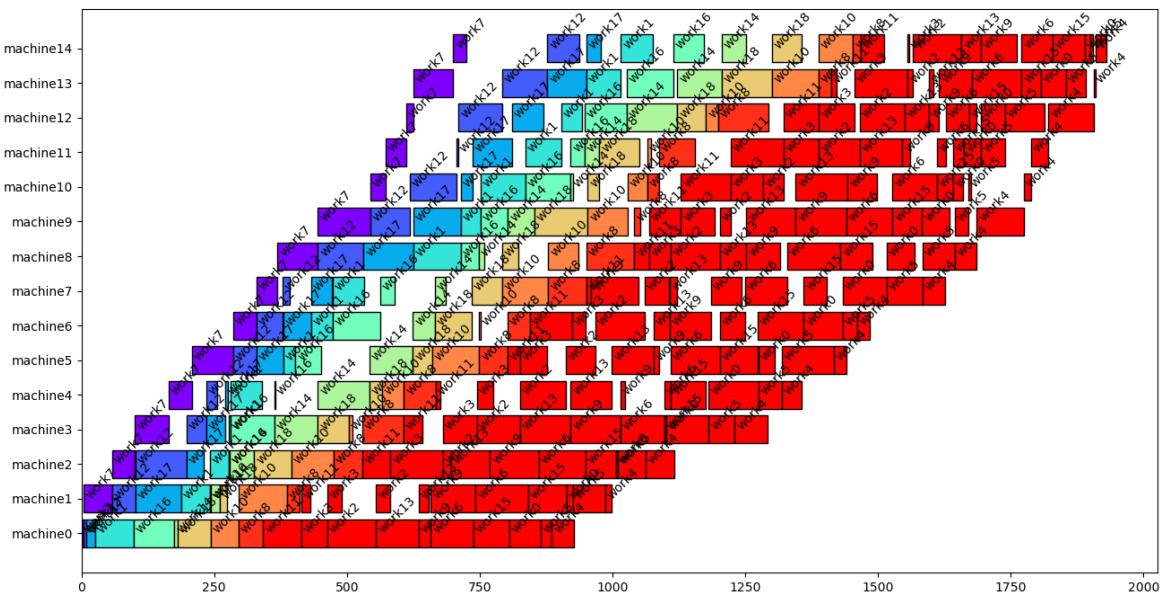
加工次序：11 4 5 1 0 10 14 7 2 16 8 13 12 6 9 3 15 17




Instance 9

最短时间：1933

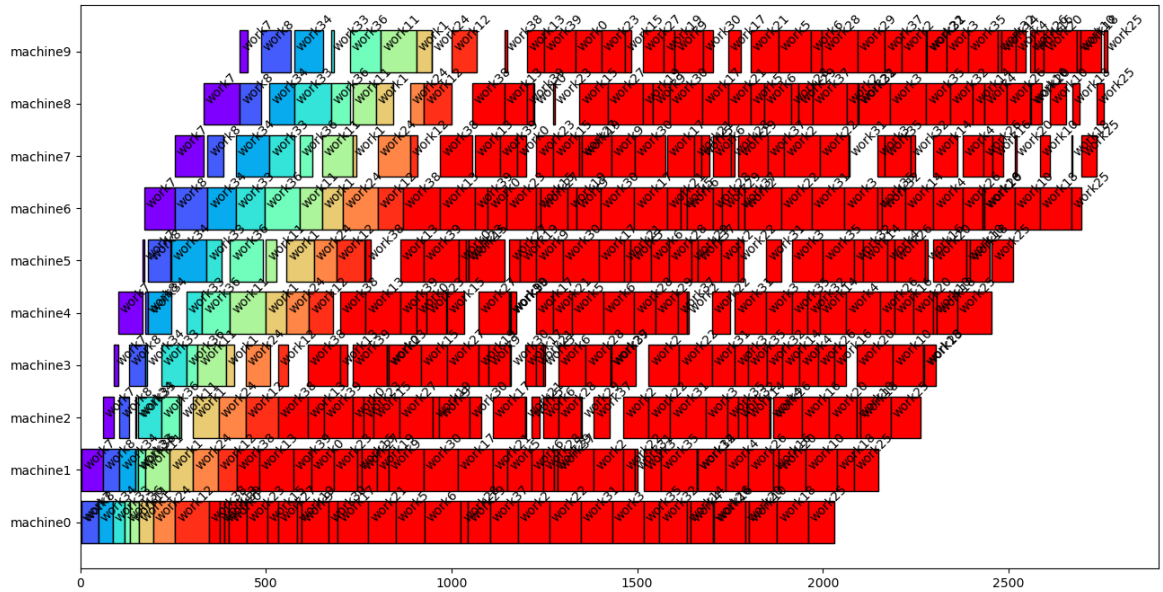
加工次序：7 12 17 1 16 14 18 10 8 11 3 2 13 9 6 15 0 5 4




Instance 10

最短时间：2766

加工次序：7 8 34 33 36 13 32 15 31 20 19 9 29 21 1 23 3 27 35 38 12 26 2 39 17 11 28 30 5 4 0 10 6 37 16 14 24 22 18 25




每个测试示例的结果均由 40 次重复模拟退火算法取最低值得出，对于以上 11 个测试示例，重复实验结果基本稳定，用时最长约为 8769ms，能够做到在可接受的时间范围内得出稳定收敛结果，实验结果合理。

**3.2.1.2　对比试验：**

该节探讨在单一变量的条件下参数的改变对于运行准确性的影响，均使用 40 次 Instance 10 重复实验来进行测试。

温度下降系数对于运行结果和运行时间的影响见表一，其余系数均设置为默认值，可见温度下降系数对于运行时间的影响非常大，这一点符合了关于时间复杂度的分析，对于运行结果优秀性有一定程度上的影响，虽然从初步的测试中有一定程度上的相关性，但是是否呈线性相关有待于进一步的验证。

**表 1**　温度下降系数与运行结果

| 温度下降系数 | 运行结果 | 40 次重复运行时间（ms） |
|---|---|---|
| 0.9999 | 2977 | 44393 |
| 0.999 | 3035 | 4331 |
| 0.99 | 3062 | 493 |
| 0.9 | 3103 | 97 |
| 0.8 | 3145 | 72 |
| 0.7 | 3172 | 65 |

冷却温度对于运行结果和运行时间的影响见表二，其余系数均设置为默认值，可见温度下降系数对于运行时间有一定程度上的影响，这一点符合了关于时间复杂度的分析，对于运行结果优秀性有一定程度上的影响，虽然从初步的测试中有一定程度上的相关性，但是是否呈线性相关有待于进一步的验证。

**表 2**　冷却温度与运行结果

| 冷却温度 | 运行结果 | 40 次重复运行时间（ms） |
|---|---|---|
| 2 | 3026 | 4428 |
| 1 | 2992 | 4863 |
| 0.5 | 2891 | 5474 |
| 0.1 | 2773 | 6725 |
| 0.05 | 2783 | 7266 |
| 0.02 | 2769 | 8159 |
| 0.01 | 2766 | 8601 |
| 0.005 | 2775 | 9467 |
| 0.001 | 2766 | 10728 |

超时次数对于运行结果和运行时间的影响见表三，其余系数均设置为默认值，实验结果显示超时次数对

于运行结果以及运行时间的影响都不大。

**表 3** 超时次数与运行结果

| 超时次数 | 运行结果 | 40 次重复运行时间（ms） |
|---|---|---|
| 000 | 2776 | 8568 |
| 10000 | 2769 | 8672 |
| 5000 | 2777 | 8597 |
| 1000 | 2792 | 8709 |
| 200 | 2776 | 8795 |
| 150 | 2782 | 8479 |
| 100 | 2775 | 8570 |
| 50 | 2770 | 8359 |

## 4 总结

本文主要探讨的是置换流水线车间调度问题（PFSP）中，如何使用模拟退火算法寻找最优解，以最小化完成所有作业所需的总时间。本文使用 Java 实现了模拟退火算法，并使用 Python 实现了甘特图的绘制，能够在可接受的时间内给出确定解并且快速生成可视化图表。同时本文对算法中参数的选择也进行了实验测试，增强了实验的可信度和可重复性。

本文只是针对一组实验数据进行了测试，没有进行大规模实验来验证算法的鲁棒性和可靠性。为了更加全面地评估算法的性能，需要在不同规模和不同数据类型的实例上进行测试，以检验算法的适用性。

为了解决这些问题，可以使用自适应模拟退火算法来优化模拟退火算法的参数[4]，同时进行更多的实验以验证算法的鲁棒性和可靠性。此外，也可以结合其他优化算法来进行求解，以提高求解的效率和精度。

**参考文献：**


[1] Singh H, Oberoi J S, Singh D. Multi-objective permutation and non-permutation flow shop scheduling problems with no-wait: a systematic literature review[J]. RAIRO-Operations Research, 2021, 55(1): 27-50.

[2] Hasija S, Rajendran C. Scheduling in flowshops to minimize total tardiness of jobs[J]. International Journal of Production Research, 2004, 42(11): 2289-2301.

[3] 刘莹, 谷文祥, 李向涛. 置换流水线车间调度问题的研究[J]. 计算机科学, 2013, 40(11): 1-7, 22.

[4] Sotskov Y N, Tautenhahn T, Werner F. Heuristics for permutation flow shop scheduling with batch setup times[J]. Operations-Research-Spektrum, 1996, 18: 67-80.


## 附录 A：考虑无等待约束的改进算法

对于置换流水车间调度问题，在实际的生产中，有些特定的工件需要满足在多个机器加工的过程之间不能有空闲时间的条件，这种约束条件称为无等待约束，对于解决无等待约束的置换流水车间调度问题，需要对算法进行一定程度上的改进。

### A.1 算法改进

在加入无等待约束之后主要发生改变的内容为代价函数的计算方法，在本文中也就是所耗费时间的计算，其余关于邻域以及模拟退火的过程本文未进行改变。接下来介绍代价函数的计算方法

由约束可知，第 n + 1 个机器对于第 n 个工件的加工起始时间一定等于第 n 个机器对于第 n 个工件的加工结束时间，所以对于同一个工件所有机器的加工时间来说，他们之间的相对时间差是恒定的。所以在每个工件的初步计算时将第一个机器的起始时间固定在上一个工件的结束时间之后再进行动态规划即可确定相对时间。为了最小化时间，我们计算每个机器加工第 n 个工件的结束时间和加工第 n + 1 个工件的起始时间之差，找到最小差值，为所有机器加工第 n + 1 个工件的时间减去最小差值即为满足相对时间下的最小加工时间。以此类推，计算所有的工件即可。整体思想类似动态规划。

### A.2 实验结果

**默认参数设置：**

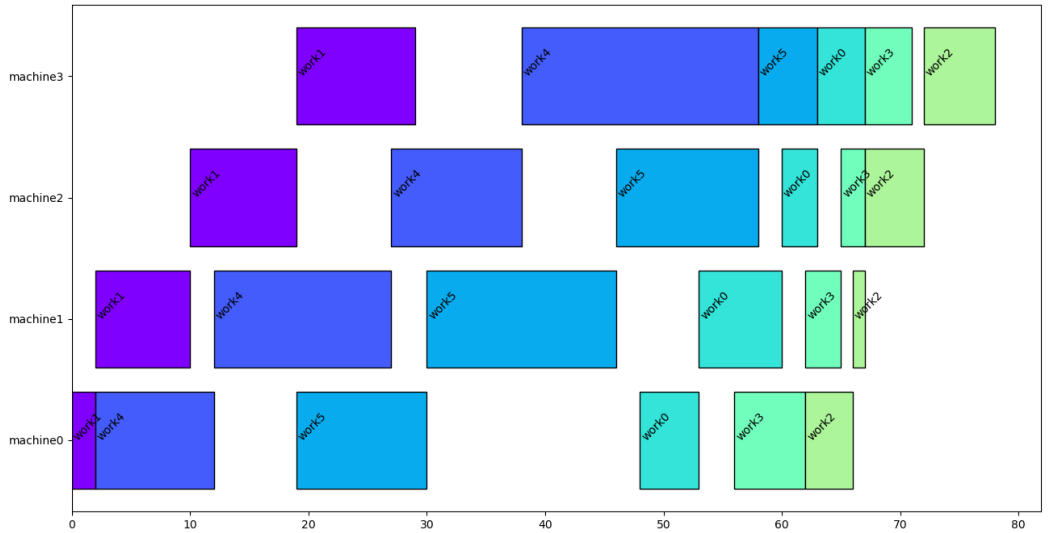
初始温度为 1000
温度下降系数为 0.999
结束温度为 0.01
超时上限为 150 次

**实验结果：**

Instance 1

**加工次序：1 4 5 0 3 2**

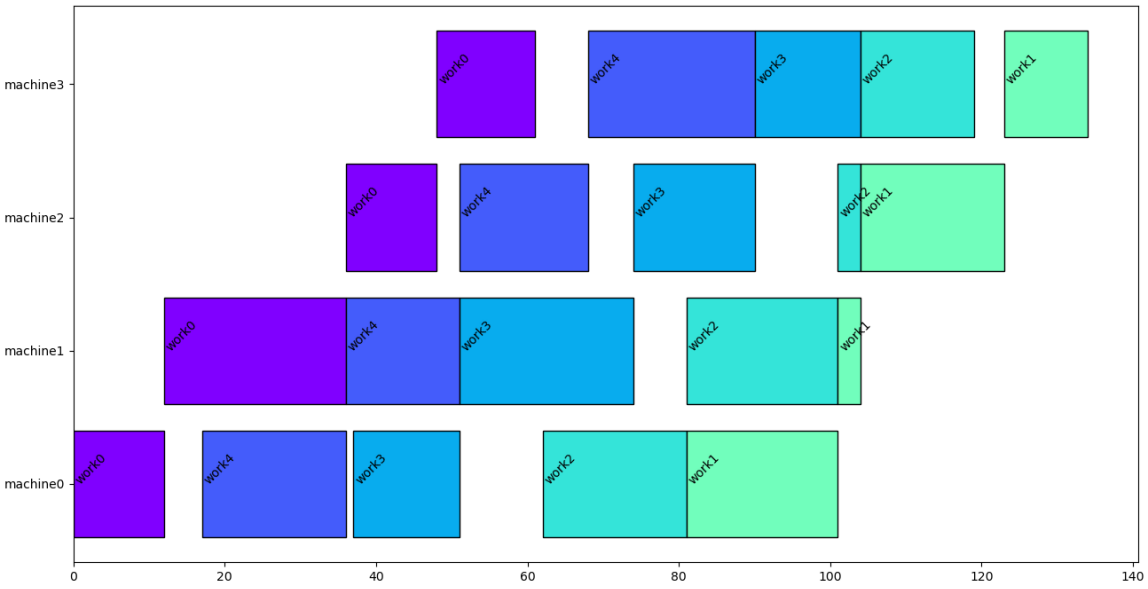
**最短时间：78**




Instance 2

**加工次序：0 4 3 2 1**

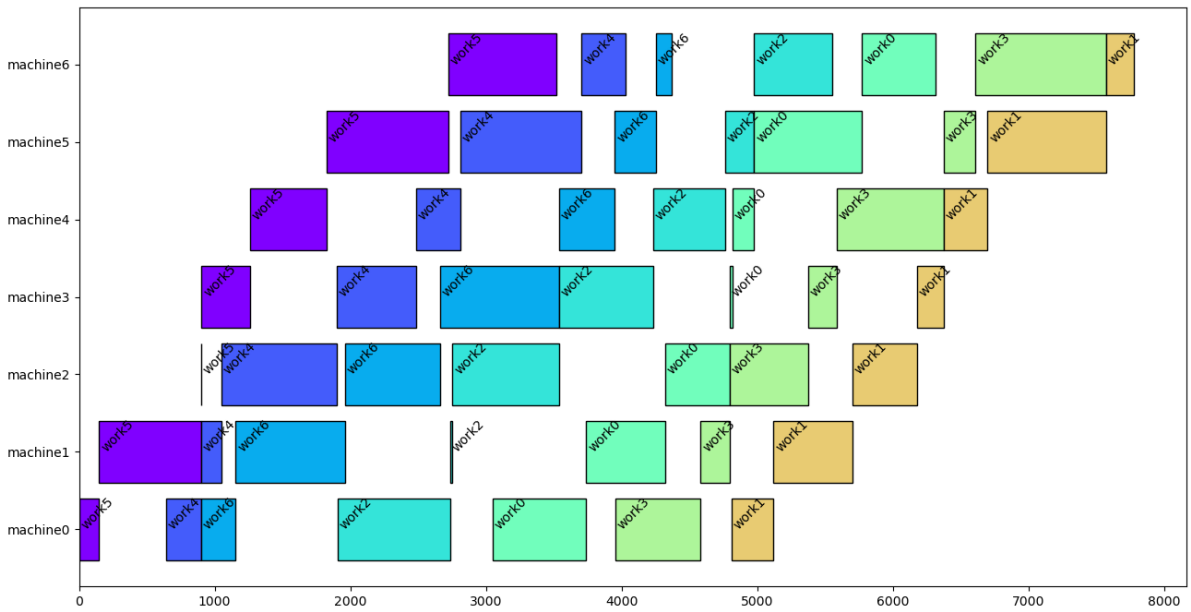
**最短时间：135**




Instance 3

加工次序：5 4 6 2 0 3 1

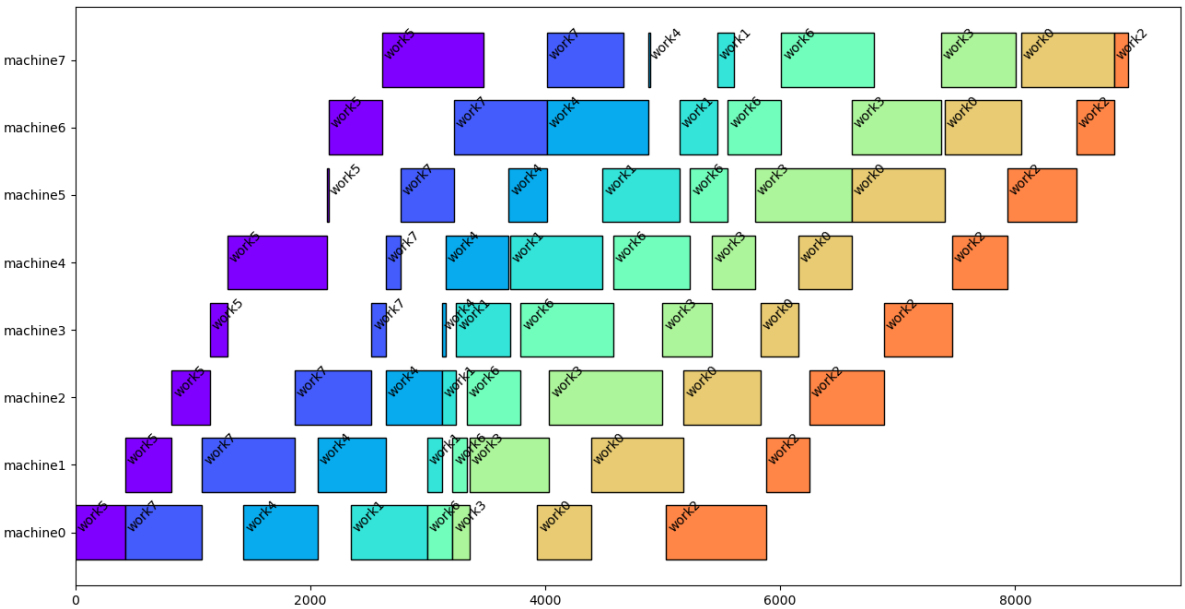
最短时间：7885




Instance 4

加工次序：5 7 4 1 6 3 0 2

最短时间：9008




能够在可接受的时间内完成无等待约束置换流水车间调度问题，能够给出收敛的结果，重复实验结果稳定，实验结果合理。