智能搜索算法求解流水车间调度问题

何宇轩

摘 要: 流水车间调度问题简述：流水车间中有若干工件需要在若干机器上进行加工，并且一台机器只能同时加工一个工件，一个工件只能同时在一台机器上加工。我们需要根据工件的加工顺序以及工件在每台机器的加工时间，求出所有工件完成加工用时最短的最优调度方案。该问题难点在于多目标多线程，算法要有较高的鲁棒性，需要对整个流水作业做出最优化策略

本文使用了两种智能搜索算法和一种算法变体分别求解该调度问题，并进行结果对比。首先是登山搜索算法，对深度优先搜索算法进行了改进，避免了遍历，但只能实现局部择优且受初始解影响较大。为了能够达到较好的搜索效果，本文采用贪心思想生成一个较优的初始解，即从0时刻开始紧密安排工件加工，使得算法能在解空间的一个较好的范围内进行搜索。

第二种算法是模拟退火算法，第三种算法是在模拟退火算法的基础上，引入了轮盘赌选择法，即轮盘赌模拟退火算法。模拟退火算法收敛速度远不及登山搜索，并且最终解和算法耗时常常受到冷却方式、冷却速率、迭代次数的影响。为了追求在较快的搜索速度下寻求最优解，采用轮盘赌策略选择邻域解，再根据Metropolis准则更新当前解，充分发挥模拟退火性能。

三种算法在实验中均表现出良好的效果。登山搜索收敛速度快，所有用例整体运行时长远小于模拟退火算法，部分用例最终值与初始值相比下降幅度显著；模拟退火最终解普遍优于登山搜索，同参数多次实验结果相比于登山搜索变化不大，具有更高的鲁棒性；而引入轮盘赌策略的模拟退火并未在兼顾速度和效果方面相比经典模拟退火有较大提升，甚至运行时间有所增长。这次试验表明对算法的改进需要进行严谨科学论证，进行算法可行性验证，两种优秀算法的融合不一定产生更优的变体。

关键词: 登山搜索算法 模拟退火算法 轮盘赌策略 贪心思想

# 引言

近年来，我国的制造业和工业发展迅速，许多现代企业积极扩展规模、自主创新，加之市场竞争不断加剧，企业从原来的品种单一、大批量、且生产方式简单，逐步转变为品种多样、小批量且生产方式复杂的模式。车间调度优化以生产效率为目标，将有限的资源合理地分配给各种生产活动，提高企业生产竞争力。因此研究车间调度这一简化模型的优化问题对流水线生产工作有十分重要的意义。

流水车间调度问题可用数学语言具体描述为：

已知：

**1**.流水车间中有工件数为，机器数为；

**2**.每件工件要经过道加工工序，加工顺序为列表，其中 为第台机器；

**3**.加工时间表即矩阵，其中第列为， 为第件工件在第台机器上的加工时间。

结合实际情况，一个工件不能同时在多个机器上加工，一个机器不能同时加工多个工件，我们将其称为**器件约束**。根据以上已知信息，我们需要求出：

**1**.工件加工顺序即列表，为每台机器第个加工的工件的序号；

**2**.加工调度表即矩阵，其中第列为，为工件在第台机器上加工的**完成时刻**。根据器件约束，需满足：

上式即约束不等式；

**3**.调度总完工时间即目标函数，即加工总时间。

在该问题的解空间中，（加工总时间最小）所对应的解即为最优调度方案.

对于上述问题，本文将采用**登山搜索算法**与**模拟退火算法**分别求解，并进行结果对比。

本文对这两种算法做出了关键性改进：对于登山搜索，采用贪心思想生成初始解，使其尽可能找出全局最优；对于模拟退火，除原始算法外，再引入轮盘赌策略选择生成的邻域解，一定程度提高收敛速度，即**轮盘赌模拟退火算法**。针对这三种算法分别进行多组参数实验对比，找出最优参数组。

对于题目所给的十一个测试用例，模拟退火的最优解普遍优于登山搜索，而登山搜索的收敛速度远快于模拟退火，模拟退火远快于轮盘赌模拟退火。对于个别用例，登山搜索最终解受参数选择影响较大，偶尔出现不迭代的情况（即初始值为局部最优）；而模拟退火对于所有用例都有较好的收敛效果，具有更高的稳定性；引入轮盘赌策略并未对模拟退火算法性能有较大提升，且运行时间受参数影响大。

# 算法设计

**2.1 算法简介**

登山搜索算法遵循贪心思想，每次从当前解的邻域解空间中选择一个最优解，直到获得一个局部最优解[1]。这种近似算法由于其搜索范围局限，得到的最终解受初始解影响很大。本文对生成初始解也采用贪心算法，即从0时刻开始紧密安排工件加工，使得算法能在解空间的一个较好的范围内进行搜索。

模拟退火算法是受到现代热处理技术中的金属的物理退火原理启发，通过模拟这种金属的物理加工方式而提出的一种启发式算法[2]。模拟退火与登山搜索类似，关键的不同在于当进行邻域搜索时，若搜索到的解劣于当前解，也有概率接受其为当前解，这使得模拟退火能尽可能的找到全局最优解。为了追求在较快的搜索速度下寻求最优解，本文尝试轮盘赌策略选择邻域解，即轮盘赌模拟退火算法。

**2.2 登山搜索算法**

流水车间调度问题，涉及多对象多流程，以原始的登山搜索算法进行求解较难得到较好的结果，因此本文对登山搜索进行了一小部分改进。

算法参数设置：-每次迭代生成邻域解个数

贪心思想生成初始解：

根据题目所给加工时间表中的数据，从第一个机器开始依次对所有机器安排工件加工。对于机器，从零时刻开始对工件按照初始顺序进行加工：工件加工开始时刻为0，工件加工开始时刻为工件的加工结束时刻；对于机器，以与机器相同的顺序对工件加工：工件加工开始时刻为其在机器上的加工结束时刻，工件加工开始时刻为其在机器上的加工结束时刻和工件的加工结束时刻中的较大值。至此，对所有工件的加工安排均遵循贪心思想，即可得到由贪心法生成的一个目标函数值较小的初始解（也即当前解），使得初始搜索域处于离全局最优解较接近甚至囊括全局最优解的区域。

随机生成邻域解：

在当前解的基础上，随机抽取两个工件交换其加工顺序，并按照上述贪心法生成邻域解。重复该步骤次，即生成个可能不同（有概率抽取到相同的工件组合）的邻域解，选择其目标函数值最小的解即为邻域最优解。的选取对算法最终解的质量至关重要：过小意味着邻域搜索范围太小，较难产生更优解；过大会导致收敛速度较慢，算法效率降低。的最优值需要通过多次实验寻找，并且对于不同加工时间表最优值一般不同。

其余步骤遵循经典登山搜索算法，下图为本文登山搜索算法流程框图：

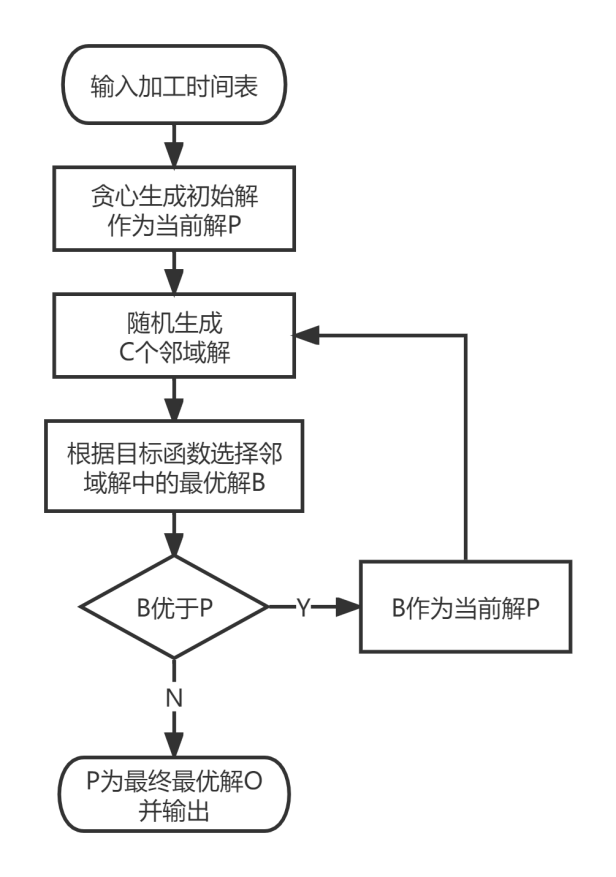


图 1登山搜索算法流程框图

**2.3 模拟退火算法**

模拟退火与登山搜索类似，关键不同在于进行邻域搜索时，有一定概率接受劣于当前解的邻域解，这使得算法能够跳出局部最优范围，相比于登山搜索的局部深挖，模拟退火在前期的迭代追求更广阔的搜索范围。在迭代次数足够多即算法进行充分搜索后，算法往往能给出全局最优解。然而为了能找到全局最优解，需要设置较高的初始温度，较低的最终温度，较慢的降温速度，这会耗费较长时间。在生成邻域解这一关键步骤，本文考虑引入轮盘赌策略，尝试在一定程度克服上述困难。

算法参数设置：

-初始温度； -最终温度； -降温速率； -随机生成邻域解个数

贪心思想生成初始解与随机生成邻域解：

与登山搜索算法中的策略完全一致；

轮盘赌算法选择邻域解：

轮盘赌选择算法的基本思想是：根据个体适应度值在轮盘上进行比例划分，个体适应度值（在此问题中与目标函数有关）和被轮盘赌选择的概率成正相关[3]，这种算法可以防止适应度值较小的个体直接被淘汰。由此可见，轮盘赌的引入可以提升模拟退火跳出局部最优，扩大搜索范围的能力。理论上，轮盘赌模拟退火可以在更低的初始温度、更快的降温速度的情况下找出全局最优解，算法效率提升。

其中针对个体适应度值的计算，由于问题最优解目标函数值为最小值，即目标函数值越小意味着适应度越高，因此需要通过取反运算将目标函数值转化为适应度值，从而适应度值计算公式如下：

对于第个邻域解，为其适应度值，为其目标函数值。将个邻域解的目标函数值的最小值与最大值相加并减去该邻域解的目标函数值，就实现了目标函数值到适应度值的转化。

下段伪代码阐述轮盘赌在本问题中的具体实现。

**function** Roulette-Selection(dict)

**inputs**: dict, contains R neighbor nodes

**local variables**: f, an array saving fitness of R neighbor nodes

q, an array saving cumulative probabilities of R neighbor nodes

c, an array saving selected times of R neighbor nodes

f ← **Fitness** (dict)

q ← **probability** (f)

**loop do for length** (dict)

r ← **random** ()

**loop do for length** (dict)

**if** r <= q[i]

c[i] ← c[i]+ 1

**break**

**return** dict[**index** (**max** (c))]

显然，当时，算法变为原始模拟退火算法  
降温方式：

本文采用指数式降温方式，即，其中为小于1的实数。这种降温方式使得在每一温度下有足够的转移尝试，收敛速度整体较慢。

其余步骤与原始模拟退火算法相同，下图为（轮盘赌）模拟退火算法流程框图

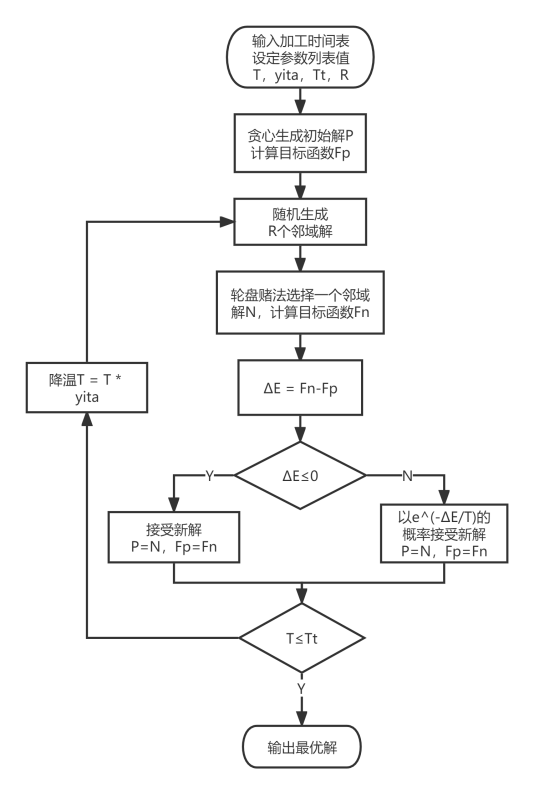


图 2（轮盘赌）模拟退火算法流程框图

【算法核心部分复杂度分析】

已知机器数，工件数，参数值

1. 时间复杂度  
   i.轮盘赌随机生成解模块：  
    第台机器给件工件加工，对于每件工件进行次器件约束检查，共台机器，则有；共生成个邻域解，故；  
   ii.主循环体（退火）模块：  
    某一温度（参数中的下标代表最终terminal，而非一个数字；与之类似，式中的），从而令，可得；  
   故。结合本问题测试用例。  
   因此时间复杂度为。
2. 空间复杂度  
   测试用例数据以二维矩阵形式存储，则。每次迭代生成个随机解，而，故空间复杂度为。

# 实验

## 实验设置

实验环境：

**表 1**实验环境

|  |  |  |  |
| --- | --- | --- | --- |
| Language | IDE | Requirements | Interpreter |
| python | PyCharm | NumPy, Matplotlib | Python3.9 |

实验最优参数：

登山搜索算法：，总运行时间平均值：29.23s；

模拟退火算法：，总运行时间平均值：15.63s。

运行主函数代码 main.py 即可重复实验。

## 实验结果

本节内容对登山搜索算法、经典模拟退火算法和轮盘赌模拟退火算法的实验结果分别进行分析，并将三种算法进行对比。

**3.2.1 登山搜索算法**

多次实验结果表明，登山搜索算法搜索次数受邻域解搜索个数影响较小，且所有测试用例基本在10次迭代以内结束搜索，初步推断这是仅寻求局部最优解的直接体现。下图为第十个测试用例在邻域解搜索个数为200时的算法迭代过程，Value 为加工总时间。

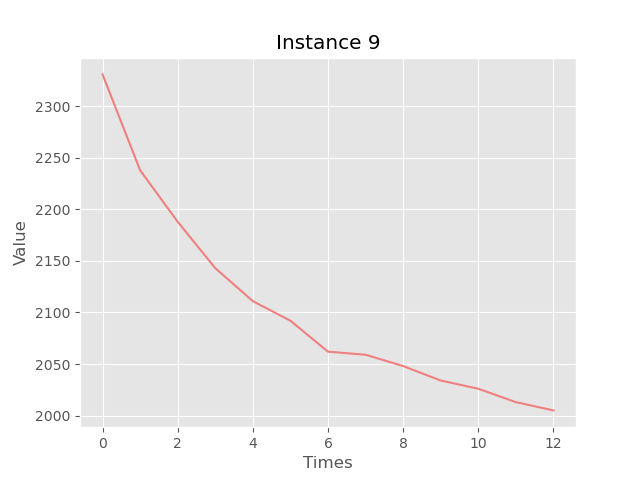


图 3 Instance9登山搜索迭代曲线图

图中曲线完全呈下降趋势，充分展现了登山搜索算法所包含的完全贪心思想。与此同时，由图中可以看出，本次搜索仅经过了12次迭代，或许可以表明登山搜索的特点是“局部深挖”而非广泛搜索，很容易陷入局部最优解。

为探究值（邻域解搜索个数）对登山搜索效果的影响，下表对比了在不同值下求解出部分测试用例最终解。（每个值均进行多次实验，取出现次数最多或相近的结果）

**表 2**不同C值的部分测试用例最终解

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instance | C=100 | C=200 | C=500 | C=1000 |
| 0 | 7038 | 7038 | 7038 | 7038 |
| 1 | 8617 | 8617 | 8617 | 8647 |
| 3 | 7594 | 8626 | 7594 | 7594 |
| 5 | 8047 | 7828 | 7832 | 8047 |
| 6 | 1583 | 1506 | 1562 | 1550 |
| 7 | 2043 | 2046 | 2010 | 2010 |
| 8 | 1133 | 1121 | 1116 | 1133 |

可以看出，实验出现两种情况：1.在不同值下大部分用例最终解十分相近或相同；2.个别用例在较大值下结果劣于较小值。

对于情况1：以Instance0为例，最终解始终稳定在7038即最优解，这说明贪心生成初始解发挥了作用，算法最初就已处于最优解附近的区域，一定程度地降低了迭代次数，提升算法收敛速度。

对于情况2：以Instance1为例，时最终解反而劣于较小值。这表明在本问题中，登山搜索算法所求得局部最优解表现出受随机性影响较大。一旦当前解跳入一个较劣的局部最优区域，即便每次迭代搜索范围大，也不一定得到较好的最终解。

整体上，登山搜索算法在本文所研究的这类调度问题上受随机性影响大，表现出较低的鲁棒性。贪心生成初始解策略在一定程度上能够缓解这种影响。

**3.2.2 模拟退火算法**

模拟退火算法涉及多个参数，具体设置参见本节开篇。本小节将对模拟退火算法引入轮盘赌策略前后进行实验结果对比分析。

Ⅰ 经典模拟退火算法：

多组参数实验对比发现，相比于参数初始温度、参数最终温度，模拟退火算法对参数降温速率更为敏感，越大的往往能带来更优的最终解。对上述三个参数进行多组对比实验，结果如下一系列表格所示。

**表 3**不同 T0、Tt 组合下最终解对比

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Instance1 | Instance2 | Instance3 | Instance4 | Instance5 | Instance6 |
| 10000 | 1e-5 | 8530 | 7166 | 7399 | 8003 | 7767 | 1485 |
| 1000 | 1e-5 | 8655 | 7166 | 7433 | 8003 | 7821 | 1481 |
| 100 | 1e-5 | 8530 | 7290 | 7433 | 8003 | 7822 | 1485 |
| 10000 | 1e-10 | 8420 | 7166 | 7400 | 8003 | 7749 | 1450 |
| 1000 | 1e-10 | 8424 | 7166 | 7366 | 8003 | 7822 | 1502 |
| 100 | 1e-10 | 8366 | 7166 | 7312 | 8003 | 7767 | 1447 |
| 50 | 1e-5 | 8655 | 7166 | 7399 | 8003 | 7767 | 1445 |

上表为在时的不同组合下实验数据，取Instance1-6展示，其余类似。可以看出，时，和下的最终解差别不大，这一特点在Instance2和Instance4尤为突出。这可能与贪心生成初始解策略有关，具体解释在上文已提及。为验证这一点，程序将算法迭代过程以曲线图绘制出来，以Instance2为例如下图所示。

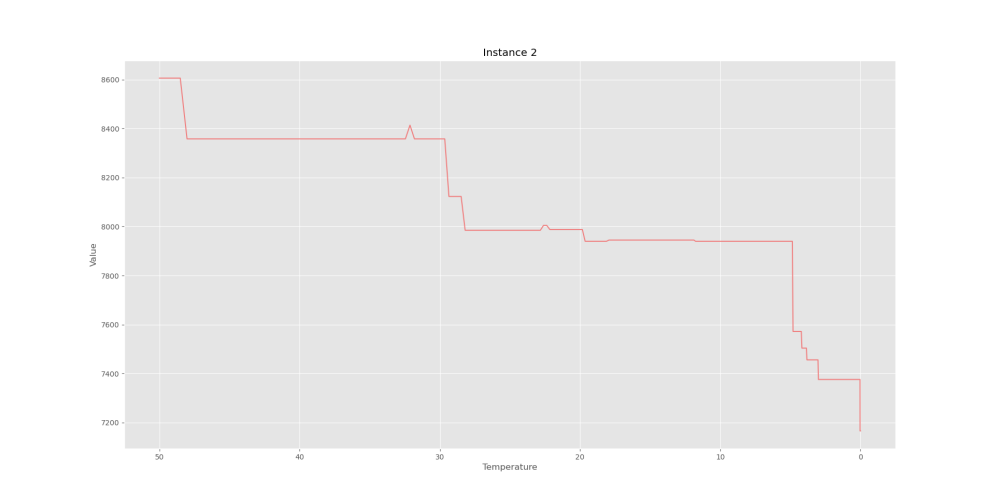


图 4 Instance2 T0=50迭代曲线图

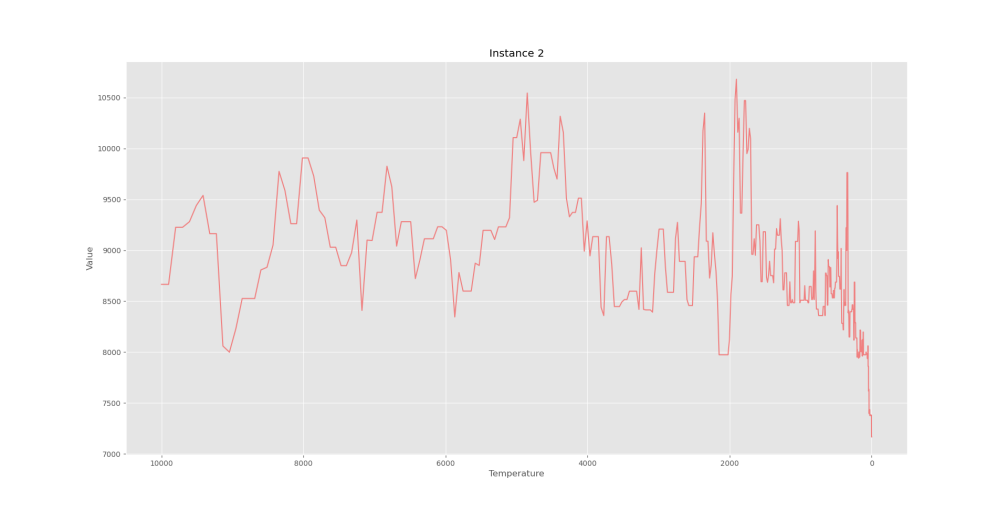


图 5 Instance2 T0=10000迭代曲线图

显然，在较高温度下，贪心生成初始解策略并未发挥作用，这是因为在高温下“分子活动性强”，即算法会在更广阔的空间内搜索，跳出局部最优解，在图中表现出曲线极为颠簸并且偶有高峰。而在温度较低时，“活动性减弱”，求解结果逐渐收敛，表现出类似登山搜索的贪心行为，也如图1所示。可以明显看出，低温下曲线较平缓且一直呈下降趋势，此时贪心生成初始解策略效果显现。

**表 4**不同yita 部分测试用例最终解对比

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Instance1 | Instance2 | Instance3 | Instance4 | Instance5 | Instance6 |
| 0.99 | 8366 | 7166 | 7312 | 8003 | 7767 | 1447 |
| 0.97 | 8424 | 7376 | 7459 | 8003 | 7822 | 1496 |
| 0.95 | 8420 | 7166 | 7531 | 8003 | 7835 | 1540 |
| 0.90 | 8477 | 7376 | 7543 | 8369 | 7835 | 1547 |
| 0.80 | 8885 | 7794 | 7543 | 8423 | 7876 | 1657 |

上表为在时不同下的实验数据。随降低，最终解质量下降，且在下结果差距已经较大。这表明在模拟退火算法当中，降温方式及其参数是敏感项，应通过严谨调整找出最优参数。下图为和时的最终解曲线图。

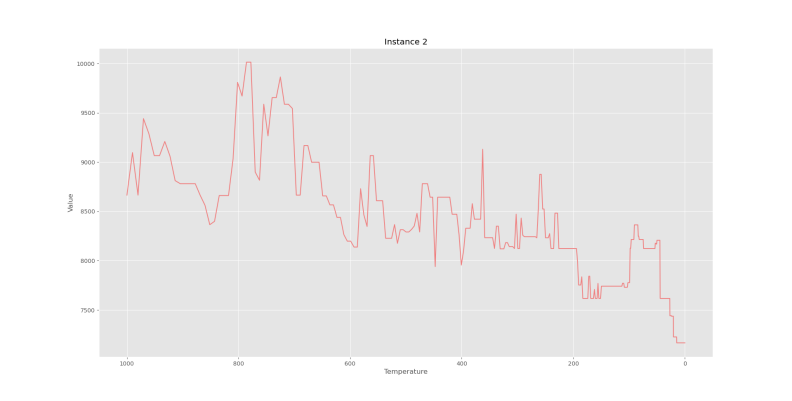


图 6 yita=0.99 迭代曲线图

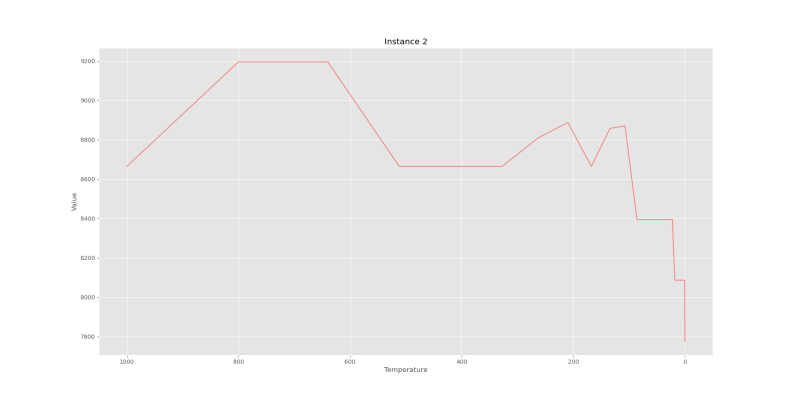


图 7 yita=0.80 迭代曲线图

越大，在其它参数相同下意味着迭代次数越大，也就意味着算法能在相同温度跨度下进行更充分地搜索，最终自然能收敛至全局最优解。当然，较高的会带来更长的运行时间。结果表明，时所有用例的总运行时长约为9.6s，时约为0.4s，差距明显。

整体上，经典模拟退火算法在本文所研究的问题当中，温度的设置对最终解影响较小，同时需要设置较低的降火速率才能够表现出较好的求解结果，而这就带来了运行时间长的问题。下面将通过实验探究轮盘赌模拟退火算法是否能解决上述问题。

Ⅱ 轮盘赌模拟退火算法：

令，对不同的轮盘赌参数做多次实验对比。以Instance6为例，结果如下表所示：

**表 5**不同R的Instance6最终解对比

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| R | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | time |
| 1 | 1530 | 1476 | 1482 | 1506 | 1571 | 1504 | 1507 | 1517 | 1508 | 1478 | 0.323s |
| 5 | 1521 | 1565 | 1548 | 1467 | 1501 | 1540 | 1512 | 1507 | 1530 | 1494 | 1.328s |
| 10 | 1548 | 1503 | 1475 | 1489 | 1531 | 1508 | 1507 | 1522 | 1482 | 1583 | 2.674s |
| 20 | 1547 | 1523 | 1474 | 1476 | 1475 | 1533 | 1492 | 1489 | 1493 | 1511 | 5.249s |
| 50 | 1530 | 1522 | 1492 | 1470 | 1523 | 1522 | 1470 | 1501 | 1488 | 1525 | 13.15s |
| 100 | 1522 | 1498 | 1492 | 1496 | 1490 | 1499 | 1474 | 1512 | 1540 | 1508 | 27.07s |

对于每一个，重复进行十次实验，运行时间取平均值。可以看到，对于Instance6，模拟退火算法仍表现出受随机性影响大的特点，10次实验结果普遍均不相同，并且跨度较大；轮盘赌策略的引入并未对此种情况进行缓解，且大大增加了运行时长。换言之，相同初始温度、最低温度下，轮盘赌模拟退火的表现并不优于原始模拟退火。这表示轮盘赌的引入对于在本文所研究的问题下的模拟退火算法并不适用，其原因可能是适应度函数设置不够契合该调度问题，亦或是生成随机解的方式不能与轮盘赌策略更好地融合。对于其他用例，轮盘赌模拟退火算法表现类似，不再赘述。

考查Instance6的具体数据：20件工件和10台机器，且加工时间差异大，分布无规律。对于这类数据杂乱的情况，模拟退火算法表现出较低的稳定性，最终解一定程度受到每次迭代的中间解的影响，未能稳定收敛于全局最优解。模拟退火算法的鲁棒性有待提高。

Ⅲ 登山搜索算法 vs 模拟退火算法

**表 6**算法运行结果比较

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 算法 | Instance1 | Instance3 | Instance5 | Instance6 | Instance7 | Instance8 | Instance9 | Instance10 | Time |
| HC | 8617 | 7594 | 7821 | 1506 | 2010 | 1116 | 1969 | 3321 | 29.23s |
| SA | 8366 | 7312 | 7720 | 1469 | 1973 | 1111 | 1949 | 3284 | 15.63s |

上表为登山搜索算法和经典模拟退火算法经过多次参数调整、多次重复实验所得的部分测试用例最优解（所有测试用例结果以及具体方案见附录）。HC代表登山搜索算法，SA代表经典模拟退火算法。

参数设置：

登山搜索算法：；

模拟退火算法：。

显然，模拟退火算法性能优于登山搜索算法：在各自最优的参数组下，模拟退火求得最优解普遍优于登山搜索，总运行时长也小于登山搜索。这再一次印证了模拟退火以一定概率接受劣解这种策略的优越性，使其相较于登山搜索更接近全局最优解。

下图展示了模拟退火算法对于Instance0给出的调度方案。可以看出该方案满足前文所述的器件约束，并以最小时间紧密安排工件加工，验证了算法结果的合理性。

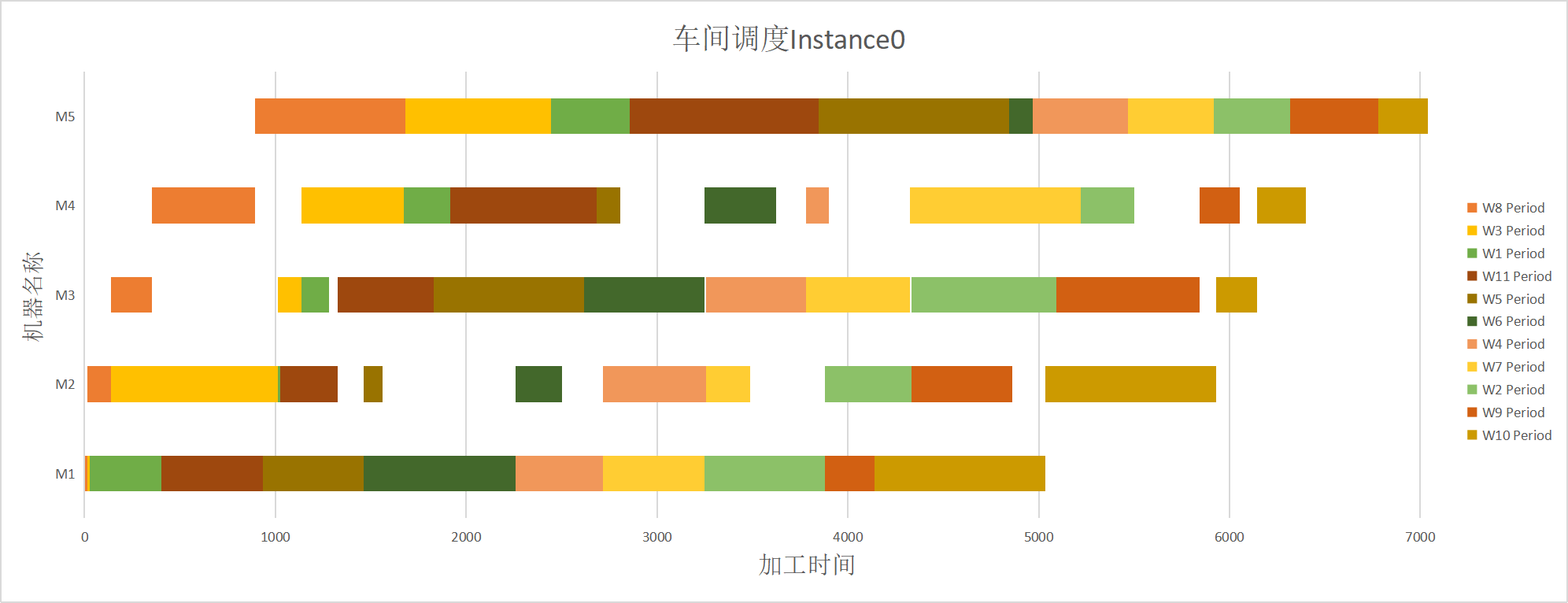


图 8 Instance0 调度时序图

综上所述，模拟退火算法对于该调度问题整体表现优于登山搜索算法，在经过严谨的参数调整下可以得到较好的最优解，且相较于登山搜索算法有更高的鲁棒性，但在个别用例上性能下降，或许可以对随机生成邻域解的方法进行改进；轮盘赌策略并未对模拟退火算法带来增益，且增加了运行时长，对算法的改进以失败告终。

# 总结

**4.1 工作总结**

本文以流水车间调度问题为研究对象，对问题进行数学形式的具体描述，借此分析流水车间调度问题的核心。运用改进后的（贪心思想生成初始解）登山搜索算法、经典模拟退火算法和引入轮盘赌策略的轮盘赌模拟退火算法，求解了11个流水车间调度问题，给出其最小加工时间和具体调度方案。根据实验结果对三种算法进分别进行了多组参数实验对比分析和算法间的横向对比分析，评估了算法性能和特点。主要结论如下：

1. 登山搜索算法：对于本问题的测试用例整体表现良好。登山搜索表现出明显的贪心思想，每次迭代都能找到更优解，这导致算法很快陷入局部最优解并停止迭代，即便增加邻域解搜索个数，也会受到随机性的影响，难以收敛到全局最优解，整体表现出较低的鲁棒性。采用贪心生成初始解策略可以缓解陷入局部最优解的情况，对于部分用例可以求出全局最优解。
2. 模拟退火算法：整体表现优于登山搜索算法，对于每一个测试用例均能求得与登山搜索相同或更好的最终解，且总运行时长甚至短于登山搜索算法。对于大部分测试用例，最终解基本相同，相比于登山搜索有更高的鲁棒性；个别用例数据杂乱，情况复杂，此时算法性能下降。轮盘赌策略的引入并未对模拟退火算法有提升，是一个失败的算法改进。

本文所研究的流水车间调度问题是各类调度优化问题的最基本简化模型，上述算法能够起到良好的优化效果。但结合实际情况：工件的加工顺序可能不同、所需机器数量可能不同、多个工件可能需要在同一台机器组合加工等等。这就需要更加完善，更加稳定的智能优化算法例如遗传算法、粒子群优化算法等来完成，因此本文完成的工作也只是车间调度问题的很小一部分，还有更多复杂的问题等待解决。

**4.2 工作展望**

（1）本文利用登山搜索算法时虽然考虑到初始解对最终解的影响，采用贪心生成初始解，但对于部分具体问题仍无法避免陷入局部最优。可以考虑采用局部束搜索算法实现更大的搜索范围，寻求全局最优解。

（2）对于登山搜索算法的鲁棒性，这应该也与生成邻域解的方式有关。本文采用随机抽取两个工件交换其加工顺序，虽然增大了随机搜索的范围，但也增加了算法的不稳定性。后续还会对如何更加稳定地生成随机解作进一步研究。

（3）轮盘赌策略改进模拟退火被实验证明不可行，这或许由诸多因素导致：与本文研究的问题不契合、适应度函数设置不能很好地反映解的适应度等。通过阅读大量参考文献得知，在采用一个算法进行问题求解之前，应对算法进行可行性验证。本文忽略了这一步严谨的科学论证，导致最后改进的失败，在以后的学术研究中应多加注意。对于模拟退火的改进，可以采用与粒子群搜索相结合的方法，在杜佰林对大荔县水资源优化配置的研究中[4]，“模拟退火粒子群算法”得到了很好的应用，可以考虑应用在本问题中。

参考文献:

1. 赵科. 无人机的智能物流调度优化方法及其应用[D].南京邮电大学,2020.DOI:10.27251/d.cnki.gnjdc.2020.000623.
2. 石弘毅. 基于智能优化算法的公交车辆调度方法研究[D].北京邮电大学,2021.DOI:10.26969/d.cnki.gbydu.2021.001593.
3. 马洁莹. 基于轮盘赌策略的混沌萤火虫算法研究[D].西安电子科技大学,2018.
4. 杜佰林. 基于模拟退火粒子群算法的大荔县水资源优化配置研究[D].西安理工大学,2021.DOI:10.27398/d.cnki.gxalu.2021.000819.

附录

所有测试用例最优解及其调度方案

Min\_Time 最小加工时间

Processing Order 工件加工顺序

所有工件在每个机器上的加工完成时间以列表形式列出

+++++++++++++++++++++++++++++

Instance 0

Min\_Time = 7038

Solution:

Processing Order: [7, 4, 0, 2, 3, 10, 6, 8, 5, 1, 9]

Machine1 [14, 542, 917, 929, 1389, 1921, 2453, 2710, 3506, 4138, 5034]

Machine2 [138, 643, 929, 1805, 2347, 2649, 2879, 3406, 3751, 4590, 5930]

Machine3 [352, 1432, 1574, 1929, 2870, 3371, 3914, 4667, 5299, 6057, 6271]

Machine4 [895, 1556, 1819, 2463, 2990, 4136, 5032, 5242, 5674, 6335, 6593]

Machine5 [1680, 2679, 3091, 3856, 4355, 5343, 5795, 6258, 6381, 6779, 7038]

+++++++++++++++++++++++++++++

Instance 1

Min\_Time = 8366

Solution:

Processing Order: [6, 2, 7, 4, 1, 0, 5, 3]

Machine1 [654, 1308, 2097, 2553, 3342, 3798, 4587, 4908]

Machine2 [975, 1431, 2244, 3342, 3465, 4452, 5241, 5697]

Machine3 [1295, 2063, 2364, 3814, 4183, 5304, 5890, 6471]

Machine4 [2053, 3028, 3667, 4179, 4861, 5449, 6714, 7135]

Machine5 [2916, 3503, 3688, 4715, 5442, 6081, 7039, 7167]

Machine6 [3368, 3828, 4691, 5567, 5963, 6506, 7051, 7314]

Machine7 [3824, 4284, 5480, 6221, 6344, 6720, 7372, 8161]

Machine8 [4613, 5267, 6134, 6344, 7133, 7787, 8243, 8366]

+++++++++++++++++++++++++++++

Instance 2

Min\_Time = 7166

Solution:

Processing Order: [6, 2, 3, 10, 7, 12, 0, 1, 8, 11, 4, 9, 5]

Machine1 [25, 37, 382, 585, 1459, 1761, 2415, 2736, 2850, 3546, 4224, 5009, 5972]

Machine2 [49, 196, 968, 1178, 1976, 2488, 2635, 3256, 4152, 4936, 5468, 6011, 6156]

Machine3 [191, 826, 1182, 1881, 2000, 2709, 3054, 4045, 4672, 5791, 6066, 6402, 6704]

Machine4 [780, 1081, 2048, 2832, 3828, 4173, 4620, 5322, 5863, 6375, 6707, 6941, 7166]

+++++++++++++++++++++++++++++

Instance 3

Min\_Time = 7312

Solution:

Processing Order: [10, 5, 11, 4, 9, 8, 2, 1, 3, 6, 7, 0]

Machine1 [210, 333, 457, 667, 1210, 2086, 2962, 3751, 4294, 4750, 5539, 5995]

Machine2 [355, 569, 816, 1601, 2133, 2831, 3594, 4605, 4750, 5502, 5682, 6532]

Machine3 [469, 901, 1654, 2620, 2720, 3153, 4182, 4830, 5499, 5646, 6437, 6655]

Machine4 [870, 1757, 1971, 2767, 3088, 3699, 5078, 5606, 5931, 6252, 6864, 7078]

Machine5 [1746, 2300, 2843, 3719, 4508, 4964, 5534, 5729, 6720, 6930, 7053, 7312]

+++++++++++++++++++++++++++++

Instance 4

Min\_Time = 8003

Solution:

Processing Order: [3, 8, 6, 12, 13, 5, 9, 0, 1, 2, 10, 7, 11, 4]

Machine1 [214, 418, 1071, 1083, 1540, 1758, 2543, 2999, 3788, 4418, 5114, 5328, 5860, 6433]

Machine2 [471, 1336, 1478, 1823, 2501, 3033, 3354, 4210, 5140, 5354, 5478, 6025, 6037, 7329]

Machine3 [791, 1481, 1628, 2677, 2800, 3785, 4548, 5511, 5532, 6007, 6221, 6753, 7010, 7453]

Machine4 [1544, 2071, 2724, 3612, 4611, 5139, 5675, 6371, 6691, 6833, 7047, 7261, 7789, 8003]

+++++++++++++++++++++++++++++

Instance 5

Min\_Time = 7720

Solution:

Processing Order: [4, 1, 3, 0, 2, 7, 5, 9, 8, 6]

Machine1 [255, 588, 810, 1143, 1395, 2283, 2838, 3837, 4726, 5284]

Machine2 [732, 843, 1047, 2134, 2356, 3321, 3887, 4776, 5364, 6263]

Machine3 [855, 1518, 1632, 3130, 3352, 4228, 4684, 4986, 5907, 7052]

Machine4 [1398, 1974, 2850, 3253, 4141, 4765, 4975, 5618, 6761, 7176]

Machine5 [1541, 2759, 3602, 3747, 4355, 4910, 5673, 6124, 7008, 7708]

Machine6 [1683, 3291, 4134, 4368, 4954, 4968, 6246, 7102, 7629, 7720]

+++++++++++++++++++++++++++++

Instance 6

Min\_Time = 1469

Solution:

Processing Order: [13, 1, 9, 8, 11, 3, 15, 12, 4, 10, 16, 19, 18, 7, 2, 0, 14, 17, 6, 5]

Machine1 [27, 52, 104, 143, 198, 239, 254, 262, 327, 399, 458, 517, 537, 630, 706, 802, 836, 929, 978, 1059]

Machine2 [38, 96, 150, 233, 276, 282, 287, 378, 469, 473, 528, 541, 632, 721, 783, 882, 968, 972, 987, 1128]

Machine3 [132, 142, 214, 288, 335, 338, 377, 459, 544, 573, 616, 709, 782, 813, 861, 938, 1055, 1098, 1110, 1193]

Machine4 [170, 211, 227, 320, 367, 400, 423, 552, 582, 676, 725, 751, 823, 837, 915, 986, 1065, 1103, 1164, 1286]

Machine5 [203, 275, 329, 347, 454, 531, 547, 566, 629, 761, 784, 795, 923, 960, 1007, 1021, 1129, 1213, 1288, 1349]

Machine6 [270, 327, 391, 400, 551, 595, 596, 682, 737, 812, 837, 844, 961, 1018, 1053, 1141, 1171, 1268, 1354, 1357]

Machine7 [278, 346, 436, 529, 592, 638, 695, 759, 810, 841, 892, 958, 1036, 1069, 1141, 1191, 1238, 1290, 1388, 1432]

Machine8 [333, 374, 516, 594, 680, 730, 785, 827, 828, 906, 978, 1020, 1045, 1165, 1219, 1234, 1289, 1368, 1400, 1449]

Machine9 [432, 504, 535, 669, 697, 749, 847, 917, 953, 1003, 1012, 1074, 1150, 1197, 1246, 1313, 1382, 1413, 1445, 1455]

Machine10 [450, 531, 600, 742, 772, 815, 882, 920, 1026, 1042, 1043, 1173, 1244, 1289, 1345, 1410, 1436, 1447, 1453, 1469]

+++++++++++++++++++++++++++++

Instance 7

Min\_Time = 1973

Solution:

Processing Order: [11, 4, 14, 5, 15, 8, 19, 13, 16, 12, 0, 18, 10, 1, 7, 2, 17, 9, 6, 3]

Machine1 [37, 75, 92, 119, 195, 287, 334, 389, 470, 474, 534, 632, 648, 727, 768, 862, 925, 994, 1017, 1040]

Machine2 [56, 107, 177, 191, 289, 333, 382, 454, 519, 576, 646, 685, 723, 780, 871, 928, 998, 1067, 1122, 1125]

Machine3 [111, 179, 212, 289, 343, 355, 471, 492, 602, 628, 697, 788, 824, 898, 988, 1007, 1021, 1138, 1205, 1287]

Machine4 [113, 186, 293, 313, 377, 414, 504, 603, 675, 704, 778, 793, 872, 985, 1031, 1108, 1193, 1247, 1253, 1374]

Machine5 [209, 258, 317, 409, 421, 496, 603, 630, 706, 797, 803, 858, 964, 1003, 1068, 1178, 1259, 1295, 1359, 1419]

Machine6 [219, 307, 352, 483, 511, 531, 613, 683, 778, 891, 895, 923, 984, 1019, 1164, 1268, 1309, 1405, 1473, 1527]

Machine7 [300, 379, 456, 516, 540, 554, 667, 725, 869, 912, 939, 982, 1039, 1083, 1263, 1304, 1373, 1408, 1492, 1603]

Machine8 [329, 410, 477, 560, 565, 608, 761, 839, 877, 942, 947, 1079, 1168, 1263, 1309, 1382, 1436, 1463, 1565, 1663]

Machine9 [331, 480, 563, 606, 665, 691, 813, 902, 942, 972, 1002, 1114, 1243, 1304, 1350, 1419, 1446, 1479, 1657, 1724]

Machine10 [365, 488, 582, 658, 669, 755, 857, 927, 1035, 1086, 1165, 1188, 1300, 1307, 1401, 1443, 1511, 1572, 1753, 1851]

Machine11 [448, 514, 626, 668, 715, 840, 930, 1016, 1119, 1150, 1223, 1273, 1389, 1409, 1442, 1471, 1537, 1666, 1756, 1919]

Machine12 [449, 595, 696, 777, 870, 884, 997, 1076, 1182, 1244, 1293, 1392, 1454, 1513, 1562, 1585, 1643, 1762, 1783, 1933]

Machine13 [494, 681, 786, 816, 902, 956, 1041, 1140, 1249, 1302, 1393, 1452, 1490, 1587, 1604, 1620, 1663, 1847, 1898, 1943]

Machine14 [517, 741, 820, 913, 991, 1045, 1081, 1216, 1285, 1331, 1413, 1515, 1551, 1620, 1684, 1781, 1845, 1881, 1901, 1960]

Machine15 [581, 815, 910, 984, 1039, 1142, 1240, 1309, 1393, 1444, 1507, 1610, 1679, 1721, 1789, 1864, 1905, 1939, 1964, 1973]

+++++++++++++++++++++++++++++

Instance 8

Min\_Time = 1111

Solution:

Processing Order: [13, 1, 2, 0, 6, 7, 3, 9, 4, 18, 10, 15, 5, 12, 16, 19, 14, 17, 11, 8]

Machine1 [32, 43, 106, 140, 184, 213, 235, 297, 373, 401, 451, 510, 530, 580, 621, 687, 699, 793, 882, 936]

Machine2 [56, 121, 188, 194, 224, 313, 359, 455, 459, 518, 531, 609, 631, 687, 707, 740, 828, 876, 951, 963]

Machine3 [79, 125, 191, 257, 312, 325, 447, 516, 550, 560, 608, 704, 711, 719, 802, 836, 900, 926, 1008, 1029]

Machine4 [166, 167, 264, 349, 417, 513, 514, 595, 604, 685, 725, 784, 787, 854, 919, 927, 941, 1034, 1035, 1109]

Machine5 [228, 301, 401, 461, 553, 624, 690, 743, 819, 839, 876, 924, 952, 998, 1018, 1023, 1036, 1039, 1109, 1111]

+++++++++++++++++++++++++++++

Instance 9

Min\_Time = 1949

Solution:

Processing Order: [17, 16, 12, 7, 1, 14, 4, 11, 6, 8, 18, 10, 2, 3, 19, 13, 15, 9, 0, 5]

Machine1 [17, 94, 99, 103, 175, 182, 224, 296, 376, 422, 483, 536, 628, 677, 706, 786, 854, 877, 936, 957]

Machine2 [103, 105, 149, 203, 257, 273, 285, 313, 475, 498, 512, 627, 654, 704, 712, 805, 927, 1011, 1022, 1083]

Machine3 [135, 136, 245, 288, 324, 370, 425, 478, 566, 620, 690, 770, 785, 884, 890, 963, 1020, 1112, 1116, 1168]

Machine4 [170, 252, 289, 352, 355, 452, 514, 550, 649, 726, 784, 792, 873, 948, 960, 1058, 1062, 1206, 1285, 1334]

Machine5 [176, 254, 310, 396, 455, 552, 589, 598, 660, 786, 810, 851, 959, 989, 1067, 1145, 1211, 1219, 1379, 1423]

Machine6 [226, 303, 354, 474, 497, 634, 658, 682, 775, 839, 875, 964, 1020, 1071, 1099, 1223, 1294, 1304, 1410, 1521]

Machine7 [279, 392, 441, 518, 558, 675, 699, 779, 826, 881, 951, 967, 1112, 1138, 1178, 1254, 1381, 1458, 1532, 1558]

Machine8 [318, 419, 454, 557, 617, 696, 741, 788, 906, 978, 1035, 1122, 1169, 1258, 1271, 1284, 1424, 1516, 1614, 1682]

Machine9 [412, 453, 540, 633, 722, 733, 782, 816, 1006, 1096, 1127, 1184, 1277, 1317, 1378, 1428, 1488, 1580, 1667, 1743]

Machine10 [501, 553, 627, 732, 769, 819, 907, 967, 1093, 1107, 1227, 1302, 1323, 1387, 1406, 1521, 1577, 1675, 1726, 1768]

Machine11 [523, 638, 727, 761, 854, 859, 921, 1061, 1177, 1199, 1248, 1339, 1379, 1447, 1486, 1619, 1649, 1730, 1749, 1774]

Machine12 [598, 664, 730, 799, 921, 949, 982, 1160, 1194, 1267, 1343, 1351, 1456, 1523, 1621, 1701, 1722, 1745, 1780, 1826]

Machine13 [657, 744, 826, 840, 960, 1055, 1140, 1227, 1270, 1364, 1418, 1441, 1540, 1607, 1690, 1747, 1761, 1780, 1826, 1901]

Machine14 [731, 831, 916, 991, 1056, 1140, 1144, 1237, 1363, 1388, 1512, 1600, 1610, 1710, 1724, 1756, 1798, 1860, 1907, 1944]

Machine15 [758, 889, 977, 1016, 1116, 1185, 1205, 1281, 1421, 1435, 1569, 1665, 1756, 1759, 1762, 1799, 1860, 1927, 1937, 1949]

+++++++++++++++++++++++++++++

Instance 10

Min\_Time = 3284

Solution:

Processing Order: [12, 13, 37, 49, 39, 41, 43, 33, 34, 8, 38, 19, 16, 40, 0, 36, 2, 18, 1, 10, 23, 9, 28, 35, 14, 31, 32, 26, 4, 22, 11, 25, 21, 20, 45, 7, 17, 3, 46, 42, 5, 24, 29, 47, 48, 44, 27, 6, 15, 30]

Machine1 [4, 49, 123, 125, 163, 178, 207, 225, 286, 371, 404, 413, 435, 478, 578, 662, 732, 744, 763, 841, 924, 984, 1040, 1065, 1138, 1142, 1193, 1266, 1346, 1429, 1527, 1537, 1620, 1649, 1678, 1762, 1852, 1902, 1975, 2050, 2097, 2108, 2167, 2254, 2340, 2354, 2439, 2479, 2556, 2626]

Machine2 [61, 106, 125, 166, 209, 228, 288, 318, 327, 417, 424, 500, 566, 664, 736, 744, 800, 892, 895, 959, 1005, 1065, 1128, 1219, 1313, 1379, 1421, 1461, 1558, 1653, 1684, 1758, 1773, 1785, 1805, 1885, 1936, 2029, 2048, 2148, 2207, 2239, 2306, 2385, 2484, 2533, 2563, 2626, 2710, 2776]

Machine3 [91, 131, 155, 207, 211, 278, 368, 429, 432, 491, 553, 615, 620, 692, 812, 907, 946, 989, 1008, 1029, 1111, 1113, 1154, 1250, 1396, 1475, 1565, 1569, 1572, 1740, 1782, 1855, 1928, 1999, 2026, 2094, 2181, 2281, 2314, 2356, 2385, 2400, 2473, 2525, 2556, 2634, 2692, 2761, 2772, 2856]

Machine4 [102, 176, 218, 306, 310, 340, 454, 521, 523, 558, 626, 672, 749, 800, 912, 973, 1044, 1065, 1133, 1138, 1140, 1190, 1277, 1279, 1455, 1518, 1650, 1670, 1680, 1769, 1855, 1954, 1986, 2069, 2135, 2217, 2244, 2352, 2388, 2455, 2458, 2485, 2550, 2635, 2660, 2753, 2842, 2863, 2945, 2977]

Machine5 [169, 183, 251, 312, 374, 449, 524, 581, 642, 710, 752, 823, 920, 963, 979, 1064, 1093, 1185, 1214, 1299, 1354, 1444, 1497, 1511, 1512, 1557, 1734, 1785, 1799, 1845, 1938, 2008, 2059, 2115, 2205, 2221, 2309, 2436, 2529, 2553, 2579, 2581, 2641, 2665, 2743, 2798, 2890, 2946, 2956, 3070]

Machine6 [173, 242, 275, 389, 484, 574, 587, 647, 665, 794, 835, 900, 948, 1047, 1056, 1105, 1196, 1260, 1282, 1354, 1408, 1464, 1505, 1608, 1680, 1763, 1792, 1810, 1842, 1912, 1986, 2097, 2103, 2211, 2306, 2351, 2427, 2500, 2550, 2616, 2636, 2679, 2740, 2829, 2839, 2933, 2948, 3021, 3030, 3126]

Machine7 [255, 343, 405, 494, 570, 668, 768, 793, 837, 926, 1004, 1080, 1141, 1154, 1159, 1247, 1347, 1441, 1457, 1472, 1557, 1635, 1715, 1806, 1871, 1926, 1999, 2036, 2128, 2217, 2288, 2371, 2374, 2386, 2393, 2493, 2560, 2627, 2671, 2686, 2736, 2759, 2853, 2903, 2944, 2979, 3061, 3117, 3184, 3210]

Machine8 [332, 385, 418, 574, 665, 703, 854, 864, 942, 960, 1071, 1145, 1227, 1298, 1385, 1389, 1475, 1542, 1555, 1578, 1581, 1691, 1761, 1890, 1952, 1977, 2007, 2054, 2195, 2290, 2339, 2376, 2405, 2475, 2486, 2589, 2609, 2656, 2722, 2767, 2793, 2872, 2958, 2962, 3028, 3074, 3151, 3161, 3211, 3252]

Machine9 [430, 487, 549, 595, 732, 783, 942, 962, 1000, 1097, 1196, 1226, 1289, 1362, 1419, 1505, 1593, 1653, 1740, 1776, 1796, 1858, 1863, 1978, 2023, 2085, 2180, 2241, 2313, 2382, 2454, 2482, 2485, 2561, 2636, 2665, 2740, 2768, 2772, 2864, 2865, 2900, 2996, 3033, 3068, 3092, 3153, 3207, 3254, 3275]

Machine10 [451, 568, 578, 649, 810, 893, 948, 1057, 1131, 1189, 1202, 1264, 1385, 1466, 1481, 1556, 1692, 1738, 1810, 1897, 1954, 1981, 2043, 2069, 2145, 2158, 2237, 2316, 2384, 2417, 2484, 2574, 2598, 2617, 2688, 2755, 2822, 2903, 2927, 2971, 3041, 3070, 3071, 3121, 3122, 3207, 3210, 3250, 3262, 3284]