基于模拟退火和登山算法的流水车间调度问题

LL （北京理工大学计算机学院 计算机科学与技术专业）

摘 要: 流水车间调度问题，是通过合理安排调度顺序使完成任务的总用时最短，以提高车间作业的效率。本文采用智能搜索算法，分别基于登山算法和模拟退火算法，建立了相应的车间调度模型，并用具体用例进行实验模拟，通过调参获得最优调度时间。同时，本文对实验结果进行详细分析：通过绘制调度方案的甘特图和迭代过程折线图，验证了算法原理实现的正确性；通过对比登山算法、模拟退火算法的实验结果，分析了 2 种智能算法的特点，其中模拟退火算法最优解更优，但其运行时间成倍增加；通过算法多参数对比实验，分析了各个参数的影响原因和适宜取值。最后，总结了整体的工作，并分析了不足，提出局部束搜索、优化编码变化策略等解决方案。

关键词: 流水车间调度问题；模拟退火算法；登山算法；局部搜索；智能算法

1 引言

1.1 问题背景

流水车间调度问题 (Flow-shop Scheduling Problem, FSP) 是指通过合理安排系统中有限的资源，解决冲突，从而充分发挥系统性能，并压缩任务完成时间。[1] 1954 年，Johnson 提出最优排序方法，研究了两台机器按次序加工的车间调度问题。[2]1975 年，越民义研究了 n 个零件和 m 台机器的加工顺序问题。[3]

在现实生活中，流水车间的工作流程繁多、协作关系复杂、生产连续性强，这使得车间调度问题直接影响企业的生产效率，并具有现实意义和工程价值。因此，流水车间调度问题作为 NP-Hard 问题，自提出以来便不断吸引人们，发展出许多相关求解最优解的智能算法。

1.2 问题分析

1.2.1 假设

本流水车间调度问题满足以下假设。

- 同一工件不能同时在不同的机器上加工。
- 每台机器在同一时刻只能加工一个工件。
- 工件在不同机器上的加工时间是确定的。
- 工件必须按照规定的工序顺序进行加工，且所有工件的加工顺序相同。
- 每个工件在每台机器上只需加工一次，且工件在机器上的加工一旦开始后不能中断，直至完成。

1.2.2 问题重述

流水车间调度问题可描述为：有 n 个待加工工件 $J = J_1, J_2, ..., J_n$，在 m 台机器 $M = M_1, M_2, ..., M_m$ 上加工。在满足 1.2.1 中假设情况下，采用流水加工的形式，即每个工件加工顺序相同——从第一台机器 M_1 到最后一台机器 M_m。设 $T_{i,j}$ 表示第 i 个工件在第 j 台机器上的加工时间，$1 \leq i \leq n$，$1 \leq j \leq m$。$E_{i,j}$ 表示第 i 个工件在第 j 台机器上加工结束时间。$Process\_T$ 表示完成任务的总加工时间。

其具体的数学描述如下：

$$\textbf{min} \quad Process\_T = max(E_{i,m})$$

$$\textbf{s.t.}: \begin{cases} E_{i,j+1} - E_{i,j} \geq T_{i,j} \\ E_{i,1} \geq 0 \\ \forall\, 1 \leq k \leq n,\ E_{k,j} - E_{i,j} \leq T_{k,j}\ or\ E_{i,j} - E_{k,j} \leq T_{i,j} \\ 1 \leq i \leq n \\ 1 \leq j \leq m \end{cases}$$

经计算，获得车间调度方案，即工件在每道工序上的加工顺序，使得完成任务的总加工时间达到最小。

1.3 解题思路

本文基于登山算法和模拟退火算法两种智能算法，以求解流水车间调度问题的最优解。登山算法通过局部搜索更优解，通过更新迭代的方式进行深挖 (Exploration)。而模拟退火算法则增加了探索 (Exploitation) 的过程，不仅以找到最优解为目标深挖，而且有一定概率探索新的区域，这在一定程度上解决了容易陷入局部最优的问题。实验结果表明，对于同一车间调度问题，登山算法求出的最优解比模拟退火算法差，这也验证了探索 + 深挖的思想相比于只深挖具有更加优秀的表现。

2 算法设计

2.1 登山算法

登山算法 (Hill Climbing, HC) 是一种启发式方法，是对深度优先搜索 (DFS) 的改进，通过局部搜索的方式寻找最优解。

2.1.1 算法关键操作介绍

登山算法参考现实登山的情景，在未知最高峰（即最优解）的情况下，以向上攀爬为唯一目标。通过不断尝试邻近的路线，如果海拔相对更高，则更可能是通向山峰的路线，于是更新当前位置向上攀爬；反之则保持不动。算法过程具体如下。

1、选择初始状态。本文通过随机的方式，在解空间中初始化初始调度方案。

2、尝试邻居状态。以一定的步长到达邻居状态，并计算邻居节点的值。

3、判断是否更新。如果邻居节点的值优于当前位置的值，则更新当前位置为邻居节点；否则当前位置保持不变，返回上一步。

登山算法关键是类似贪心的方式寻找最优解，但这也导致登山算法具有依赖于初始状态、容易陷入局部最优的问题。

2.1.2 具体步骤

登山算法的流程图如下：

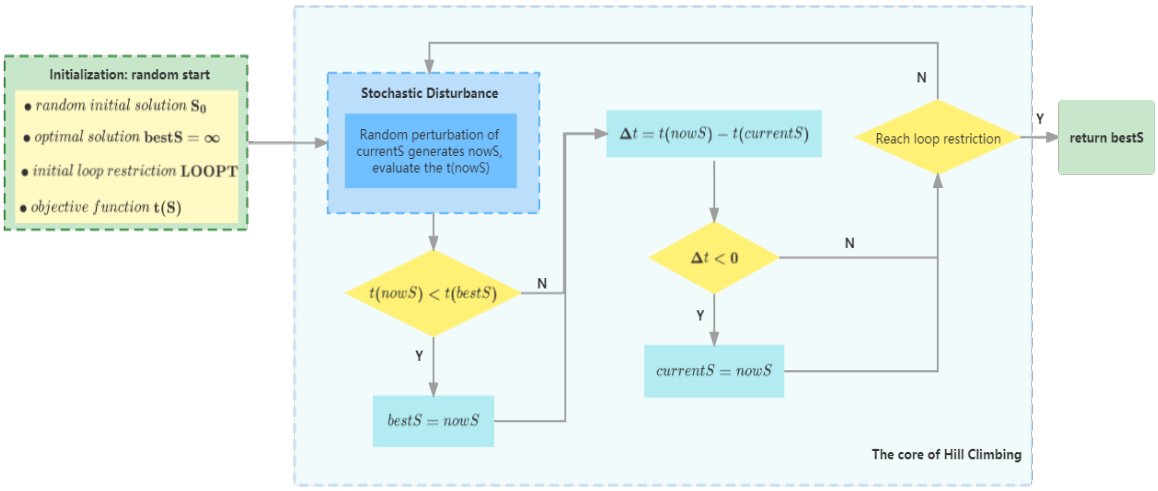

图 2-1 登山算法流程图

Figure 2-1 The flow chart of hill climbing algorithm

2.1.3 时空复杂度分析

1) 时间复杂度

由于计算当前策略任务的完成时间为动态规划，外循环为机器数 m，内循环为工件数 n，其时间复杂度为 $O(mn)$。而登山算法部分主要由循环次数 LOOPT 决定。

因此结合这两部分，总的时间复杂度为 $O(mn * LOOPT)$。

2) 空间复杂度

动态规划部分经过优化，只需记录上一状态和当前状态，因此该部分的空间复杂度为 $O(n)$。由于整个算法的空间复杂度主要由动态规划部分决定，所以总空间复杂度为 $O(n)$。

2.2 模拟退火算法

模拟退火算法 (Simulated Annealing, SA) 是一种启发式的蒙特卡洛方法，最早在 1953 年由 Metropolis 提出，并于 1988 年被 Matsuo 运用在车间调度问题的研究上。

2.2.1 算法关键操作介绍

模拟退火算法的思想来源于固体退火，相比于淬火是将固体介质迅速冷却，退火则是把固体加热到一定温度后，保持足够长的时间，以适宜速度进行冷却。而在高温状态下，固体内部粒子熵值较大，粒子的运动呈现无序状态；随着温度的下降，固体内部粒子逐渐趋于有序，最终达到平衡状态。

相比于登山算法的搜索策略，模拟退火算法的改进关键是可以一定概率下山移动 (downhill moves)。这也使得模拟退火不仅具有和登山算法一样的深挖，而且能够以一定概率探索改进解，跳出陷入局部最优的问题。同时，其下山移动的概率和下山移动的步长，随着时间的推移不断减小，这使得算法最终能够收敛。

2.2.2 具体步骤

模拟退火算法的流程图如下：

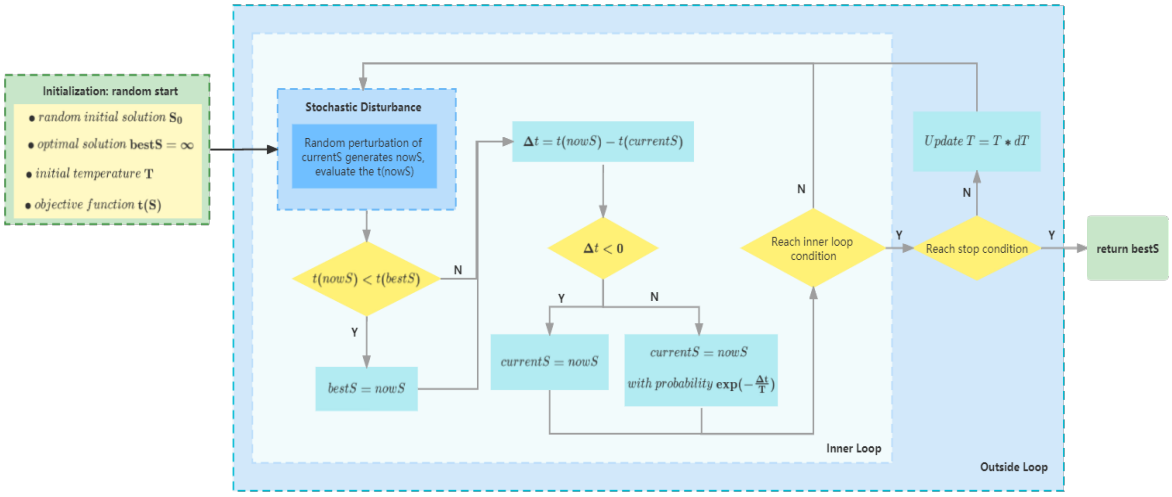

图 2-2 模拟退火算法流程图

Figure 2-2 The flow chart of Simulated Annealing algorithm

其中计算当前策略下任务的完成时间使用动态规划的方法。

Algorithm 1 Algorithm of SA dynamic programming part

Input: process[i][j]: Processing time of workpiece j under machine i
n: The number of machines
m: The number of workpieces
Output: sta[n][m+1]: Total scheduling time

```
sta[i][j]← 0
for each i ∈ [1, n] do
    for each j ∈ [1, m] do
        sta[i+1][j]← max(sta[i+1][j],sta[i][j]+process[i][j])
        sta[i][j+1]← max(sta[i][j+1],sta[i][j]+process[i][j])
    end for
end for
return sta[n][m+1]
```

2.2.3 时空复杂度分析

1) 时间复杂度

动态规划部分，外循环为机器数 m，内循环为工件数 n，因此该部分总的时间复杂度为 $O(mn)$。模拟退火算法部分外循环由温度决定，内循环是同一温度下在一段时间内的探索，假设外循环的循环次数为 $logT$，内循环为 t，则该部分的时间复杂度为 $O(tlogT)$。

结合两部分，总的算法复杂度为 $O(mntlogT)$

2) 空间复杂度

动态规划部分经过优化，只需记录上一状态和当前状态，因此该部分的空间复杂度为 $O(n)$。由于整个算法的空间复杂度主要由动态规划部分决定，所以总空间复杂度为 $O(n)$。

3 实验

3.1 实验设置

3.1.1 实验环境

算法部分：c++ 语言环境，可在 Visual Studio Code 或 Dev-C++ 上直接运行。

绘图部分：Process on、Matlab 语言环境、Excel

3.1.2 参数设置和运行时间

1) 登山算法

算法中的唯一参数：迭代循环次数 = 100000，其对应 11 个用例的运行时间如表3-1所示

表 3-1 登山算法参数设置与运行时间
Tab 3-1 Parameter settings and runtime of Hill Climbing

用例	运行时间 (s)
0	0.006
1	0.097
2	0.029
3	1.096
4	0.215
5	0.220
6	0.201
7	0.539
8	0.005
9	0.080
10	0.916

2) 模拟退火算法

具体的参数设置和运行时间如表3-2所示

表 3-2 模拟退火算法参数设置与运行时间
Tab 3-2 Parameter settings and runtime of Simulated Annealing

用例	初始温度 T	退火率 dT	内循环次数 t	结束温度 eps	运行时间 (s)
0	10000	0.9	1000	1e-6	0.966
1	10000	0.9	1000	1e-6	1.005
2	10000	0.9	1000	1e-6	1.005
3	10000	0.9	1000	1e-6	0.577
4	10000	0.9	1000	1e-6	0.104
5	10000	0.9	1000	1e-6	2.222
6	10000	0.9	1000	1e-6	2.535
7	100000	0.97	5000	1e-6	3.491
8	10000	0.9	1000	1e-6	1.512
9	100000	0.97	5000	1e-6	4.032
10	100000	0.9	1000	1e-6	0.520

3.2 实验结果

3.2.1 实验最优结果

为了确保实验结果的准确性，通过调整算法的参数，在相同参数下跑 10 遍，取其中最优解作为结果。最终获得各测试用例的最优调度时间如下表3-3所示

表 3-3 各测试用例最优调度时间和调度方案
Tab 3-3 The optimal scheduling time and scheduling scheme for each case

用例	工件数	机器数	最佳调度时间	调度顺序
0	11	5	7038	[7 4 8 2 3 0 10 1 9 6 5]
1	8	8	8366	[6 2 7 4 1 0 5 3]
2	13	4	7166	[6 2 3 10 8 4 7 1 11 5 12 9 0]
3	12	5	7312	[10 11 5 4 9 8 2 1 3 6 7 0]
4	14	4	8003	[3 8 6 12 7 10 13 0 4 11 9 5 1 2]
5	10	6	7720	[3 4 1 0 2 7 5 9 8 6]
6	20	10	1431	[15 1 13 8 11 3 12 9 18 10 19 0 17 14 2 4 7 6 16 5]
7	20	15	1973	[5 14 8 4 19 13 18 12 0 10 1 11 16 7 2 17 15 9 6 3]
8	20	5	1109	[5 13 6 0 1 2 16 7 10 15 9 11 12 8 3 18 4 14 17 19]
9	20	15	1909	[12 17 19 16 18 7 8 3 11 10 2 14 13 1 9 6 15 0 5 4]
10	50	10	3277	[12 13 9 38 2 16 49 34 29 39 41 17 43 44 26 1 36 20 33 19 7 22 27 40 24 3 47 31 46 42 4 8 21 35 5 11 6 0 10 14 18 45 37 28 48 25 32 15 23 30]

其中以*用例 3*为例，对实验结果进行具体分析。登山算法和模拟退火算法所得到的最优调度时间、参数和运行时间具体如下。

登山算法对应的参数和结果如下表所示。

表 3-4 登山算法参数设置与运行时间
Tab 3-4 Parameter settings and runtime of HC

工件数	机器数	最佳调度时间	调度顺序	循环次数	运行时间 (s)
12	5	7543	[5 11 9 6 3 4 10 2 1 8 0 7]	100000	0.235

模拟退火算法对应的参数和结果如下表所示。

表 3-5 模拟退火算法参数设置与运行时间
Tab 3-5 Parameter settings and runtime of SA

最佳调度时间	调度顺序	初始温度	退火率	内循环次数	结束温度	运行时间 (s)
7312	[10 5 11 4 9 2 8 1 3 6 7 0]	10000	0.9	3000	1e-6	1.415

两种算法对应甘特图如下，其中同一种颜色表示同一个工件。

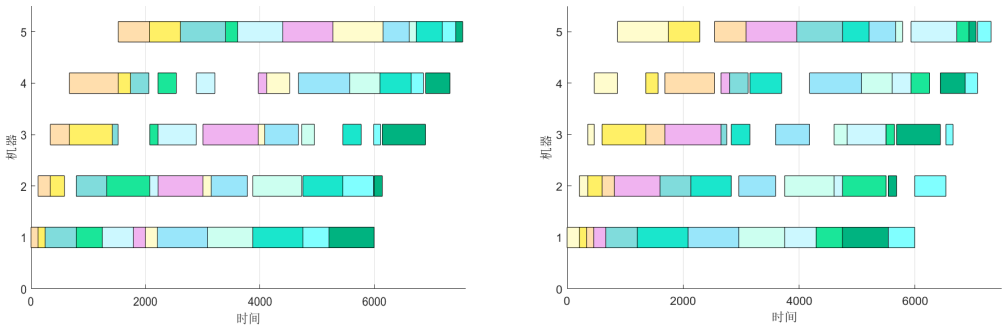

图 3-1 登山算法和模拟退火算法调度甘特图
Figure 3-1 The Gantt Chart of Hill Climbing and Simulated Annealing

两种算法当前调度时间随迭代次数优化过程，如下图所示。

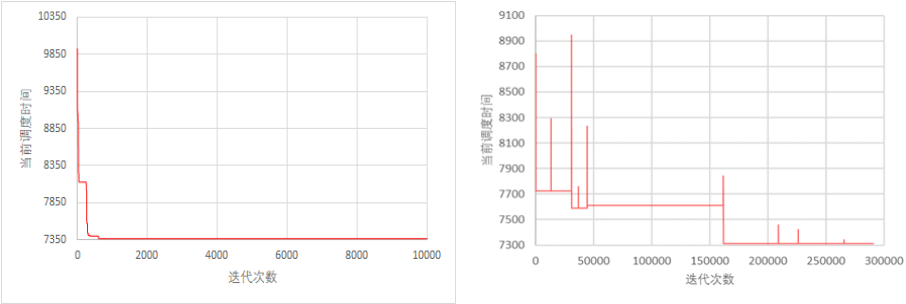

图 3-2 登山算法和模拟退火算法优化过程
Figure 3-2 The process of Hill Climbing and Simulated Annealing

由图可知，从迭代次数来看，登山算法收敛速度很快，基本在 1000 次左右迭代时就收敛到最终结果；相比之下，模拟退火算法迭代到 160000 次左右时才收敛，收敛速度相对慢很多。从当前调度时间变化来看，登山算法的调度时间保持不断更新下降，且最后深陷局部最优；模拟退火算法则在不断振荡的过程中保持下降趋势，直至趋于稳定值。其中振荡的幅度和频度随迭代次数增加而减少，这是因为模拟退火算法有一定概率探索更差的策略，而这个概率随着时间增加、温度下降而不断降低，这与模拟退火算法过程相符。

3.2.2 不同算法对比实验

为了保证实验结果的准确性，最优调度时间为每一用例在同一参数下跑 10 遍取最优，运行时间为该最优结果对应的程序运行时间。

表 3-6 登山算法和模拟退火算法实验结果对比
Tab 3-6 Hill climbing v.s. Simulated Annealing

用例	登山算法最优调度时间	运行时间	模拟退火最优调度时间	运行时间
0	7038	0.006	7038	0.966
1	8366	0.097	8366	1.005
2	7166	0.029	7166	1.005
3	7366	1.096	7312	0.577
4	8076	0.215	8003	0.104
5	7720	0.22	7720	2.222
6	1479	0.201	1431	2.535
7	2006	0.539	1973	3.491
8	1113	0.005	1109	1.512
9	1997	0.08	1909	4.032
10	3300	0.916	3277	0.52

由表可得，对于同一用例，模拟退火算法的最优调度时间结果比登山算法更短，但其运行时间相对更长。

表 3-7 登山算法与模拟退火算法算法思想对比
Tab 3-7 Hill climbing v.s. Simulated Annealing

	登山算法	模拟退火算法
探索 (Exploration)	×	以一定概率 downhill
深挖 (Exploitation)	探索到更优解更新	探索到更优解更新

事实上，在搜索最优解方面，模拟退火算法比登山算法更优。这是由于模拟退火算法能够以一种与时间和温度相关的概率尝试较糟糕的解，而这能在一定程度上摆脱局部最优问题的约束，扩展解的搜索范围，从而找出全局最优。而在运行时间方面，登山算法收敛速度更快，运行花费的时间比模拟退火算法少得多。这是因为登山算法只有在搜索到更优解时才更新，因此很快达到其初始点

附近的最优解；而模拟退火算法则会以一定概率探索可行解，即使搜索到当前最优解后，也可能尝试较差解而跳出局部收敛，所以收敛速度较慢，相应的运行时间较长。

3.2.3 多参数对比实验

登山算法: 改变迭代次数，其他参数不变

表 3-8 仅改变迭代次数登山算法的运行结果与运行时间
Tab 3-8 The results and running time of Hill Climbing Algorithm change with the number of iterations

迭代次数	用例 3 最优调度时间	用例 3 运行时间 (s)	用例 5 最优调度时间	用例 5 运行时间 (s)	用例 10 最优调度时间	用例 10 运行时间 (s)
100	7882	0.001	7843	0.001	3593	0.005
1000	7527	0.003	7821	0.003	3370	0.1
10000	7725	0.023	7843	0.023	3383	0.014
50000	8126	0.111	7832	0.108	3371	0.486
100000	7543	0.235	7720	0.22	3300	0.916
500000	7366	1.096	7738	1.072	3316	4.559
1000000	7531	2.117	7825	1.98	3439	9.089
5000000	7611	10.32	7824	10.028	3325	46.548

由表可知，从运行时间来看，随着迭代次数和用例规模的增大，运行时间越长；从最优调度时间来看，规律性较少，这是由于登山算法的结果受“出生点”的影响，即受到初始化时所在位置的约束性大，否则容易陷入局部最优而无法找到全局最优的结果。

模拟退火算法:

1) 改变初始温度 T_0，其他参数不变

表 3-9 仅改变初始温度 T_0 的运行结果与运行时间
Tab 3-9 The results and running time of Simulated Annealing Algorithm change with the initial temperature

	初始温度 T	用例 3 调度时间	用例 3 运行时间 (s)	用例 4 调度时间	用例 4 运行时间 (s)	用例 10 调度时间	用例 10 运行时间 (s)
其他参数	1000	7443	0.517	7843	0.001	3593	0.005
	5000	7400	0.554	7821	0.003	3370	0.1
温度下限:	10000	7312	0.577	7843	0.023	3383	0.014
$EPS = 1e-4$	50000	7400	0.618	7832	0.108	3371	0.486
衰减系数:	100000	7531	0.644	7720	0.22	3300	0.916
$dT = 0.8$	500000	7543	0.685	7738	1.072	3316	4.559
内循环上限:	1000000	7328	0.707	7825	1.98	3439	9.089
$t = 500$	5000000	7400	0.756	7824	10.028	3325	46.548

由表可知，在其他参数不变的情况下，初始温度 T 过低或过高时可能找不到最优解，其中过高的初始温度 T 可能导致运行时间指数级增加；同时，不同的用例规模也影响其最优解所对应的适宜温度。

究其根本，初始温度 T 设置过低时，downhill 的概率过低，导致陷入局部最优解中；而初始温度 T 设置过高时，可能 downhill 的概率过高，而导致算法过于随机而丧失了收敛到最优解的特性，并且还会导致运行时间增加。由此可见，初始温度主要影响跳出局部最优解的能力和解的搜索范围。因此，初始温度需要先尝试多个数量级，从而选出最适合该用例的参数大小。

2) 改变温度衰减系数 dT，其他参数不变

表 3-10 仅改变衰减系数 dT 的运行结果与运行时间
Tab 3-10 The results and running time of Simulated Annealing Algorithm change with the attenuation coefficient

	衰减系数 dT	用例 3 调度时间	用例 3 运行时间 (s)	用例 5 调度时间	用例 5 运行时间 (s)	用例 10 调度时间	用例 10 运行时间 (s)
温度下限:	0.7	7590	0.160	7768	0.157	3333	0.364
$EPS = 1e-6$	0.8	7480	0.247	8599	0.247	3353	0.532
初始温度:	0.9	7531	0.510	7821	0.505	3389	1.061
$T = 10000$	0.95	7312	1.028	7835	0.998	3280	2.083
内循环上限:	0.99	7312	5.161	7720	4.948	3277	21.368
$t = 1000$	0.999	7312	52.229	7750	51.425	3277	207.469

由表可知，在其他参数不变的情况下，当衰减系数 dT 较小时，得到的解比较糟糕；随着衰减系数 dT 不断增大，虽然运行时间增加，但所得到的最终解更加优秀；而衰减系数 dT 过大，相比于适宜的衰减系数，其运行时间过长。

究其根本，衰减系数主要影响迭代次数和退火温度的变化幅度：dT 越小，温度下降越快，算法收敛速度过快且缩小了解的搜索范围；dT 越大，温度下降速度越慢，则算法解的搜索范围越大，越有可能搜索到最优解；然而 dT 过大，运行时间指数级增加。因此，不同用例的规模输入，选取适合的衰减系数。

3) 改变温度下限 EPS，其他参数不变

表 3-11 仅改变温度下限 EPS 的运行结果与运行时间
Tab 3-11 The results and running time of Simulated Annealing Algorithm change with the lower temperature

	温度下限 EPS	用例 3 调度时间	用例 3 运行时间 (s)	用例 5 调度时间	用例 5 运行时间 (s)	用例 10 调度时间	用例 10 运行时间 (s)
温度衰减系数:	1e-2	7366	3.103	7767	3.063	3284	12.751
$dT = 0.99$	1e-3	7366	3.512	7738	3.532	3280	14.960
初始温度:	1e-4	7399	4.091	7720	4.191	3284	16.922
$T = 10000$	1e-5	7399	4.504	7749	4.74	3284	19.069
内循环上限:	1e-6	7312	5.079	7720	5.212	3280	21.363
$t = 1000$	1e-7	7312	602	7720	5.65	3277	23.446

温度下限 EPS 决定退火达到稳定状态的温度，即结束迭代的条件约束。由表可知，当温度下限 EPS 较大时，算法还未搜索到最优解即达到稳定状态；随着温度下限 EPS 减小，搜索到最优解的可能性增加，且运行时间也不断增加。因此，选择适合的温度下限 EPS 既能获得最优解，又能尽量减少运行时间。

4) 改变退火内循环上限 t，其他参数不变

表 3-12 仅改变退火内循环上限 t 的运行结果与运行时间
Tab 3-12 The results and running time of Simulated Annealing Algorithm change with the upper limit of internal circle

	内循环上限 t	用例 3 调度时间	用例 3 运行时间 (s)	用例 5 调度时间	用例 5 运行时间 (s)	用例 10 调度时间	用例 10 运行时间 (s)
温度衰减系数: $dT = 0.99$	100	7443	0.509	7749	0.484	3280	2.027
初始温度: $T = 10000$	500	7372	2.332	7720	2.222	3285	9.769
温度下限: $EPS = 1e-5$	1000	7312	4.594	7738	4.477	3277	19.452

由表可知，退火内循环上限 t 越大，则获得的调度结果越优，而运行时间则更长。退火内循环上限决定在同一温度下算法内部的迭代次数，内循环迭代次数越多，在温度较高时解的搜索范围越

大，从而更有可能搜索到更优解。因此，选择适合的退火内循环上限大小，能够增加搜索到最优解的概率。

4 总结

4.1 总结工作

本文使用登山算法和模拟退火算法来解决流水车间调度问题，并比较两个算法在该问题中的优缺点。本文首先从流水车间调度问题出发，阐述了问题的研究背景；接着，设计了分别基于登山算法和模拟退火算法解决调度问题的具体算法，同时依据动态规划算法计算方案的调度时间；最后，通过两种算法的结果对比和多参数对比实验，分析出两种算法各自的特点和优缺点，以及算法的不同参数对调度结果的影响，并阐述导致规律性结果的原因。

4.2 未来展望

4.2.1 基于登山算法的调度问题

不足：

登山算法求得的最优解受到其初始位置的影响较大，这导致求得最优解的概率较为随机而不够稳定。

解决方案：

1) 随机重启：多次随机选取新的初始位置，重新开始登山搜索。

2) 局部束搜索：并行运行多个随机初始点，同时开始登山搜索。

4.2.2 基于模拟退火算法的调度问题

不足：

随着问题规模增大，模拟退火算法的运行时间过长。

解决方案：

1) 通过改变参数，调整到适宜该问题规模的大小。

2) 增加提早停止条件，即当调度最优时间在较大的循环次数下都未被更新，则说明算法可能已经收敛到最优解，可以提前结束运行。

3) 将模拟退火算法和遗传算法相结合，优化随机解的编码变化规则。

参考文献

[1] 赵诗奎, 王林瑞, 石飞. 作业车间调度问题综述 [J]. 济南大学学报：自然科学版, 2016, 30(1):7.

[2] Johnson S M . Optimal two- and three-stage production schedules with setup times included[J]. Naval Research Logistics Quarterly, 1954, 1.

[3] 越民义, 韩继业. n 个零件在 m 台机床上的加工顺序问题 (I)[J]. 中国科学, 1975, 5(5):462.