求解车间调度问题的登山和模拟退火算法

黄昱欣 学号:1120200807

摘　要:　车间调度问题（Job Shop Scheduling, JSP）应用十分广泛，但由于解空间大，是一类 NP 完全问题，难以短时间得到最优解。登山算法和模拟退火(SA)算法是两种较为成熟的启发式算法，本文采用这两种算法对 JSP 问题进行求解。得到 10 个样例的最优生产时间为：7038 8366 7166 7312 8003 7720 1431 1950 1109 1902 3277。最后对两种算法的性能、改进策略进行了探讨。

关键词:　登山算法；模拟退火算法；车间调度；

1　引言

1.1　问题背景介绍

调度是在一定的约束条件下，把有限的资源在时间上分配给若干个任务，以满足或优化一个或多个性能指标。车间生产过程中的调度问题是制造系统运筹技术、管理技术与优化技术发展的核心[1],有效的调度方法与优化技术的研究和应用成为制造技术实践的基础和关键，所以对它的研究有重要的理论和实用价值。

1.2　数学语言描述

考虑一个由机器处理作业的车间，需要在 m 台机器$M_1,M_2,\cdots M_m$上完成 n 项作业$J_1,J_2,\cdots J_n$，每个作业J_i都由一定数量p_i的操作$O_{i1},O_{i2},\cdots O_{ipi}$组成。操作顺序固定，并且任意两个作业之间的操作顺序是独立的，没有关联。每项操作O_{ij}都需要在一台专门的机器上完成，有一个固定的加工时间t_{ij}，同一时间一台机器上只能有一项操作在进行。目标在于为所有机器找到处理工件的序列，从而最小化作业完成时间。

1.3　解决方案以及实验结果

登山算法是一种简单的贪心搜索算法，通过每轮搜索邻域，模拟登山过程，选择邻域的最优解作为当前解，最终迭代到最优解，完成求解。

模拟退火算法也是一种贪心算法，但是模拟退火算法引入了随机因素。模拟退火算法学习冶金学中金属加热冷却的过程，以搜寻空间内一个任意点作起始：每一步先选择一个“邻居”，然后再计算从现有位置到达“邻居”的概率。若概率大于给定的阈值，则跳转到“邻居”；若概率较小，则停留在原位置不动。每次随机出下一个解，如果由于当前解，则解更新为下一个解，否则以一定概率更新为更劣解，从而有跳出局部最优解的可能，以此搜索全局最优。

本文采用这两种方法解决车间调度问题。结果表明，模拟退火算法在本问题上的表现优于登山算法，但是登山算法所使用的时间更短。

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

在本问题中，由于 n 项作业的操作顺序相同，都从第一台机器加工到最后一台机器，因此，在各台机器上，n 个工件加工的顺序也是一样的，所以我们采用顺序编码，用一个 n 维的向量$(x_1,x_2,\cdots,x_n)$来代表每台机器上工件加工的先后顺序，解空间中共有 $n!$个解。对于解空间中的任意一个解，它的邻域是有交换向量中任意两个位置上的数字得到的向量组成的，邻域中共有$C_n^2=n(n-1)/2$ 个解。

2.1 登山算法Hill-Climbing

2.1.1 算法思路简介

登山算法借鉴登山过程，将一个解看做解空间中的一个点，山的高度则是目标函数的大小，对每次迭代，都从当前解的邻域中找到目标函数最高的解，存在则更新。

2.1.2 算法关键操作

登山算法有如下步骤：

1）在解空间随机选择一点作为初始解

2）在当前解的邻域中找到使目标函数值最高的解

3）如果该解的目标函数值高于当前解，则将当前解更新为该解，回到步骤 2

4）如果小于等于当前解，则结束迭代，结果为当前解的目标函数值

2.1.3 算法流程图

算法设计流程如图所示：

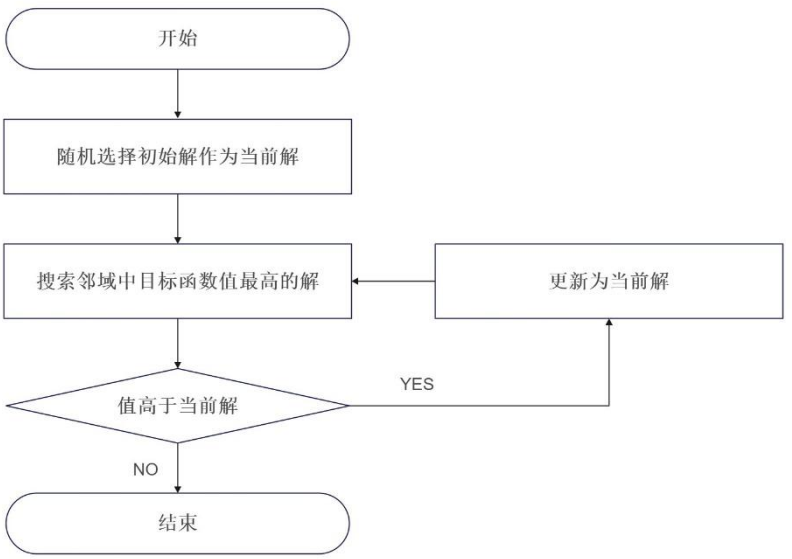

图 1 登山算法流程图

2.1.4 算法核心部分的时间及空间复杂度分析

登山算法的核心部分是搜索邻域中目标函数值最高的解，假设算法迭代 MaxIter 次后不再能找到目标函数值更高的解，工件数为 N，机器数量为 M。计算给定的加工序列需要的时间复杂度为 $O(M*N)$，空间复杂度为 $O(M)$。每次迭代中，需要搜索领域中 $N(N-1)/2$ 个加工序列，并调用函数计算他们的加工时间，所以每次迭代的时间复杂度为 $O(M*N^3)$。每次迭代中需要两个大小为 N 的数组，存储当前加工序列和邻域中的一个解，由于计算加工时间的函数中使用的数组是全局数组，所以空间复杂度为 $O(M+2N)$。

因此算法核心部分的时间复杂度为 $O(\text{MaxIter}*M*N^3)$，空间复杂度为 $O(M+2N)$。

2.2 模拟退火算法Simulated Annealing

2.2.1 算法思路简介

模拟退火算法的核心在当随机得到下一个解时，通过 Metropolis 法则以概率 P 接受更劣解：

$$P=\begin{cases}1 & , \quad VALUE[next]<VALUE[current] \\ e^{-\frac{VALUE[next]-VALUE[current]}{T}} & , \quad VALUE[next]\geq VALUE[current]\end{cases}$$

其中，$VALUE[next]$为下一个解的目标函数值，$VALUE[current]$为当前解的目标函数值，T 为模拟退火的温度值。可以看出，温度越高、更劣解的程度越低，接受该更劣解的可能性越大。这样的更新策略使得模拟退火算法有可能跳出当前的局部最优解，达到全局最优解。

2.2.2 算法关键操作

模拟退火算法有如下步骤：

1）初始化初始温度、学习率，随机选择初始解作为当前解，并计算目标函数值 $VALUE[current]$

2）如果 T 低于设定值，则退出循环

3）随机在邻域中选择下一个解，计算目标函数值 $VALUE[next]$

4）如果 $VALUE[current]$大于 $VALUE[next]$，则将当前解更新为临域中的解，以公式 1 更新温度，返回步骤 2

5）如果小于等于，则以公式 2 计算出概率，以该概率更新当前解，并更新温度，返回步骤 2

2.2.3 算法流程图

算法设计流程如图所示：

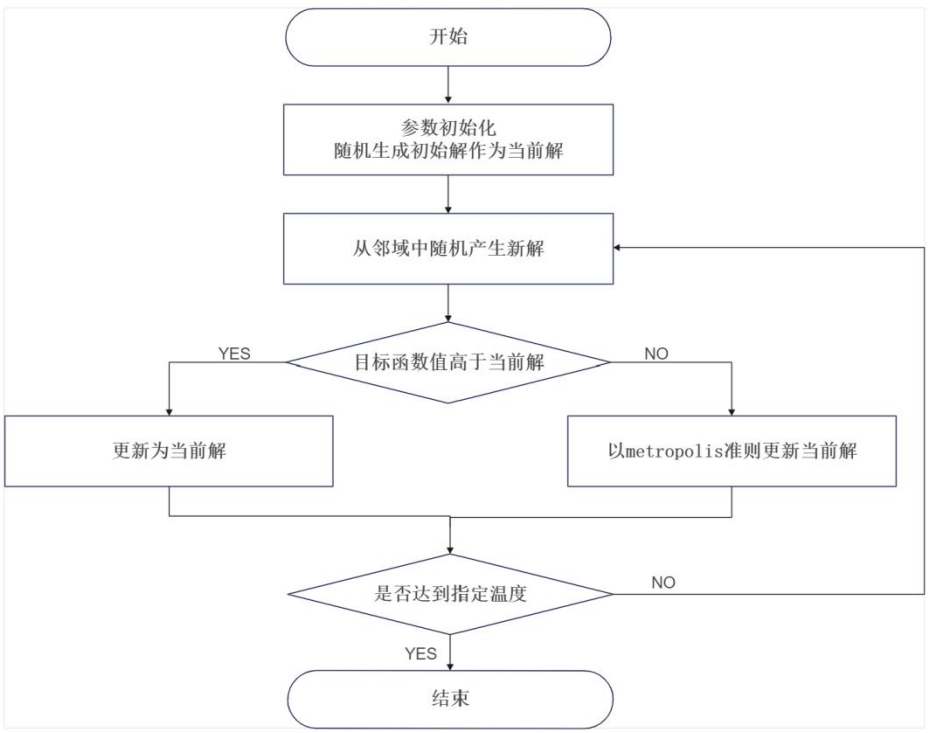

图 2 SA 算法流程图

2.2.4 算法核心部分的时间及空间复杂度分析

模拟退火算法的核心部分包括外部降温循环和每次更新解两部分，假设降 MaxIter 次后达到指定温度，工件数为 N，机器数量为 M。计算给定的加工序列需要的时间复杂度为 $O(M*N)$，空间复杂度为 $O(M)$。每次迭代中，随机选择邻域中的一个解，并计算这个解的加工时间，根据 Metropolis 法更新当前解，所以每次迭代的时间复杂度为 $O(M*N)$。每次迭代中需要两个大小为 N 的数组，存储当前加工序列和邻域中的解，由于计算加工时间的函数中使用的数组是全局数组，所以空间复杂度为 $O(M+2N)$。

因此算法核心部分的时间复杂度为 $O(\text{MaxIter} * M * N)$，空间复杂度为 $O(M + 2N)$。

3 实验

3.1 实验设置

3.1.1 实验环境

1）处理器类型

AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz

2）RAM

16.00GB

3）操作系统

64 位 Windows 10 家庭中文版

4）编程环境

Visual Studio 2019

5）编程语言

C++

3.1.2 运行时间

登山算法 10 个用例的总运行时间为 0.091s。

算法时间为多次运行的平均时间。

样例号	时间（s）
0	0.001
1	0.001
2	0.001
3	0.001
4	0.001
5	0.001
6	0.009
7	0.011
8	0.002
9	0.012
10	0.052
ALL	0.091

表 1 登山算法运行时间

2）SA 算法

算法时间选用较优的一组参数的运行时间。

样例号	时间（s）
0	13.74
1	2.184
2	6.902
3	2.532
4	7.444
5	3.308

6	2.371
7	2.304
8	4.256
9	2.046
10	7.207
ALL	54.294

表 2 SA **算法运行时间**

3.1.3 SA 算法参数

1）初始温度

模拟退火的初始温度T_0的初始值设置是影响模拟退火算法全局搜索性能的重要因素之一、初始温度高，则搜索到全局最优解的可能性大，但因此要花费大量的计算时间；反之，则可节约计算时间，但全局搜索性能可能受到影响。[2]

2）学习率α

在模拟退火中，每一轮迭代过后需要对温度进行更新，更新公式为 $T(t+1)=\alpha * T(t)$，其中α为略小于 1 的常数。学习率α的大小会影响温度降低的快慢，也会决定迭代次数，进而影响算法的运行时间。

3）终止温度T_f

当 $T < T_f$时，迭代终止。只有T_f充分小时，才有可能得出高质最终解。[2]

3.2 实验结果

1）最优结果及甘特图

各个样例的最优结果及甘特图如下：

样例号	最优结果	最优结果对应加工顺序
0	7038	7 4 2 10 6 1 8 9 5 3 0
1	8366	6 2 7 4 1 0 5 3
2	7166	6 2 3 10 1 11 7 12 5 8 4 0 9
3	7312	10 5 11 4 9 2 8 1 3 6 7 0
4	8003	3 13 11 5 12 8 0 6 9 4 7 10 2 1
5	7720	4 1 3 0 2 7 5 9 8 6
6	1431	15 1 13 11 8 3 9 18 10 19 12 14 7 4 2 0 17 16 6 5
7	1950	4 12 11 0 8 14 5 18 13 16 19 7 2 15 1 9 10 17 3 6
8	1109	5 13 6 0 1 7 16 3 12 2 10 9 18 4 15 14 17 11 8 19
9	1902	12 17 1 16 7 8 18 10 2 14 19 3 11 13 9 6 15 0 5 4
10	3277	12 13 24 38 34 49 29 18 2 36 44 1 4 39 41 22 47 26 9 35 17 10 16 20 5 21 19 45 32 7 43 14 8 33 15 25 3 11 40 42 46 37 0 6 31 28 48 27 23 30

表 3 **样例最优结果**

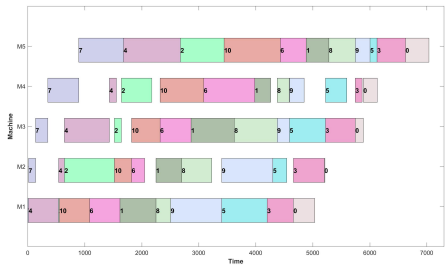

图 3 样例 0 最优结果甘特图

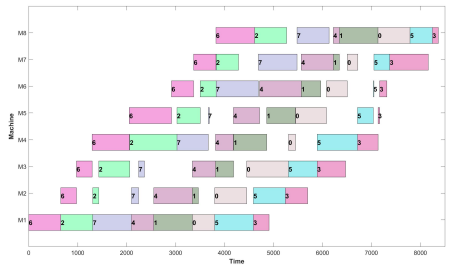

图 4 样例 1 最优结果甘特图

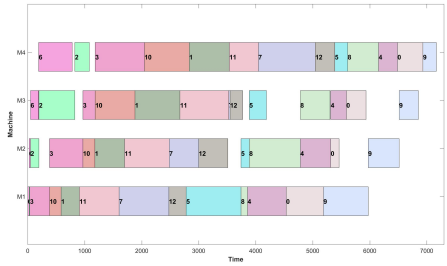

图 5 样例 2 最优结果甘特图

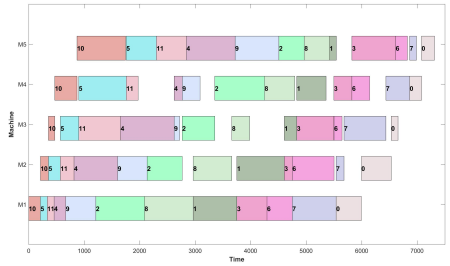

图 6 样例 3 最优结果甘特图

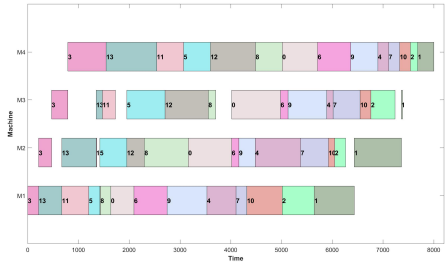

图 7 样例 4 最优结果甘特图

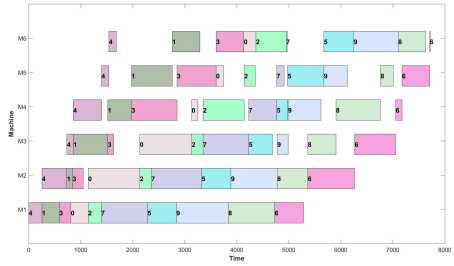

图 8 样例 5 最优结果甘特图

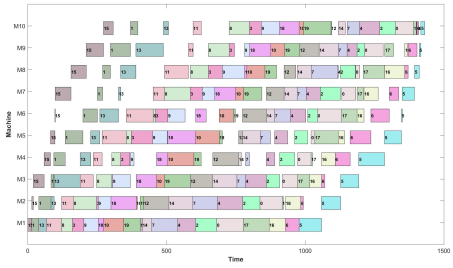

图 9 样例 6 最优结果甘特图

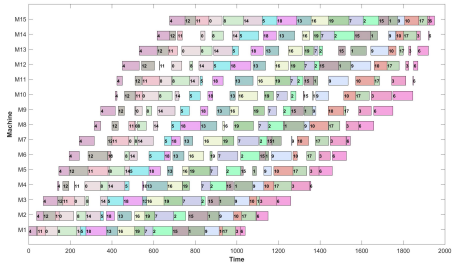

图 10 样例 7 最优结果甘特图

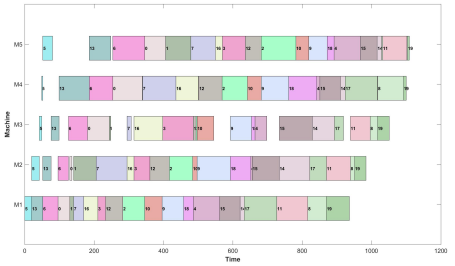

图 11 样例 8 最优结果甘特图

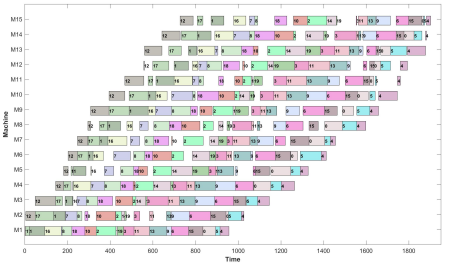

图 12 样例 9 最优结果甘特图

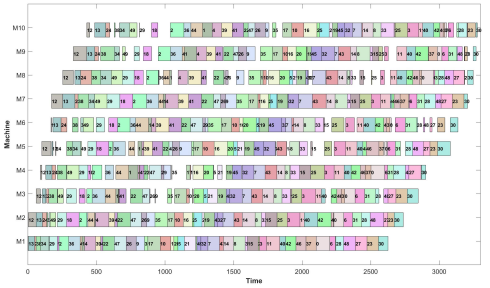

图 13 样例 10 最优结果甘特图

2）登山、SA 及其优化算法对比

样例号	登山算法	时间(s)	改进登山算法	时间(s)	SA算法	时间(s)	改进SA算法	时间(s)
0	7168.2	0.0204	7038	0.001	7038	13.74		
1	8526.2	0.016	8564	0.001	8366	2.184		
2	7640.2	0.0228	7376	0.001	7166	6.902		
3	7491	0.0204	7399	0.001	7312	2.532		
4	8147.8	0.0206	8129	0.001	8003	7.444		
5	7809	0.0172	7832	0.001	7720	3.308		
6	1527	0.054	1541	0.009	1431	2.371		
7	2025.8	0.0492	2008	0.011	1973	2.304	1950	512.117
8	1149.2	0.1604	1114	0.002	1109	4.256		
9	2010	0.056	2007	0.012	1950	2.046	1902	363.268
10	3358.6	0.1484	3295	0.052	3277	7.207		

表 4 登山与 SA 算法及改进算法得到的结果

3.2.1 SA 算法参数实验

由于T_f充分小时，才能得到较好的解，所以在参数试验中，始终固定$T_f = 10^{-30}$。

1） 初始温度

选取学习率为$\alpha = 0.999$，终止温度$T_f = 10^{-30}$。

样例号 \ 初始温度	1	10	100	1000	10000
0	7038	7038	7038	7038	7038
1	8424	8530	8366	8366	8366
2	7166	7166	7166	7166	7166
3	7312	7312	7312	7312	7312
4	8003	8003	8003	8003	8003
5	7738	7720	7720	7720	7720
6	1452	1462	1456	1431	1440
7	1977	1965	1975	1973	1980
8	1111	1111	1111	1109	1109
9	1961	1938	1952	1950	1935
10	3277	3277	3277	3277	3277

表 5 初始温度与加工总时间的关系

对于用例 0、2、3、4、10，初始温度的提高没有对加工时间有减小的作用，因此认为对于这几个用例，已经达到了最优解。而对于其于用例，随着初始温度的提高，加工总时间有所减少，说明高初始温度有利于得到更优解。

样例号 \ 初始温度	1	10	100	1000	10000
0	7.548	10.339	12.619	13.74	13.688
1	2.365	2.365	2.365	2.365	2.365
2	5.569	7.026	6.25	6.902	7.652
3	1.677	2.053	2.177	2.532	3.621
4	6.062	7.596	6.985	7.444	8.405
5	1.641	3.106	2.498	3.308	3.604
6	0.661	1.084	1.255	2.371	3.295
7	0.582	0.968	1.675	2.304	3.313
8	2.33	3.31	3.511	4.256	5.559
9	0.516	0.634	1.327	2.046	2.938
10	5.266	5.888	6.438	7.207	8.317

表 6 初始温度与程序运行时间的关系

由数据可以看出，对于所有用例，随着初始温度的升高，迭代次数增加，运行时间有所增长，但是增长幅度最大为 5s，增长程度并不显著。总体来看， 1000 是一个较优的初始温度，既可以保证程序运行时间段，又可以保证得到的最优加工时间较短。

2）学习率

选取初始温度为$T_0 = 1000$，终止温度$T_f = 10^{-30}$。

样例号 \ 学习率	0.9	0.99	0.999	0.9999
0	7038	7038	7038	7038
1	8896	8366	8366	8366
2	7166	7166	7166	7166
3	7400	7312	7312	7312
4	8129	8003	8003	8003
5	7966	7767	7720	7720
6	1520	1469	1431	1431
7	2044	1983	1973	1961
8	1151	1110	1109	1109
9	2041	1976	1950	1933
10	3362	3284	3277	3277

表 7 学习率与加工总时间的关系

对于用例 0、2，学习率增大没有减少加工总时间，因此认为较小的学习率也能达到最优解。对于其他用例，更高的学习率可以带来更小的加工总时间。

样例号 \ 学习率	0.9	0.99	0.999	0.9999
0	0.332	2.483	13.74	86.992
1	0.135	0.208	2.184	18.296
2	0.255	1.679	6.902	69.405
3	0.186	0.302	2.532	25.889
4	0.444	0.742	7.444	73.399
5	0.037	0.352	3.308	27.452
6	0.089	0.315	2.371	22.806
7	0.159	0.327	2.304	24.31
8	0.176	0.423	4.256	42.284
9	0.088	0.199	2.046	20.498
10	0.307	2.501	7.207	72.919

表 8 学习率与程序运行时间的关系

与探究最优初始温度时不同，学习率增大带来的程序运行时间涨幅是指数级的，最大涨幅达到了 86s，由于学习率 0.999 和 0.9999 的加工总时间只在用例 7、9 有所增加，因此总体衡量，认为选取 0.999 的学习率更合适。

综上，SA 算法选取初始温度为$T_0 = 1000$，终止温度$T_f = 10^{-30}$，学习率$\alpha = 0.999$ 较为合适。

3.2.2 实验结果分析

从表中可以看出，与登山算法相比，SA 算法得到的加工时间更短，但是相应的，程序与运行时间也更长，这是因为 SA 算法的迭代次数远高于登山算法，并且 SA 算法具有“回头”的机制，能接受更劣解，因此可以得到更优的加工序列。

1） 登山算法

NEH 启发式算法是由 Nawaz,Enscore,Ham 提出的启发式算法[3]，能在较短时间得到一个较优解，其算法步骤为:

步骤 1：计算所有工件的总加工时间，并按非增顺序排列各工件，得到初始排序。

步骤 2：取出该排序的前两个工件，将其排序得到两个调度，将最大完成时间较小的一个作为当前调度。

步骤 3：依次取出排序中的工件，将其插入到已有调度的可能位置，选取最大完成时间最小的调度作为当前调度，直到左右工件都被加入调度，得到结果。

由于原始的登山算法随机因素较大，因此加入 NEH 优化，可以保证算法得到一个较优的初始解，保证算法的稳定性。

2） SA 算法

在 SA 算法的参数实验中可以看出，除了用例 7 和用例 9，其他用例都出现了不同参数得到的最优加工时间相同的情况，因此认为算法在用例 7 和用例 9 上的表现还可以继续优化。所以在本文在原来模拟退火算法的基础上加入了外部循环，参数选用初始温度为$T_0 = 1000$，终止温度$T_f = 10^{-30}$，学习率$\alpha = 0.999$，共循环 200 轮。第一轮 SA 算法的初始解为随机解，此后每轮 SA 算法都以全局最优解作为初始解，运行改进的模拟退火算法得到了用例 7 和用例 9 的更优解。

4 总结

车间调度问题是一类 NP 完全问题，登山算法和模拟退火算法是两种较为成熟的启发式算法，本文采用这两种算法对 JSP 问题进行求解。针对 SA 算法进行了参数实验，同时针对这两种算法在实际用例上运行时出现的不足之处，提出了相应的解决策略，登山算法使用初始化优化，SA 算法使用多次迭代优化。

4.1 登山算法的优点

1）登山算法实现简单。

2）登山算法能较快地得到一个较优解。

4.2 登山算法的缺点

1）登山算法容易陷入局部最优解。

2）登山算法得到的结果依赖于初始解。

4.3 SA算法的优点

1）SA 算法具有接受更劣解的能力，因此搜索全局最优解的能力较好。

2）如果参数选取得当，能在可接受的时间内搜索到全局最优解。

4.4 SA算法的缺点

1）耗时长于登山算法。

2）因为本文的 SA 算法的初始温度、学习率和终止温度都是提前规定好的，因此对所有样例来说，迭代次数都相同。由于有些样例迭代次数较小时也能搜索到全局最优解，而有的样例则需要较高的迭代次数才能搜索到全局最优，算法不会根据实际情况对参数进行设置，没有适应能力。

4.5 改进策略

根据登山算法的特性，选用 NEH 优化，可以使登山算法拥有稳定的较优解。

根据 SA 算法的温度下降策略可以看出，当温度降到充分小时，跳出“陷阱”的概率就越来越小，因此，可以在降温的基础上增加回温过程，使算法的接受率再次提高，从而有望跳出局部最优解，搜索到全局最优解。[2]

参考文献:

[1] 何霆,马玉林,杨海. 车间生产调度问题研究[J]. 机械工程学报,2000,36(5):97. DOI:10.3321/j.issn:0577-6686.2000.05.025.

[2] 朱婧. 求解车间调度问题的改进模拟退火算法[C]. //第五届中国管理学年会(MAM2010)论文集. 2010:1833-1852.

[3] Nawaz M, Enscore E, Ham I. A heuristic algorithm for the m machine, n job flow shop[J]. The International Journal of Management Sciences, 1983,11(1):91-95.