基于模拟退火算法解决流水车间调度问题

姓名：周一鸣 (学号: 1120192000)

摘　要: 流水车间调度问题是实际流水线生产调度的简化模型，可以描述为： n 个工件需要依次在 m 台不同的机器上加工，求出最优调度方案使得加工总时间最短。本文主要使用模拟退火算法，其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性。该算法在求取最优解的时候有一定概率放弃局部最优解，且随着温度下降概率逐渐降低，从而降低求取到局部最优解的可能性。本实验能在较短时间内求出相对较优的调度方案与对应时间，在效率与性能间找到了平衡点，比传统解法更有优势。

关键词: 模拟退火算法，流水车间调度

1 引言

流水车间调度(Flow-shop Scheduling Problem, FSP)是许多实际流水线生产调度的一个简化模型，是研究较为广泛的生产调度问题之一，许多生产制造、装配、运输等问题均可简化为该模型。

流水车间调度问题可以用数学语言如下描述：

1、n、m 分别表示工件数与机器数

2、$T_{i,j}$表示工件 i 在机器 j 上的加工时间(0≤i≤n-1, 0≤j≤m-1)

3、p_i表示第 i 个工件的加工序号(0≤i, p[i]≤n-1)

4、$ans_{i,j}$表示在 p 序列调度下第 i 个工件完成第 j 个工序所花时间

则流水车间调度问题可以转化为求解$\min(ans_{n-1,m-1}|p,T)$，有约束条件：

$$\begin{cases} ans_{i,j}=\begin{cases} T_{p_i,j}|i=j=0, \\ ans_{i,j-1}+T_{p_i,j}|i=0, \\ ans_{i-1,j}+T_{p_i,j}|j=0, \\ T_{p_i,j}+\max\left(ans_{i-1,j},ans_{i,j-1}\right)|i\neq 0,j\neq 0 \end{cases}, \\ 0\leq i\leq \mathrm{n}-1, \\ 0\leq \mathrm{j}\leq \mathrm{m}-1, \\ p=\{p_0,p_1,\dots,p_{n-1}\}, \\ p_0\neq p_1\neq\cdots\neq p_{n-1}, \\ 0\leq p_0,p_1,\dots,p_{n-1}\leq \mathrm{n}-1 \end{cases}$$

由于该问题属于 NP-Hard 问题，用传统解法需要花费很长时间，因此本实验采用了智能优化算法中的模拟退火算法求解。先使用蒙特卡洛算法生成随机初始解，并在此基础上使用模拟退火算法，其优势在于前期可以有较高的概率选择不如当前解的方向前进，从而增加求得全局最优解的概率；后期逐渐向最优解的方向收敛，可以使得结果更接近全局最优解。

2 算法设计

模拟退火算法来源于固体退火原理，将固体加温至充分高，再让其徐徐冷却，加温时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。根据 Metropolis 准则，粒子在温度 T 时趋于平衡的概率为$\exp\left(-\frac{\Delta E}{kT}\right)$，其中$E$为物体在温度$T$时的内能，$\Delta E$为其变量，$k$为 Boltzmann 常数。用固体退火模拟组合优化问题，将内能 E 模拟为目标函数值 f，温度 T 演化成控制参数 t，即得到解组合优化问题的模拟退火算法：由初始解s_0和控制参数初值t开始，对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步衰减 t 值，算法终止时的解即为所得

近似最优解，这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。退火过程由冷却进度表(Cooling Schedule)控制，包括控制参数的初值t及其衰减因子Δt、每个t值时的迭代次数L和停止条件 S。

伪代码：

Begin

1、根据工件序号升序初始化加工顺序

2、蒙特卡洛算法初始化加工顺序MAX_T次。

3、初始化温度T，温度下限$T1$，每个T的迭代次数L，降温系数dt

4、while($T \geq T1$)

5、$for\ i := 1,2,\dots,L$ then 随机调换 x 中两个工件的加工次序产生新解p'

7、计算增量$\Delta f := f(p') - f(p)$，$f(p)$表示序列p对应的加工时间

8、若$\Delta f < 0$则接受p'为新解，否则以概率$exp(-\Delta f/T)$接受p'为新解。更新 t_best 和 sequence.

9、$T := T * dt$

10、输出p与 t_best

End

时间复杂度分析：

蒙特卡洛算法复杂度为$O(n)$，模拟退火算法复杂度为$O(1)$，本算法结合两者，整体的时间复杂度为$O(n)$。

空间复杂度分析：

计算 ans 数组需要至少 n*m 的空间，因此空间复杂度为$O(n*m)$。由于更新 ans 数组时只与其左侧和上方数值相关，因此可以将二维数组继续优化为一维以降低空间复杂度，可以达到$O(m)$。

3 实验

3.1 实验设置

实验环境：Visual Studio Code

运行时间：5min 以内

实验参数：MAX_T = 8000

用例序号	T0	T1	dt	L
0	4000.0	0.1	0.999	50
1	4000.0	0.1	0.999	50
2	2000.0	0.01	0.999	50
3	2000.0	0.01	0.999	50
4	2000.0	0.01	0.999	50
5	2000.0	0.01	0.999	50
6	2000.0	0.01	0.999	50
7	1000.0	0.001	0.999	50
8	1000.0	0.001	0.999	50
9	1000.0	0.001	0.999	50
10	1000.0	0.001	0.999	50

3.2 实验结果

每次实验读入包括初始用例的 11 组数据，得到每组数据对应的调度方案与时间如下：

用例序号	调度顺序	运行结果	运行时间（秒）
0	7,2,4,1,0,10,8,6,3,5,9	7038	1.206
1	6,2,7,4,1,0,5,3	8366	1.081
2	6,2,3,12,7,0,10,8,1,11,9,5,4	7166	1.271
3	10,5,4,11,9,8,2,1,3,6,7,0	7312	1.275
4	3,13,6,12,2,8,11,7,0,10,5,1,9,4	8003	1.366
5	4,1,3,0,2,7,5,9,8,6	7720	1.264
6	13,1,8,3,15,9,11,18,10,19,12,7,2,4,14,0,17,16,6,5	1431	2.053
7	4,12,8,14,18,11,13,19,16,5,0,7,2,15,1,9,10,17,3,6	1959	2.935

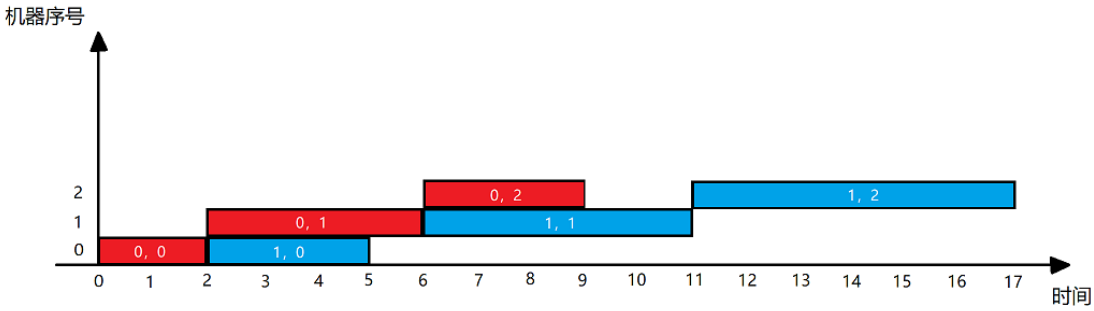
8	5,13,1,8,6,18,3,2,0,7,16,11,15,9,4,12,10,14,17,19	1109	1.578
9	12,17,1,16,7,8,18,10,2,14,19,3,11,13,9,6,15,0,5,4	1902	3.112
10	12,13,39,38,49,8,24,41,32,33,1,43,7,37,40,35,36,48,25,4,10,20,2,16,18,26,44,47,27,34,22,6,45,19,11,28,9,21,14,3,42,17,0,29,15,5,46,31,23,30	3277	3.875

调度顺序 sequence 表示每个机器处理零件的次序。对于示例:

Component = 2, Machine = 3

sequence = {0, 2, 1, 4, 2, 3}

sequence = {0, 3, 1, 5, 2, 6}



甘特图

由于每次实验具有随机性，因此对于同一组参数，可能产生不同的结果。每组测试用例运行 100 次并选择其中结果最好的组作为最终结果。

实验中经常出现优化某个参数反而使结果更差的情况出现，分析其原因可能是其他参数没有跟着一同优化的缘故。因此优化参数的时候要考虑参数间的制衡关系，不能只盲目优化一个参数。

下面四个表格分别对 T0，T1，dt 三个参数进行控制变量实验，部分分析参考自 CSDN 博客[3]：

表 1 对参数 T0 的对比实验

参数 (T0, T1, dt, L)	运行结果	运行时间 （秒）
1000.0, 1.0, 0.97, 50	8424	0.220
10000.0, 1.0, 0.97, 50	8420	0.432
100000.0, 1.0, 0.97, 50	8366	0.746

起始温度 T0 越高，状态改变时有更大的概率选择结果偏大的策略，从而降低陷入局部最优解的可能性，增加全局搜索的性能，运行结果相对更好；与此同时，增大参数 T0 也会导致运行时间增加，因此在设置参数时需要综合考虑运算成本和结果两个因素。

表 2 对参数 T1 的对比实验

参数 (T0, T1, dt, L)	运行结果	运行时间 （秒）
1000.0, 10.0, 0.97, 50	8477	0.022
1000.0, 0.1, 0.97, 50	8477	0.032
1000.0, 1e-3, 0.97, 50	8366	0.046

终止温度 T1 越小，退火次数越多，算法越能稳定收敛，得到的运行结果也更优。如果 T1 的值偏高，可能导致在尚未到达局部最优解时就停止。另外，随着 T1 的降低，运行所消耗的时间也在上升，因此应当考虑计算耗时的同时追求更优的解。

表 3 对参数 dt 的对比实验

参数 (T0, T1, dt, L)	运行结果	运行时间 （秒）
10000.0, 0.1, 0.90, 50	8564	0.021

10000.0, 0.1, 0.92, 50	8420	0.052
10000.0, 0.1, 0.99, 50	8366	0.092

降温系数 dt 应当略小于 1。降温系数越大降温速度越慢，反之越快。如果降温系数偏大，则程序运行的时间开销过高；如果降温系数偏小，则可能在得到最优解之前就降到了终止温度以下。

4 总结

本实验采用了模拟退火算法，对流水车间调度问题进行了求解。根据不同用例的特点，在调整初始参数 T，T1，dt，L 后可以在较短时间内得到相对优秀的调度方案与对应的耗时。该算法既解决了传统算法耗时过长的问题，同时又能保证算法收敛到相对优秀的解上，平衡了算法搜索性能与时间开销。

参考文献:

[1] lyrichu，模拟退火算法（SA）求解 TSP 问题（C 语言实现），https://www.cnblogs.com/lyrichu/p/6688459.html

[2] 青天白云飞，模拟退火算法（SA），https://blog.csdn.net/huahua19891221/article/details/81737053

[3] zhubaohua_bupt，模拟退火算法参数分析，https://blog.csdn.net/zhubaohua_bupt/article/details/79178341