[基于模拟退火算法实现的流水车间调度问题]

刘栋 (学号:1120191493)

摘　要: 本问题是一个经典的流水车间调度问题：n 个工件在 m 个机器上加工，每个工件需要有序地经过 m 个工序的加工，每个工序对应一个机器。注意，每个机器在一个时刻只能加工一个工件，每个工件不能同时在不同的机器上加工。给定每个工件各个工序所需时间，要求得到 n 个工件在每台机器上最优的加工顺序，使总完工时间达到最小。我采用的实验方法是模拟退火算法，这是一种通过模拟物质退火过程得到全局最优或者局部最优的方法，主要的实现思路是通过完全接受领域较优解和通过一定概率接受领域次优解，从而实现寻找到全局最优或者局部最优。通过这次实验，我成功求取到了给定十个样例的较优解，并通过了控制变量法进行实验，观测了实验中的各个参数对于运行结果，运行时间，退火次数的影响，同时从原理的角度分析了造成这些影响的原因。

关键词: [模拟退火算法]、[流水车间调度问题]

1 引言

a) [问题背景]

在当今社会，效率变得越来越重要。因此，任务优化问题成为了一个非常重要的问题。在实际生产中，常常会遇到多个机器负责工件的不同工序的加工模型，如何合理地指定不同机器加工工件的顺序，使得能在较短的时间内协调不同机器加工完更多的工件，成为了一个非常重要的工作调度问题。在这种背景下，合理地使用优化算法编程实现求解此类流水车间调度问题，成为了一个非常有实用意义的问题。

b) [使用数学语言详细描述出其中的优化调度问题]

有机器 $M_1,M_2,\ldots, M_m$,有工件 $c_1,c_2,\ldots, c_n$,他们的在各个机器上的加工时间可以用一个 m*n 的矩阵 H 来表示，其中在序号为 i 机器 M_i 上工件 j 加工的时间为 H_{ij}，约束为每个时间内每个机器只有一个工件可以被加工，我们的目的是要找出使得总时间开销最小的加工工件顺序。

设前 i 个机器加工到第 j 个工序时所耗总时为 S_{ij}

min S_{nk}

var.

$S_{ij} =\min\{S_{(i-1)j},S_{i(j-1)}\}+ H_{ij}$,1<=i<n,1<=j<k

$S_{00}=H_{00}$,i=0 且 j=0

$S_{0j}=H_{00}+ H_{01}+\ldots+ H_{0j}$,i=0 且 1<=j<n;

$S_{i0}= S_{(i-1)0}+H_{i0}$, j=0 且 1<=i<k

s.t. $S_{ij} <S_{(i+1)j}$ $S_{ij} < S_{(i+1)j}$

c) [解决方案]

本实验主要通过模拟退火算法实现此类算法。首先设置初始温度 T_0，衰减因子 alpha，之后在每一次迭代过程中将 T_0*alpha 赋于 T_0, 模拟实现退火过程。在每一次迭代过程中，会随机选取一个相邻状态作为领域(这里的相邻状态是指与当前工序只有一对工序不同的状态)，并根据实验结果决定是否接受此领域解。最终当温度下降到 T_{min} 时，算法结束。输出实验得到的最优解和最优车间顺序。

2 算法设计

[算法思路简介]

算法思路：模拟退火法，即通过模拟物质的退火过程来实现求取最优或次优解的过程。首先设定一个极

高的初始温度，然后开始演算算法，即进行多轮循环，在每次循环给温度以某个系数降温，直到温度由 T_0 降到规定温度 T_{min} 为止。首先根据随机算法 shuffle 设定一个初始状态，之后在循环的过程中，首先计算此状态所需时间，然后随机计算此状态的某个次态，如果此次态所用时比现在的状态优，则接受这个解；否则以 exp(-dE/T)的概率接受这个次态解。

[算法关键操作]：

1. 随机初始化初始解:通过 shuffle 算法随机选点保证加工顺序初始解的随机性。
2. 随机选取领域解：随机选取当前解的两个位置交换作为领域解。
3. 根据领域解的结果预估是否接受此解。如果计算得到的领域解解法较优，则接受此领域解；否则以 exp(-dE/T)的概率接受此解。

[算法伪代码]：

Algorithm 1 Pseudocode of Simulated Annealing Algorithm for Flow Shop Scheduling Problem

Input: $state_0$: initial processing state; T_0: a high enough initial temperature; T_{min}: the lowest limit of temperature; α: the decay factor

Output: $state_{best}$: optimal state or approximate optimal state;

randomly select a initial value of $state_0$;
set $state_{best} = state_0$
set $T = T_0$
compute initial energy function: $E(state_0)$;
while $T > T_{min}$ **do**
    randomly select a next state $state_{next}$ of $state_{best}$
    compute energy function of $state_{next}$: $E(state_{next})$;
    compute $\Delta E = E(state_{next}) - E(state_{best})$;
    if $\Delta E < 0$ **then**
        $state_{best} = state_{next}$;
    else
        the probability $P = exp(-\Delta E/T)$;
        if $rand(0,1) < P$ **then**
            $state_{best} = state_{next}$;
        end if
    end if
    $T = T * \alpha$;
end while

@Author: Liu Dong @Lisence: (C)Copyright 2020-2021, BITLD

[算法核心部分的时间复杂度]：$O(\log_{(factor)}(T_0/T_{min}))$

[算法核心部分的空间复杂度]：O(m*n)

3 实验

3.1 实验设置

实验环境：vscode

运行时间：所有样例控制在 10min 以内。

实验参数：初始温度 T_0，截至温度 T_{min},下降速率因子 factor

3.2 实验结果 [对实验结果进行分析]-3分

3.2.1 实验结果

用例	运行结果	加工方案	运行时间(秒)
例一	**8366**	(6,2,7,4,1,0,5,3)	**3.234**
例二	**7166**	(6,2,3,10,8,0,12,1,7,4,11,5,9)	**8.546**
例三	**7312**	(10,5,4,8,11,9,2,1,3,6,7,0)	**32.828**
例四	**8003**	(3,13,10,12,5,2,8,11,6,0,1,9,7,4)	**164.053**
例五	**7720**	(3,4,1,0,2,7,5,9,8,6)	**80.405**
例六	**1431**	(15,13,8,11,1,19,3,12,9,18,2,0,17,14,4,7,10,16,6,5)	**148.704**
例七	**1961**	(12,8,4,14,5,18,13,0,19,16,7,2,11,15,9,10,17,6,1,3)	**324.226**
例八	**1109**	(5,13,6,1,12,0,3,2,7,16,10,9,15,18,4,8,14,11,17,19)	**212.347**
例九	**1909**	(12,17,19,7,16,3,10,2,18,1,14,8,11,13,9,6,15,0,5,4)	**289.785**
例十	**3277**	(12,13,37,49,18,38,26,43,35,1,33,2,19,34,8,32,41,14, 31,10,0,11,4,16,44,27,21,9,24,17,5,22,25,20,39,28, 36,42,40,46,3,7,45,29,6,47,48,15,23,30)	**464.281**

3.2.2 对实验结果进行分析

从对十个例子的加工方案上来看，一般来说，就是数据规模越大，运行的轮数越大，所需时间越多。比如用例一相对于用例十所耗时小于 1/100。另外一点发现就是数据规模越大，需要的 factor 要更加接近于 1，需要的 T_0 需要越大，T_{min} 需要越小。

3.2.3.不同实验参数的对比分析

我对于三个参数进行了控制变量实验，即通过控制 T_0, T_{min},factor 中的三个参数中两个参数不变去观察变化的参数对实验结果的影响。所得实验效果如图：

3.2.3.1 factor 对实验结果，实验运行时间，实验迭代次数的影响。

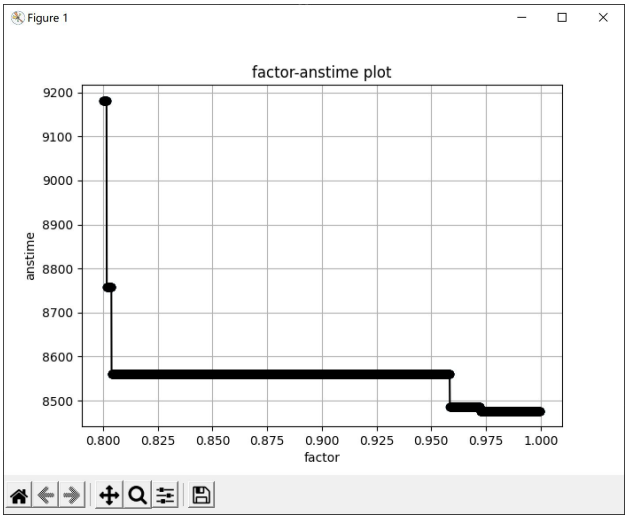

fig1-factor-实验最终结果(anstime)之间关系

(数据:用例一,T_0=1,T_{min}=1e-5,factor=0.8 到 0.9999)

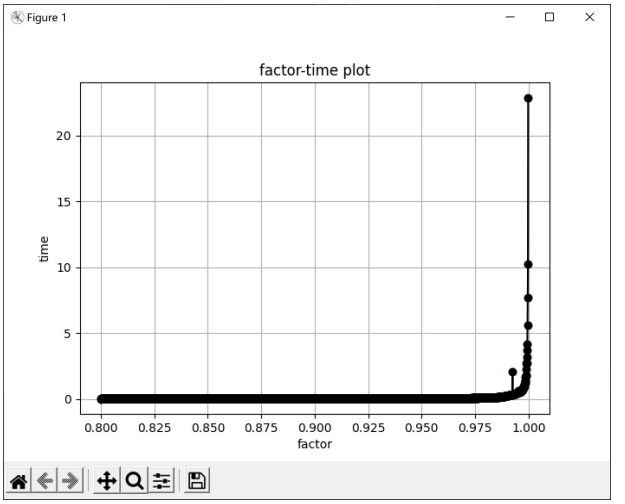

fig2-factor 与实验运行时间(time)之间的关系
(数据:用例一,T_0=1,T_{min}=1e-5,factor=0.8 到 0.9999)

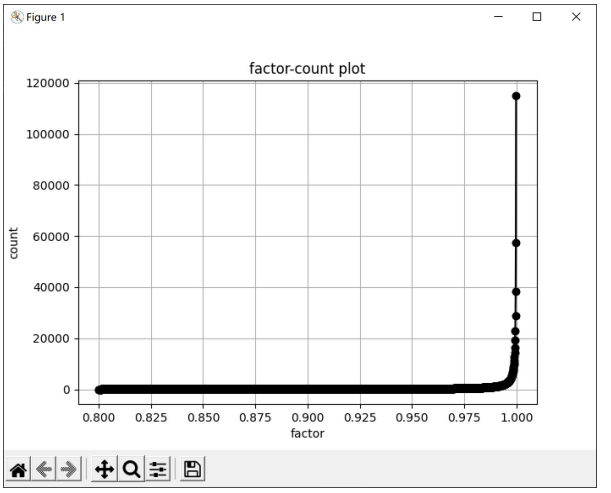

fig3-factor 与退火次数(count)之间的关系
(数据:用例一,T_0=1,T_{min}=1e-5,factor=0.8 到 0.9999)

现象：从上述实验可以看出，随着 factor 越来越趋近于 1，anstime 会出现周期性下降，但是时间和退火次数会出现指数级上升。

分析：由于 factor 与退火次数的关系是指数级别的(时间复杂度为 $O(\log_{(factor)}(T_0/T_{min}))$可以看出，所以当 factor 越来越趋近于 1，实验中的退火次数会大大增加，退火次数的增加会使得产生领域解的机会大大增多，这也就增大了产生较有领域解的机会，所以 factor 会出现周期性下降。

3.2.3.2　T_0 对实验结果，实验运行时间，实验退火次数的影响。

通过实验观察，通过现象改变 T_0，实验结果，实验运行时间，实验退火次数的变化不明显，因此决定通过改变 T_0 来观察 factor 与实验结果，实验运行时间，实验退火次数的各个图的变化来观察 T_0 对实验结果，实验运行时间，实验退火次数的影响。

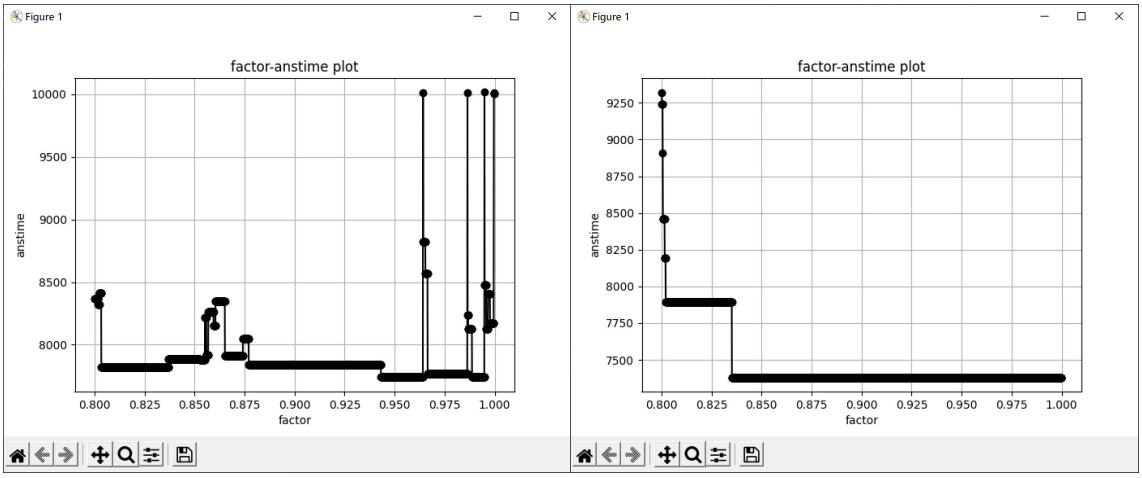

fig4-对比两种 T_0 下 factor 与实验最终结果(anstime)的关系

(左图:数据:用例三,T_0=100,T_{min}=1e-5,factor=0.8 到 0.9999)

(右图:数据:用例三,T_0=1,T_{min}=1e-5,factor=0.8 到 0.9999)

现象:当 T_0 较大时，实验结果会出现较大波动，而 T_0 较小时，实验结果会出现均匀稳定下降。

分析: 当 T_0 较大时，迭代次数较多，产生较差解的数量较大，因此在大量的次优解下，由较优解跳转到较差解的总数相对较大，因此会产生较明显波动波动。而当 T_0 较小时，迭代次数较小，产生的较差解数量较小，由较优解跳转到较差解的总数相对较小，因此波动不明显，大致呈现阶段性下降。

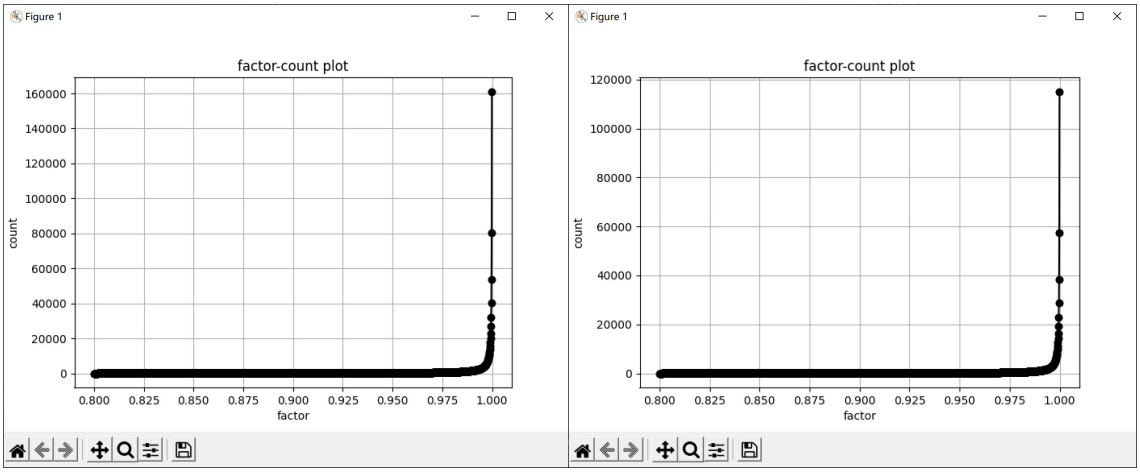

fig5-对比两种 T_0 下 factor 与退火次数(count)的时间

(左图:数据:用例三,T_0=100,T_{min}=1e-5,factor=0.8 到 0.9999)

(右图:数据:用例三,T_0=1,T_{min}=1e-5,factor=0.8 到 0.9999)

现象:通过观察坐标差异可知，当 T_0 较大时，退火次数较多。当 T_0 较小时，退火次数较小。

分析:根据 $T_{min}=T_0*factor^k$(k 表示退火次数)。当 T_{min}，factor(0<factor<1)一定时，T_0 越大，退火次数越大，

也就对应了高温退火到低温时所需退火次数较大; T_0 越小，退火次数越小，也就对应了较低温退火到一定温度时所需退火次数较小。

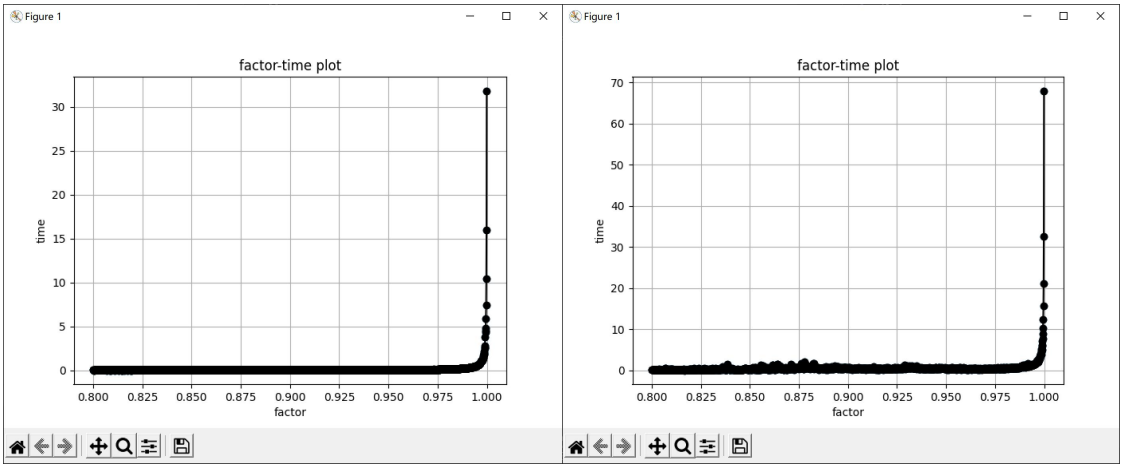

fig6-对比两种 T_0 下 factor 与实验运行时间(time)的图

(左图:数据:用例三,T_0=100,T_{min}=1e-5,factor=0.8 到 0.9999)

(右图:数据:用例三,T_0=1,T_{min}=1e-5,factor=0.8 到 0.9999)

现象:当 T_0 较大时，总程序用时较短。当 T_0 较小时，总程序运行时间较长。

分析:由前面的实验结果可知当 T_0 较大时，退火次数较多，当 T_0 较小时，退火次数较小。但是用时出现了相反的现象。我通过进一步实验，发现 T_0 较大时，虽然退火次数较多，但是每一次退火过程耗时较短；当 T_0 较小时，虽然退火次数较少，但是每一次退火过程耗时较大。通过分析，我认为这主要由两个因素有关，其一，运行时间与是否接受较差解的概率有关。在产生次态解时，其实还是以产生较差解为主(毕竟较优解还是少数的)，所以接受较差解概率的大小对程序的运行时间影响很大。当产生解为此优解时，我们会以 exp(-dE/T)的概率接受此较差解，其中 dE>0，因此当 T 越小，那么就意味着接受较差解的概率越大。其二，运行时间与状态更改时的耗时有关。在接受一个解的时，会产生状态的更改，即将次态赋值给当当前态，而这一步会消耗较长时间。综上所述，两个原因共同导致了 T_0 较小时耗时反而较大。

3.2.3.3 T_{min} 对实验结果，实验运行时间，实验退火次数的影响。

通过实验观察，通过现象改变 T_{min}，实验结果，实验运行时间，实验退火次数的变化不明显，因此决定通过改变 T_{min} 来观察 factor 与实验结果，实验运行时间，实验退火次数的各个图的变化来观察 T_{min} 对实验结果，实验运行时间，实验退火次数的影响。

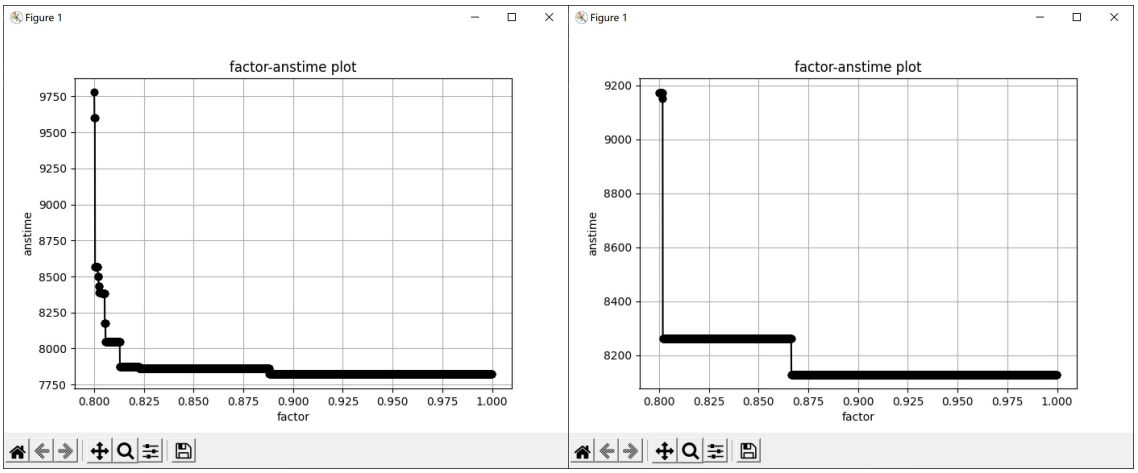

fig7-对比两种 T_{min} 下与实验最终结果(anstime)的关系

(左图:数据:用例五, T_0=1, T_{min}=1e-3,factor=0.8 到 0.9999)

(右图:数据:用例五, T_0=1, T_{min}=1e-6,factor=0.8 到 0.9999)

现象:当 T_{min} 较大时，程序到达最终结果的过程中下降次数较多，且各段降幅较小，当 T_{min} 较小时，程序到达最终结果的过程中下降次数较小，且各段降幅较大。

分析: 当 T_{min} 较大时，由于退火的次数较小，那么就会直接导致接触到的解的数目较少，所以就会容易满足与在较小解空间中的局部最优解，因此会出现多次下降；当 T_{min} 较小时，由于退火的次数较大，使得解空间较大，不容易限于较小解空间中的局部最优，因此不会出现多次下降。

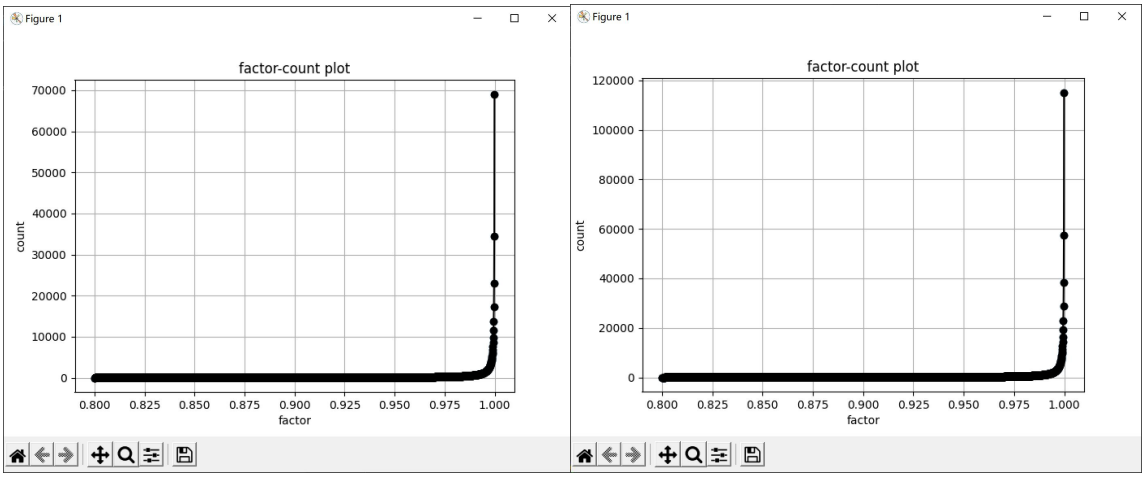

fig8-对比两种 T_{min} 下与退火次数(count)的关系

(左图:数据:用例五, T_0=1, T_{min}=1e-3,factor=0.8 到 0.9999)

(右图:数据:用例五, T_0=1, T_{min}=1e-6,factor=0.8 到 0.9999)

现象: 当 T_{min} 较大时，退火次数较小。当 T_0 较小时，退火次数较大。

解释: 根据 $T_{min}=T_0*factor^{k}$(k 表示退火次数)。当 T_0，factor(0<factor<1)一定时，T_{min} 越大，退火次数越小，

也就对应了从同一温度退火到较高温度时所需退火次数较小; T_{min} 越小，退火次数越大，也就对应了从同一温度退火到较低温度时所需退火次数较大。

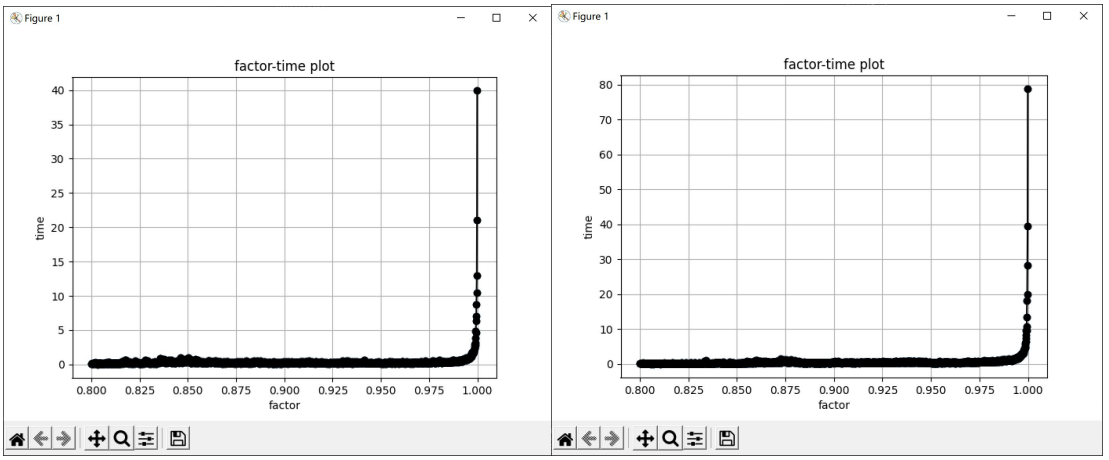

fig9-对比两种 T_{min} 下 factor 与实验运行时间(time)的图

(左图:数据:用例五, T_0=1, T_{min}=1e-3,factor=0.8 到 0.9999)

(右图:数据:用例五, T_0=1, T_{min}=1e-6,factor=0.8 到 0.9999)

现象: 当 T_{min} 较大时，总程序用时较短。当 T_{min} 较小时，总程序运行时间较长。

解释: 这与退火次数相吻合。T_{min} 较大时，退火次数较小，总程序用时短; T_{min} 较小时，退火次数较大，总程序用时长。(注:由于此处与分析 T_0 对实验运行时间影响不同，对其分析时提到第一个因素:"是否接受较差解的概率"在此处不成立。因为此处为观察 T_{min} 较对实验运行时间的实验，即 T_{min} 确定。在 T_0 确定的情况下，接受较差解概率有着相同的上限，而且它们的下限也都是趋于 0)。

4 总结

通过这次实验，我进一步熟悉了模拟退火算法解决了流水车间调度问题，通过这次实验我成功求取到了给定十个样例的较优解，并通过了控制变量法进行实验，观测了实验中的各个参数对于运行结果，运行时间，退火次数的影响，同时从原理的角度分析了造成这些影响的原因。模拟退火算法是最优化理论中的一个经典算法，掌握模拟退火算法为之后进一步进行学习和科研大有脾益。

笔者对于有一个错误的证明如下，原因是没有考虑到工件加工的节省时间的前后性，即，没有考虑到：即使前面浪费时间，后面也也能节省时间。希望作为反例，引以为戒。

证明:
数学归纳法:
前提：如果我们对于第一个机器的确定了一个加工顺序(此处不保证最优)
1. 对于第一个机器，最小加工时间产生于此加工顺序无间隔的执行—由此确定了第一个机器的各个工件的加工时刻。
2. 对于第 i+1 个机器，下证如果第 i 个各个工件的加工顺序 s 确定了，为了达到最小加工时间，第 i+1 机器的加工顺序 s 必为第 i 个机器的加工顺序。
   2.1 设第 i 个机器的结束时刻为 t1e,t2e,t3e,……tne 总时间花销为 T0，工作时间为 Ts(i)，每个工件加工用时分别为 T(i)
   2.2 对于第 i+1 个机器工作时间为 Ts(i+1)，易知，无论以什么顺序，机器的工作时间必为 Ts(i+1)，那么我们的工作就是尽量让工件之间的时间间隔尽量地小，如果第 k 个机器的不在该机器的所在位置，设 tke 后第一个开始工作的机器为 tre'，而此设此处可能有 delta=tre'-tke(if tre'>tke)那么就一定有时间冗余，如果 tre'<=tke 的时候 delta=0。
      2.2.1 如果 tre'>tke
         2.2.1.1 那么就会有比直接在 tke 处直接加工第 k 个工件的产生时间差 delta1=tre'-tke;
         2.2.1.2 另外考虑由于第 k 个工件加工本身产生的时间差(从前一个工件加工到此工件加工完)：如果直接加工 k 工件，由于不需要延时，所以 t1=0+T(i)(tke 开始)；如果之后加工 k 工件，由于在时间 tke 之后无需等待，那么就是 t2=0+T(i)(tle'之后某时刻开始),delta=t2-t1=0。
      2.2.2. 如果 tre'<=tke,类似 2.2.1.2,容易证明 delta=0
   2.3 delta>=0 成立，所以第 i+1 机器的最优解必然是保持原有顺序。
3. 结论：由上述数学归纳法可得，如果第一个机器的加工顺序固定，对于之后的每个机器加工顺序 s，必然在维持与第一个机器相同顺序时有最优时间。

本问题采用的是一维全搜索的方法，即，通过模拟退火算法选择一个较优的机器的加工顺序，而所有机器的加工与此工序相同。笔者通过查阅资料，从《混合流水线生产计划与调度问题研究》和《基于差分进化的优化算法及应用研究》两篇论文中都提到了对于本问题的解释，其中《混合流水线生产计划与调度问题研究》中明确指出，当各个机器加工时间方差较小时，选择一维搜索算法与二维搜索算法得到结果相近，而且可以大大节省算法运行时间。

另外，对于解决本问题全搜索的方法，笔者给出一个经典的工作调度竞赛问题:“NOIP2006 提高组：作业调度方案”。在此问题的正规解决方案中，巧妙地节省了空间复杂度至 o(n*m)，(如果考虑全搜索的话，一般解法大概率会提升至 o((n!*m))),因此如果想要暴力全搜索本问题，可以参考本题。

参考文献:

[1]刘建国. 混合流水线生产计划与调度问题研究[D]. 南京航空航天大学, 2008.
[2]董明刚. 基于差分进化的优化算法及应用研究[D]. 浙江大学, 2012.
[3]韩玉艳. 基于进化优化的多目标批量流水线调度[D]. 中国矿业大学, 2016.