基于登山算法和模拟退火算法的车间调度优化方法

摘要：车间调度问题是指确定元件在机器生产中的加工顺序，使得最大完工时间最小化。该问题是典型的 NP-Hard 问题，没有精确的最优解算法。而传统的最优化方法难以在最少的时间成本内得到该问题的最优解。本文基于登山算法和模拟退火算法两种经典的智能优化算法以及一些优化办法提出了解决车间调度问题的可行的解决方案，并且使用具体用例测试方案的可行性。此外，通过实验比较了同一算法设置不同参数的效果差异（包括时间成本与最优结果）、同一算法优化前后的效果差异以及不同智能优化算法之间的优劣差异。最终的实验结果表明在其他条件相同的情况下，模拟退火算法能够在较短的时间成本内给出优于登山算法的结果。

关键词：车间调度问题；登山算法；模拟退火算法；全局搜索性

JSP Based on Hill-Climbing Algorithm and Simulated Annealing Algorithm

Abstract: The shop scheduling problem involves determining the order in which components are processed in machine production so that the maximum make-time is minimized. This problem is a typical NP-Hard problem, and there is no exact optimal solution algorithm. However, it is difficult for traditional optimization methods to obtain the optimal solution to this problem in the least time cost. Based on the two classic intelligent optimization algorithms of mountain climbing algorithm and simulated annealing algorithm, as well as some optimization methods, this paper proposes a feasible solution to solve the workshop scheduling problem, and uses specific use cases to test the feasibility of the solution. In addition, through experiments, the differences in the effect of the same algorithm setting different parameters (including time cost and optimal results), the difference in the effect of the same algorithm before and after optimization, and the difference between the advantages and disadvantages of different intelligent optimization algorithms are compared. The final experimental results show that under the same conditions, the simulated annealing algorithm can give better results than the mountain climbing algorithm in a shorter time cost.

Key word: Job-shop scheduling problem; hill-climbing algorithm; simulated annealing algorithm; global searchability

1 引言

1.1 问题背景

随着现代工业的发展和科技的进步，特别是近年来电商、交通、物流和工业化生产等产业的快速发展，组合优化问题越来越广泛地出现在生产生活中的各个领域。车间调度问题作为组合优化中一类重要的调度问题，涉及到交通运输、网络通信、航空航天和工业生产作业计划等领域，应用十分广泛。车间调度问题的核心目标是通过通过对现有资源的合理分配与调度，实现资源的高效利用，提高工作的生产效率，降低生产成本。因此，研究车间调度问题对于具有人力、机器和时间成本等约束的情况（如：现代企业生产等）具有十分重要的意义。

1.2 问题形式化描述

车间调度问题可以描述为：在 m 台机器上加工 n 个元件，每个元件都有 m 个机器进行加工（机器加工的顺序一定）。问题的约束条件如下：

1. 不同元件的加工没有先后顺序的约束，同一元件的加工顺序有先后约束。
2. 同一时刻，一个元件只能在一台机器上加工，一台机器一个时刻只能加工一个元件，并且机器一旦开始加工就不能中断

下面给出对于车间调度问题的抽象描述 [1]：

1. n 个待加工元件，记为 $J_1, J_2, \cdots, J_n$
2. m 台加工机器，记为 $M_1, M_2, \cdots, M_m$
3. s_{ij} 表示第 i 个元件在第 j 台机器上的开工时间，t_{ij} 表示第 i 个元件在第 j 台机器上的加工时间，c_{ij} 表示第 i 个元件在第 j 台机器上的完工时间，M 是一个无穷大的数

车间调度问题求解的目标是使最大完工时间 C_{max} 最小化，以及寻找与之相对应的元件加工顺序。车间调度问题的数学描述如下所示：

$$C_{max} = min(\max_{1\le i\le n, 1\le k\le m} c_{ik})$$

$$s.t. \begin{cases} c_{ik} - t_{ik} + M(1 - a_{ihk}) \ge c_{ih} \\ c_{jk} - c_{ik} + M(1 - x_{ijk}) \ge t_{jk} \\ c_{ik} - t_{ik} \ge s_{ik} \\ s_{ik} \ge 0 \end{cases}$$

其中，

$$a_{ihk} = \begin{cases} 1 & \text{机器}h\text{先于机器}k\text{加工元件}i \\ 0 & \text{其他} \end{cases}$$

$$x_{ijk} = \begin{cases} 1 & \text{元件}i\text{先于元件}j\text{到达机器}k \\ 0 & \text{其他} \end{cases}$$

$$i, j = 1, 2, \cdots, n \quad h, k = 1, 2, \cdots, m$$

1.3 研究现状与解决方案

鉴于传统优化算法在处理车间调度问题时的缺点，本文采用智能优化算法中的登山算法和模拟退火算法作为求解车间调度问题的主要优化算法。针对车间调度问题以及各种算法的特点，文章采用封装的思想设计了测试用例的读入以及算法的处理接口。此外，各种算法的封装便于重复使用用例对算法进行测试，依此进行多次对比实验得到一系列实验结果。

实验的主要步骤如下：

1. 建立问题模型：针对车间调度问题给出形式化描述并抽象成数学语言，建立模型将测试数据进行封装。
2. 设计搜索算法：设计搜索最优解的算法流程，封装后为问题模型提供处理接口。
3. 编写程序并试运行：通过编程语言实现算法并输出结果。
4. 对比优化：对算法、程序等进行优化或者调试参数，再次运行程序，将所得结果进行对比得出最优方案

经过一系列实验测试后，对比不同的实验结果得出的结论如下：

1. 模拟退火算法能够在较短的时间成本内给出优于登山算法的结果，算法性能要优于登山算法；
2. 优化后的模拟退火算法性能略强于经典模拟退火算法；
3. 模拟退火算法与其他几种智能优化算法相比各有优劣。

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

2.1 思路简介

由于车间调度问题是 NP-Hard 问题，并且在实际生活中车间调度问题的规模更大、复杂性更高，具有很强的不确定性，传统的最优化算法很难在有限的时间和资源内给出令人满意的结果 [2]。因此，设计的算法一般不要求目标函数和约束条件具有连续性与凸性，而且必须对计算中数据的不确定性具有较强的适应能力，能够在解空间中进行高效搜索，同时避免陷入局部最优。为了找到合适的方案，文章使用了登山算法和模拟退火算法作为研究问题的主要算法。另一方面，文章对这两种智能优化算法也作出了一定优化，主要从提高解的质量和缩小最优解搜索空间的角度，来提高智能优化算法在车间调度问题的搜索效率。[3]

2.2 两种智能优化算法

2.2.1 登山算法

登山算法是一种启发式的最优化算法，具有局部贪心的特点。算法的基本思想是将最优解处当作 ‘‘山顶’’，随机指定搜索的起点后，比较临近解空间中解与当前解的优劣，不断用更优解更新所在位置，最终在搜索到某个峰值所在的位置后停止，就像登山的过程一样不断向着高处迈进。

Algorithm 1 Hill-Climbing Algorithm

```
for i=0 to iteration do
    随机初始化生成解
    while 循环达到搜索次数上限或者解没有更新 do
        遍历临近解空间所有解，选择最优解更新为当前解
    end while
end for
return result
```

Algorithm 1 为登山算法的基本流程。

循环的过程需要更新的变量有：

1. 存储最优解对应元件加工顺序的数组，大小为 n；
2. 计算加工总时间使用的二维数组，大小为 $m \times n$；
3. 其他个别临时变量。

因此登山算法的总空间复杂度为 $O((m+1)n)$。

登山算法的时间复杂度与其邻域的选择有关。一般来说登山算法中常见的邻域选择有以下几种方式（设外循环的次数为 C）：

1. 邻位对换法

   随机选择元件排列中除了最后一个元件之外的某一元件，将其与相邻的下一元件调换顺序。采用该方法生成邻域时，将得到较窄的邻域，邻域中排列的个数为 $n-1$。此时算法内循环的时间复杂度为 $O(m \times n^2)$，算法整体的时间复杂度为 $O(C \times m \times n^2)$。

2. 两点对换法

   该方法随机选择元件排列中的任意两个元件，调换所选两个元件的加工顺序，此方法将得到较为宽泛的邻域，邻域中排列的个数为 $\frac{n(n-1)}{2}$。此时算法内循环的时间复杂度为 $O(m \times n^3)$，算法整体的时间复杂度为 $O(C \times m \times n^3)$。

3. 区间逆序法

   随机选择元件排列中的任意两个元件，将排列中所选的两个元件及其之间的元件加工顺序做逆序处理，此方法将生成过于宽泛的邻域，这样会使得时间复杂度大大提高，与算法的初衷相违背，故实验中并没有采用此种方法。

2.2.2 模拟退火算法

经典模拟退火算法 模拟退火算法来源于固体退火原理：将固体加温至充分高，再让其缓慢降温（即退火）使之达到能量最低点；反之，如果急速降温（即淬火）则不能达到最低点。算法的基本思想是先从一个较高的初始温度出发，逐渐降低温度，直到温度降低到满足热平衡条件为止。在每个温度下，进行 n 轮搜索，每轮搜索时对旧解添加随机扰动生成新解，并按一定规则接受新解。模拟退火算法与登山算法最大的区别在于模拟退火算法有一定概率接受并不局部最优的解，能够更好地避免陷入局部最优的陷阱。

冷却进度表、新解产生器、Metropolis 准则三者构成了模拟退火算法的关键部分。

Algorithm 2 Simulated Annealing Algorithm

$s := s_0; e := E(s); T := T_0$
while $T > T_{min}$ **do**
  $s_n := neighbour(s)$
  $e_n := E(s_n)$
  if $random() < P(e, e_n, temp(k/kmax))$ **then**
    $s := s_n; e := e_n$
  end if $k := k + 1$
end while
return s

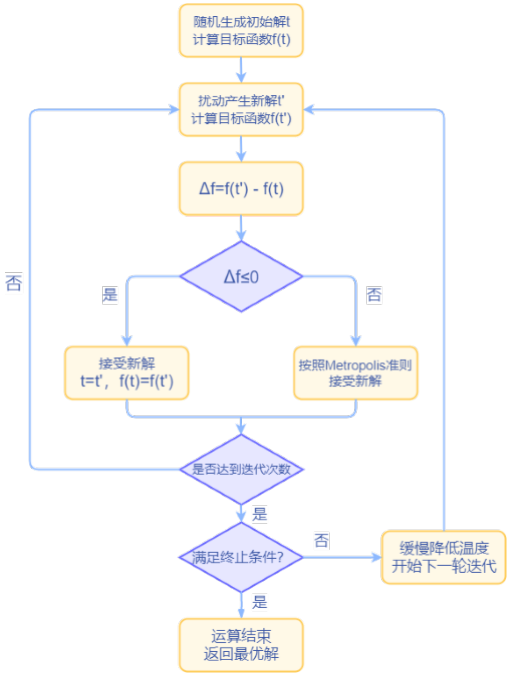

Figure 1: 模拟退火算法流程图

- 冷却进度表

  由六个基本参数构成：初始温度 T_0，衰减系数 α、搜索步长因子、每个温度值的迭代次数（马尔科夫链的长度）、初始解状态和结束温度阈值。

- 新解产生器

  首先由一个新解产生函数从当前解产生一个位于解空间的新解。为了便于后续的计算和接受，减少算法耗时，通常选择由当前解经过简单地变换即可产生新解的方法。

- Metropolis 准则

  Metropolis 准则用来判断一个新解是否被接受，这也是模拟退火算法最重要的部分。根据 Metropolis 准则，系统从当前状态 i 进入下一状态 j 时：若 $E_j < E_i$，则接受 j 为当前状态；否则，如果 $p = exp(-\frac{E_j - E_i}{T}) > rand(0,1)$，则接受状态 j 为当前状态；如果不成立，则保留状态 i 为当前状态。

Algorithm 2 和图 1 给出了算法的伪代码及流程图。

Algorithm 3 Tempering Strategy in Simulated Annealing Algorithm

```
BEGIN
if T_k / (T_{k+1} + Tempering^γ) > random[0,1) then
    T_{k+1} := T_{k+1}
    Tempering := Tempering + 1
else
    T_{k+1} := T_k
end if
```

模拟退火算法的空间复杂度的计算方法与登山算法类似，不再赘述，总空间复杂度仍然是 $O((m+1)n)$。

模拟退火算法分为内外层循环，其内循环的时间复杂度为 $O(m \times n^2)$，设外循环次数为 C（C 与初始温度以及温度阈值有关），故模拟退火算法的时间复杂度为 $O(C \times m \times n^2)$。

经典模拟退火算法的优化 对于模拟退火算法的优化主要体现在对于退火方式的调整。主流的退火方式是通过固定参数使得温度下降，也即 $T_{k+1} = \alpha T_k$。但是当退火到温度很低时，实际上此时的 Metropolis 准则对算法的运行已经起不到脱离局部最优陷阱的作用，并且在经典做法中，模拟退火算法的初始温度通常设置一个较大的值，与问题或者算法的运行过程无关。目前对于模拟退火算法主要从两个角度进行优化：

1. 数学角度：回火策略 [4]

   回火策略具体来说就是跟踪算法搜索的过程，观察退火搜索在中后期的状态，判断算法是否陷入局部搜索，如果是则通过回火以跳出局部搜索。为了将温度与算法的运行过程相关联，在温度的下降过程中引入了自适应设置：

   $$T_{k+1} = \alpha Q(T) + \beta |f_k - f_{k+1}|$$

   其中 $Q(T)$ 为正则化函数。当 $\alpha = 0$ 时，回火策略失效，公式退化成经典模拟退火算法的退火策略。为了反映历史模拟退火过程的影响，正则化补偿可以表示为

   $$\alpha Q(T) = \sum_{i=2}^{k-1} \alpha_i (\beta |f_{i-1} - f_i|)$$

   正则化补偿既利用了当前的求解结果，也利用了退火过程的 ‘‘历史信息’’，即利用算法的运行过程对退火温度进行预估和校正。当相邻两次退火搜索的目标函数值改变较小时，惯性项对温度的改变起到了主要的作用，相当于经典的模拟退火算法；当相邻两次退火搜索的目标函数值改变较大时，函数值的变化，也即 Δf 成为主项，惯性项起到了辅助作用。因此，引入正则化补偿后，模拟退火算法分成两个阶段：

   1. 前期保持较高的退火温度，扩大搜索范围
   2. 后期由惯性项做小幅度调整，以局部寻优为主

   Algorithm 3 是回火策略的模拟退火算法的简单流程。

2. 物理角度：$Cauchy-Lorentz$ 跃迁分布 [5]

经典模拟退火算法的能量跃迁分布采用了高斯形式：

$$g(\Delta x) \propto exp(-\frac{(\Delta x)^2}{T})$$

这里，Δx 是变量 x 的试探跃迁距离，T 是约化单位的退火温度。如果跃迁使得新状态的能量降低，则新状态被接受；否则，用 Metropolis 准则来判断是否接受新解：

$$p = min\{1, exp(-\frac{\Delta E}{T})\}$$

经典模拟退火算法采用的高斯跃迁形式的收敛速度相对较慢。后来有学者对此作出了优化，将高斯跃迁形式更改成半局域的 $Cauchy-Lorentz$ 跃迁分布：

$$g(\Delta x) \propto \frac{T}{(T^2 + (\Delta x)^2)^{\frac{D+1}{2}}}$$

这里，D 指的是变量空间的维数。由于 $Cauchy-Lorentz$ 跃迁分布是半局域的分布，所以在同样的温度下，$Cauchy-Lorentz$ 跃迁分布使得温度有更大的几率进行长距离的跃迁 [6]。

3 实验

3.1 实验设置

以下所有实验除特殊说明，都默认按照以下参数进行实验。

1. 相关实验环境：

处理器：AMD Ryzen 5 4600H with Radeon Graphics

操作系统：Windows 11

运行环境：Microsoft Visual Studio Code

编译器：编译器：MinGW-W64-builds-4.3.5

编程语言：C++

2. 登山算法实验参数：

最外层循环 $C = 3000$

3. 模拟退火算法实验参数 (如表 1 所示)

3.2 实验结果

3.2.1 登山算法和模拟退火算法的对比实验

对于车间调度问题的分析，甘特图是一种十分常用的方法，因为它通过条状图清晰明了地显示变量内在关系随着时间发展的情况。为了便于分析与展示，下面用一个简单的例子来展示甘特图的分析成果。

Table 1: 模拟退火算法实验参数

用例	初始温度 T_0	结束温度 T_d	衰减系数 α	迭代次数 $iter\_num$
0	1e4	1e-4	0.99	1000
1	1e4	1e-4	0.99	1000
2	1e4	1e-4	0.99	1000
3	1e4	1e-4	0.99	1000
4	1e4	1e-4	0.99	1000
5	1e4	1e-4	0.99	1000
6	1e4	1e-4	0.99	1000
7	1e6	1e-7	0.998	5000
8	1e4	1e-4	0.99	1000
9	1e5	1e-6	0.995	3000
10	1e4	1e-4	0.99	1000

Table 2: 甘特图数据用例

	0 号机器	1 号机器	2 号机器
0 号元件	4	2	3
1 号元件	1	7	6
2 号元件	3	4	9
3 号元件	6	5	8
4 号元件	5	3	4

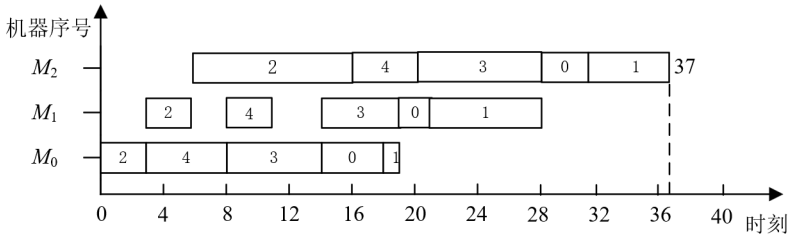

Figure 2: 甘特图

Table 3: 登山算法与模拟退火算法实验结果对比

用例	登山算法结果	模拟退火算法结果	模拟退火算法运行时间/s
0	7038	7038	2.68
1	8366	8366	2.80
2	7381	7166	2.43
3	7506	7312	2.77
4	8129	8003	2.60
5	7767	7720	2.89
6	1607	1431	8.34
7	2117	1950	186.23
8	1176	1109	4.06
9	2092	1902	51.73
10	3550	3277	17.53

表 2 是 5 个元件和 3 台机器对应的加工时间。

甘特图如图 2 所示。

甘特图的图示结果与登山算法和模拟退火算法运行后程序的输出一致。由此可见，甘特图的图示结果与算法运行结果相对比不失为一种评估算法的有效方法。

表 3 是模拟退火算法和登山算法的实验结果。其中模拟退火算法的结果均与甘特图的结果一致，准确率较高。

由表 3 可以看出，在实验设置中规定的参数的情况下，登山算法的实验结果普遍大于或等于模拟退火算法的实验结果，也就是登山算法计算出的最短元件加工时间要比模拟退火算法计算出的结果要长，并且平均相差近 100 左右，差距比较明显。由此可以确定，模拟退火算法的优化效果要显著好于登山算法。

本次实验中并未对登山算法和模拟退火算法做任何优化，采用的是两点对换法的登山算法和经典模拟退火算法（使用默认参数）。下面对两种算法进行分别讨论和具体优化。

3.2.2 针对登山算法的实验

根据之前的分析，登山算法的性能与邻域的选择有关。邻域的选择方式有两种：邻位对换法和两点对换法。为了使得两种方法的运行时间差异更加明显，同时尽量减小时间误差的影响，将登山算法外层循环的次数改成 10000 进行实验。在仅改变邻域选择的方式、其他参数不变的前提下，表 4 列出了两种方法下登山算法的运行结果以及运行时间。

可以看出，除了用例 7 和 8，两点对换法的登山算法运行结果都小于或等于邻位对换法的登山算法。这说明两点对换法随机生成的调换方式以及较为宽泛的邻域增加了登山算法的随机性，有利于登山算法避免陷入局部最优，得到更优的结果。在算法的运行时间方面，除了用例 1 和 10，两点对换法的运行时间都要比邻位对换法的运行时间长。由于两点对换法的邻域更宽泛，时间复杂度更高，所以相对来说耗费的时间也更多，与之前的时间复杂度分析结果相吻合。

Table 4: 两种方法下登山算法的运行结果

用例	两点对换法运行结果	两点对换法运行时间/s	邻位对换法运行结果	邻位对换法运行时间/s
0	7038	1.38	7038	1.47
1	8366	1.01	8366	0.67
2	7381	1.63	7381	0.81
3	7506	1.31	7594	0.72
4	8129	1.59	8214	0.72
5	7767	1.44	7832	0.66
6	1607	1.84	1608	1.17
7	2117	1.84	2115	1.02
8	1176	1.81	1161	1.42
9	2092	1.84	2109	1.20
10	3550	3.00	3563	3.69

Table 5: 初始温度对模拟退火算法的影响

初始温度 T_0	用例 9 运算结果	用例 9 运算时间/s	用例 10 运算结果	用例 10 运算时间/s
0.1	1920	18.92	3280	13.29
1	1924	25.45	3278	13.89
10	1917	31.63	3277	14.46
100	1911	38.60	3277	14.89
1000	1905	45.00	3277	16.03
1e4	1903	50.19	3277	17.53
1e5	1902	53.73	3277	18.02
1e6	1902	57.42	3277	19.86

3.2.3 针对模拟退火算法的参数实验

对于模拟退火算法，影响算法性能的参数主要有三个：初始温度 T_0、结束温度阈值 T_1 和衰减速率 α。其中，因为结束温度阈值 T_1 一般不取较大值，而取值过小时浪费时间和计算资源且算法效果提升并不显著，故在此不设置对照试验。为了便于实验参数的比较，节约时间成本，根据表 3 中列出的模拟退火算法的运行时间以及运行结果，下面选用典型用例 9 和 10 进行实验。

初始温度 T_0 初始温度是模拟退火算法中十分重要的参数之一。一般来说，更高的初始温度有利于算法避免陷入局部最优。但是同时，过高的初始温度也会增大循环次数，造成算法的运行过慢，容易浪费大量的资源。另一方面，过高的初始温度可能会使得算法的解分布过于分散，而不能往最优解的方向合理发展。控制其他参数不变，测试不同初始温度对算法性能影响的实验结果如下：

从表中可以看出，在一定范围内，初始温度 T_0 越大，算法运行的结果越接近最优值；当初始温度达到一定值后，运行结果已经达到最优值不会改变，继续增大初始温度会增加外循环的次数，使得运行时间变长，浪费计算资源。因此，实际进行最佳初始温度选择时，应注意合理比较，多次实验，以取得较好的效果。

Table 6: 衰减系数对模拟退火算法的影响

衰减系数 α	用例 9 运算结果	用例 9 运算时间/s	用例 10 运算结果	用例 10 运算时间/s
0.96	1930	5.93	3277	6.62
0.97	1933	7.82	3277	9.10
0.98	1932	12.16	3277	13.36
0.99	1920	24.01	3277	26.86
0.995	1902	49.37	3277	54.37

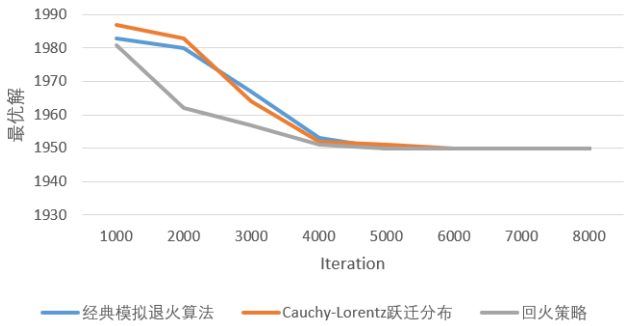

Figure 3: 用例 7 运行结果

衰减系数 α 衰减系数指温度下降的速率，用于控制温度的退火速度，此处的衰减系数为固定参数的比较。衰减系数一般取 $\alpha \in (0.8, 1)$，控制温度缓慢下降，逐步到达最优点。不同衰减系数的运行结果和运行时间如图所示。

从表中可以看出，其他参数保持不变的情况下，衰减系数越接近于 1，算法的运行结果越容易稳定在最优解附近，但运行时间也会大幅度增加，尤其是当 $\alpha > 0.99$ 后，算法运行时间几乎成倍增长；当衰减系数下降时，算法的运行结果也变得不稳定，且离最优解较远，优化效果衰退明显。在本实验的用例 9 和用例 10 中，0.99 是一个较为理想的结合了算法优化效果与运行时间的合适的衰减系数的取值。在实际应用时，应当权衡效果的优劣与时间资源成本，合理选择适当的退火速率 α。

3.2.4 针对模拟退火算法的对比实验

对于模拟退火算法的优化，目前主要体现在对于退火方式的调整。根据之前对于模拟退火算法的优化描述，现将经典模拟退火算法、回火策略的模拟退火算法和 $Cauchy - Lorentz$ 跃迁分布的模拟退火算法得到的运行结果进行对比。由于测试用例的数据规模有限，根据之前记录的经典模拟退火算法的运行时间，为了使得实验结果显现出的两种优化方案和经典算法的差异最大化，本实验选用用例 7 和用例 9 进行实验，通过改变内循环的次数观察算法的运行结果和运行时间。将最终得到的结果绘制成图像如图 3-6 所示。

从图中可以看出，可能是数据规模较小的原因，$Cauchy - Lorentz$ 跃迁分布的模拟退火算法和经典模拟退火算法的运行结果和运行时间都相差不大，两者性能并没有太大差异；而回火策略的模拟退火算法和经典模拟退火算法相比，回火策略的模拟退火算法的运行结果要略优于经典模拟退火算法。但是回火策略的运行时间成本要略高于经典模拟退火算法，尤其是在两者都未达到最优解的情况下，回火策略的模拟退火算法由于正则化补偿的效果显著，消耗的时间成本要大于经典模拟退火算法。在实际应用中，应当综合考虑数据的规模大小、时间

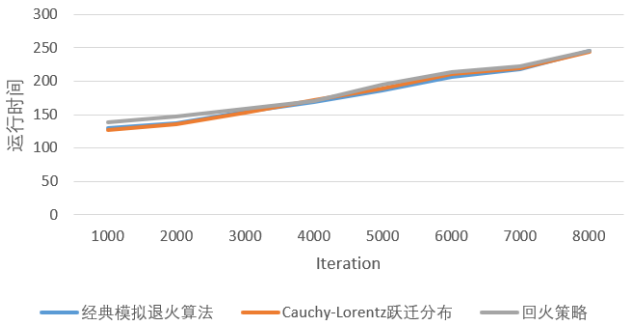

Figure 4: 用例 7 运行时间

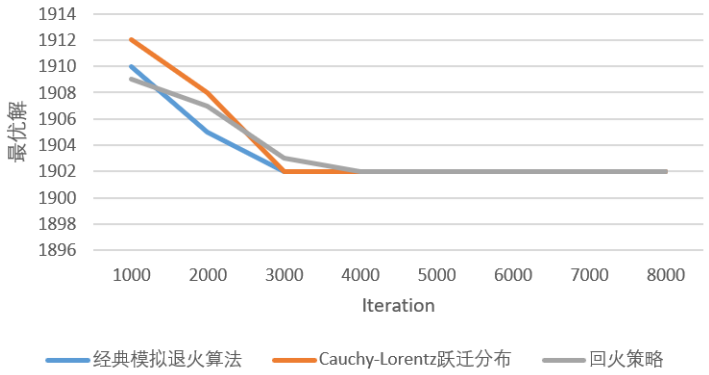

Figure 5: 用例 9 运行结果

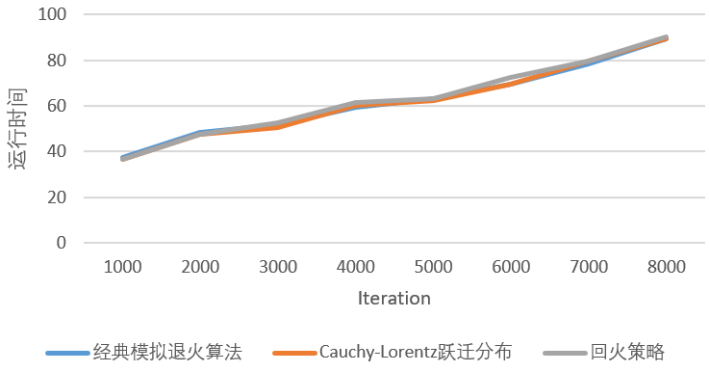

Figure 6: 用例 9 运行时间

Table 7: 模拟退火算法与蚁群算法的对比实验 1

用例	模拟退火算法结果	蚁群算法结果	模拟退火算法运行时间/s	蚁群算法运行时间/s
7	1950	1950	186.23	180.58
9	1902	1902	51.73	53.40
10	3277	3277	17.53	13.37

Table 8: 模拟退火算法与蚁群算法的对比实验 2

用例	模拟退火算法结果	蚁群算法结果
ulysses16	75.62	78.60
ulysses22	82.32	79.18
att48	37528.45	34458.79
eil51	504.76	448.29
berlin52	8551.55	7888.01
st70	853.25	749.99

成本等因素选择最合适的算法。

3.2.5 模拟退火算法与蚁群算法的对比实验

为了更加深入地了解模拟退火算法的性能，将其与其他智能优化算法进行比较，这里选取了蚁群算法。表 7 是在处理用例 9 和用例 10 时，蚁群算法和模拟退火算法的表现。

从表 7 可以看出，由于数据规模过小，蚁群算法和模拟退火算法的性能对比差距并不是十分明显，存在一定的偶然性。因此，实验摘录了部分从国际通用的 TSP 数据库 TSPLIB 中选取的数据进行辅助对比 [7]，如表 8 所示。

将表 7 和表 8 的实验结果结合起来不难看出，在问题规模较小时，蚁群算法和模拟退火算法都能取得不错的结果，而蚁群算法的运行时间要明显比模拟退火算法短；但是当问题规模变大时，模拟退火算法得到的最优解明显开始恶化，而蚁群算法依然保持不错的性能。

从理论上来说，小规模输入时，蚁群算法局部搜索能力强、收敛速度快的优点可以得到突出的显示，因为蚁群算法一次循环就取得最优解的概率要高于模拟退火算法。但当问题规模增大时，模拟退火算法具有全局搜索能力较强的特点就会凸显出来，而相比较来说蚁群算法更有可能陷入局部最优解。但在本次实验的过程中模拟退火算法在大规模输入上的处理性能不如蚁群算法。可能是因为模拟退火算法在算法即将结束时也有一定概率接受次优解，想办法随着算法的进行降低 Metropolis 准则的接受概率或许能够解决问题。而在运行时间方面，由于模拟退火算法采用的马尔科夫链具有串行性的特点，所以模拟退火算法的运行速度相对可能较慢，不如蚁群算法的收敛速度快。

从本次实验中，可以看出模拟退火算法虽然具有蚁群算法所不具有的全局搜索性的优势，但是实际操作中由于最优参数的调节以及马尔科夫链的串行性等因素的影响，模拟退火算法处理大规模输入时的性能反而不如蚁群算法。但鉴于模拟退火算法针对避免陷入局部最优做出的优化，或许将模拟退火算法和蚁群算法相结合，能够综合蚁群算法局部搜索能力强和模拟退火算法全局搜索性的特点，得到更好的结果。

4 总结

本文的研究核心是用登山算法和模拟退火算法来解决车间调度问题。首先给出车间调度问题的大致分析与数学描述，针对该问题的数学特点 (NP-Hard 问题)，选择了智能优化算法作为处理车间调度问题的有效算法。然后，本文从智能优化算法中选定了登山算法和模拟退火算法，了解了两种算法的相关研究成果后，对两种算法的思想和流程进行了详细的描述，分析了算法的复杂度并给出了可行的优化方案。同时，对车间调度问题与两种算法模型均采用类封装的思想处理，便于接口的衔接与调用。最后将思想落实到可执行程序上，对于给定的 11 个样例分别给出了对应的最短调度时间和最优调度方案。此外，本文还设计了多组参数实验和对比实验对两种算法的性能进行进一步的比较和研究，分析各自的优点与不足，给出实际应用时选择参数的评判标准参考，提出可能的改进措施优化算法性能。

同时，本文的实验设置存在一定的改进空间。在做模拟退火算法参数比较实验时，鉴于时间资源的限制，本文并没有对结束温度阈值做出详细的参数实验参考。而对于模拟退火算法的优化对比实验，由于实验设置的用例数值规模较小，可能并不能真正体现出优化后算法相较于优化前算法的性能的提升。此外，对于部分样例，程序的运行时间过长，并没能找到更加完美的参数组合或者优化方法。同时，程序的运行时间过长也是使得实验效果不够理想的重要因素。鉴于时间资源的限制，本文在进行模拟退火算法的实验时均只采用了两个用例，得到的结果不一定具有普遍性。这些问题可以通过增设对比试验的组数，多做几次实验取平均值消除系统计时带来的时间误差或者采取多线程运行程序等方式解决。

References

[1] 乔东平, 柏文通, 文笑雨, 李浩, 王雅静. 基于关联规则的作业车间调度问题改进遗传算法研究 [J]. 河南理工大学学报 (自然科学版),2022,41(02):138-148.

[2] 袁亮, 袁逸萍, 冯欢欢, 孙文磊. 基于智能优化算法的车间调度问题研究 [J]. 新疆大学学报(自然科学版),2014;31(03):363-368.

[3] 祁建程. 基于车间调度问题的智能优化算法研究 [D]. 浙江大学,2010.

[4] 李元香, 蒋文超, 项正龙, 张伟艳. 基于弛豫模型的模拟退火算法温度设置方法 [J]. 计算机学报. 2020;43(11):2084-100.

[5] 庞峰. 模拟退火算法的原理及算法在优化问题上的应用 [D]. 吉林大学,2006.

[6] Szu H, Hartley R. Fast simulated annealing[J]. Physics letters A. 1987 Jun 8;122(3-4):157-62.

[7] 许智宏, 宋勃, 董建波. 用蚂蚁算法和模拟退火算法解大规模 TSP 问题的研究 [J]. 计算机工程与科学,2008,(10):43-44+57.