# Research On the No-Wait Flow Shop Scheduling Problem Based on Simulated Annealing Algorithm.

Ming Kai *

School of Computer Science, Beijing Institute of Technology, Beijing, China, 102400

* Corresponding Author Email: mingkai1123@outlook.com

**Abstract.** This paper focuses on the No-Wait Flow Shop Scheduling Problem and conducts research based on an improved Simulated Annealing Algorithm. The No-Wait Flow Shop Scheduling Problem is of great significance in practical production, but its solution complexity is high, requiring the search for appropriate optimization methods. In this study, a solution based on an improved Simulated Annealing Algorithm is proposed. Simulated Annealing Algorithm, as a global optimization method, introduces random perturbations to the current solution and accepts or rejects the new solution based on Metropolis criterion to escape local minima and search for better solutions. In this paper, the termination condition of the improved Simulated Annealing Algorithm is enhanced. Specifically, if no better solution is obtained after a certain number of consecutive temperature calculations or when the maximum number of outer loop iterations is reached, the algorithm terminates and outputs the result. This improvement aims to increase efficiency and reduce unnecessary computations.By comparing the results with other heuristic algorithms, it is found that the solution based on the improved Simulated Annealing Algorithm can effectively provide better scheduling solutions for the No-Wait Flow Shop Scheduling Problem, thereby improving production efficiency, reducing costs, and optimizing resource utilization.

**Keywords:** Improving Simulated Annealing Algorithm, No-Wait Flow Shop Scheduling, Local Search, Optimization Methods.

## 1. Introduction

No-Wait Flow Shop Scheduling is a special type of flow shop scheduling problem where parts can immediately move to the next machine for processing after completing their operations on one machine, without waiting for other parts to finish[1]. In traditional flow shop scheduling, parts need to be processed in a strict order on different machines, and the processing time for each part on each machine is continuous. However, in No-Wait Flow Shop Scheduling, parts can be processed in parallel on different machines, reducing the overall processing time. The advantage of this scheduling approach is that it can improve production efficiency and resource utilization by significantly reducing the total processing time of parts.

Most of the current research on this problem adopts other heuristic algorithms such as Improved Whale Algorithm, Particle Swarm Optimization, etc. However, there is still limited research on solving this problem based on the Simulated Annealing algorithm. Therefore, this paper mainly focuses on solving the No-Wait Flow Shop Scheduling problem from the perspective of improving the Simulated Annealing algorithm[2].

The paper primarily employs the Simulated Annealing algorithm approach. Firstly, the objective function is derived and the relevant constraints are determined based on the requirements of the No-Wait Flow Shop Scheduling problem. Then, a random initial solution sequence is generated according to the objective function. The algorithm calculates the corresponding processing time for this solution sequence by calling the function. A new solution sequence is obtained by swapping or reversing the order of operations. The algorithm then calculates the processing time for the new solution sequence in a similar manner. If the processing time for the new solution sequence is smaller, it is selected as the optimal solution sequence. Otherwise, the new solution is accepted or rejected according to the Metropolis criterion. This process is iterated until a certain number of iterations or no better solution is found. Finally, the obtained optimal solution sequence is outputted, and a Gantt chart is generated to visualize the scheduling plan.

## 2. Materials and Methods

### 2.1. Data Source and Preprocessing

Retrieve the completion time required for each individual part to be processed on each machine from the required database. Then, process the processing time for each part on each machine into a matrix format as shown in the following diagram. We need to first obtain the required processing time for each part on each machine. Then, for each machine, process its data into each row of the matrix, where each element represents the processing time of that part on each machine, and the preceding index indicates the number of the machine.The specific processing form is illustrated in Figure 1.

```
++++++++++++++++++++++++++++++
instance 1
6 4
0 5 1 7 2 3 3 4
0 2 1 8 2 9 3 10
0 4 1 1 2 5 3 6
0 6 1 3 2 2 3 4
0 10 1 15 2 11 3 20
0 11 1 16 2 12 3 5
++++++++++++++++++++++++++++++
instance 2
5 4
0 12 1 24 2 12 3 13
0 20 1 3 2 19 3 11
0 19 1 20 2 3 3 15
0 14 1 23 2 16 3 14
0 19 1 15 2 17 3 22
```

```
++++++++++++++++++++++++++++++
instance 3
7 7
0 692 1 581 2 475 3 23 4 158 5 796 6 542
0 310 1 582 2 475 3 196 4 325 5 874 6 205
0 832 1 14 2 785 3 696 4 530 5 214 6 578
0 630 1 214 2 578 3 214 4 785 5 236 6 963
0 258 1 147 2 852 3 586 4 325 5 896 6 325
0 147 1 753 2 2 3 356 4 565 5 898 6 800
0 255 1 806 2 699 3 877 4 412 5 302 6 120
++++++++++++++++++++++++++++++
instance 4
8 8
0 456 1 789 2 654 3 321 4 456 5 789 6 654 7 789
0 654 1 123 2 123 3 456 4 789 5 654 6 321 7 147
0 852 1 369 2 632 3 581 4 472 5 586 6 320 7 120
0 145 1 678 2 965 3 421 4 365 5 824 6 758 7 639
0 632 1 581 2 475 3 32 4 536 5 325 6 863 7 21
0 425 1 396 2 325 3 147 4 852 5 12 6 452 7 863
0 214 1 123 2 456 3 789 4 654 5 321 6 456 7 789
0 654 1 789 2 654 3 123 4 123 5 456 6 789 7 654
```

**Figure 1.** Case Study Presentation

### 2.2. Simulated Annealing Algorithm

Simulated Annealing Algorithm is a heuristic global optimization algorithm inspired by the formation of crystal structures during the annealing process in solid materials. It mimics the process of gradually cooling a metal at high temperatures, where the thermal expansion and contraction lead to the discovery of better structural states [3]. The main principle of the algorithm is analogous to the solid annealing process in statistical mechanics, where the particles inside gradually transition from a disordered state to an ordered state as the temperature increases and decreases, ultimately reaching an equilibrium state[4].At the same time, the algorithm incorporates a certain probability of accepting worse solutions, compared to traditional iterative algorithms, in order to escape local optima and eventually reach the global optimum. The steps of the algorithm are as follows:

1) Setting the initial temperature $Temp$ to a sufficiently high value, specifying the initial solution $w$, and defining the maximum number of outer loop iterations $outLoopNum$.

2) Setting the temperature reduction factor $\partial$ (typically between 0.8 and 0.99) and calculating the new temperature using the formula $Temp = \partial \times Temp$.

3) Generating a new solution using a specific generation method $w'$.

4) Calculating the cost difference $\Delta T = F(w') - F(w)$, where $F()$ is the evaluation function.

5) Accepting the new solution as the current solution if $\Delta T < 0$, or accepting $w'$ with a probability $e^{-\frac{\Delta T}{Temp}}$ according to the Metropolis criterion.

6) Repeating $inLoopNum$ times steps 3 to 5 at the current temperature.

7) Checking if the termination condition is met (no better solution obtained after consecutive temperature calculations or reaching *outLoopNum* (the maximum number of outer loop iterations)). If the condition is met, terminate the algorithm; otherwise, go back to step 2.

The algorithm flowchart is presented in Figure 2.

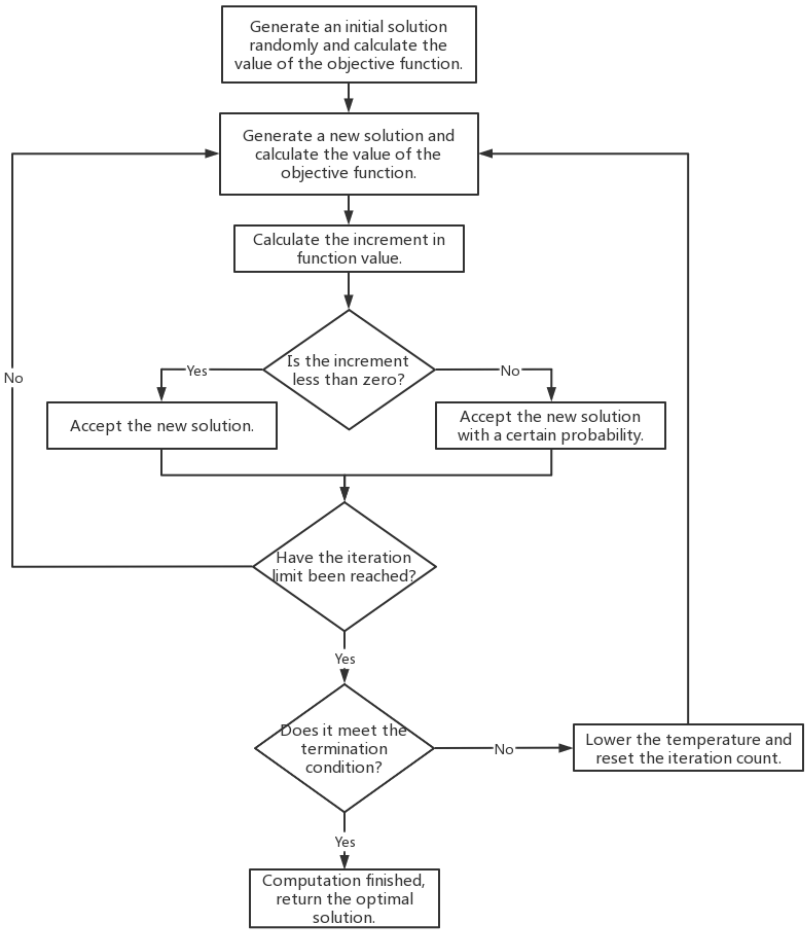



**Figure 2.** Algorithm Flowchart

## 3. Model Construction and Solution

### 3.1. Model Derivation

Before proceeding with the derivation, it is important to clarify a fundamental fact for the No-Wait Permutation Flow Shop Scheduling problem. The key requirement is that for any given job, its processing on adjacent machines should be continuous without any gaps or waiting periods. This means that once the start time of a job on one machine is determined, the processing times on other machines for that job are automatically determined as well[5-7] . Therefore, the focus should be on determining the start time of each job on each machine, which subsequently determines the processing times on other machines.

In order to solve for the start time of each job on the first machine, we can follow the steps below:

Let's denote $Begin_{i,j}$ as the start time of the $i$ th job on the $j$ th machine, $End_{i,j}$ as the completion time of the $i$ th job on the $j$ th machine, and $Ctime_{i,j}$ as the processing time of the $i$ th job on the $j$ th machine[8].

$$Begin_{1,1} = 0 \tag{1}$$

$$Begin_{i,j} = End_{i,j-1}(j = 2,\cdots,machqty) \tag{2}$$

$$End_{i,j} = Begin_{i,j} + Ctime_{i,j} \tag{3}$$

$$Begin_{i,j} = Begin_{i,j-1} + Ctime_{i,j-1}(j = 2,\cdots,machqty) \tag{4}$$

$$\because Begin_{i,j} \geq End_{i-1,j}(i = 2,\cdots, jobqty) \tag{5}$$

$$\therefore Begin_{i,j} \geq \begin{cases} End_{i-1,j}, k = 0 \\ End_{i-1,j+k} - \sum_{n=1}^{k} Ctime_{i,j+n-1}, k = 1,\cdots,machqty \end{cases} \tag{6}$$

$\because$ *To minimize the total completion time*

$\therefore$ *To minimize the value of* $Begin_{i,1}$ *as much as possible*

$$\therefore Begin_{i,1} = \max(End_{i-1,1}, \max_{k\in\{1,\cdots,machqty-1\}}(End_{i-1,1+k} - \sum_{n=1}^{k} Ctime_{i,n})) \tag{7}$$

### 3.2. Generating New Solutions

Generating New Solutions in Two Ways:

1) Swapping to Generate New Solutions: First, randomly select two indices in the old sequence that correspond to job numbers[9]. Then, swap the values of these job numbers in the sequence to create a new solution sequence.

2) Reversing to Generate New Solutions: First, randomly select two indices in the old sequence[10]. Then, reverse the order of job numbers between these two indices to update the sequence and generate a new solution sequence.

## 4. Result Analysis

Based on four case studies, the proposed solution using the Simulated Annealing algorithm has provided favorable solutions.Specific results will be shown in Figure[3-6].

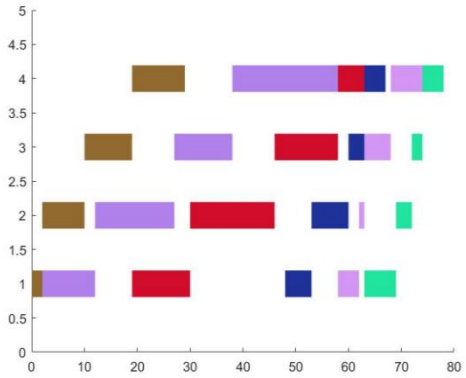



**Figure 3.** Gantt Chart for Case Study 1

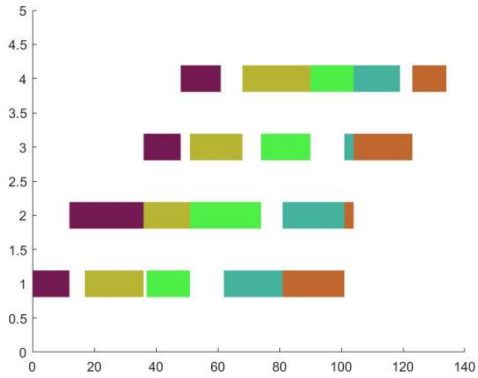



**Figure 4.** Gantt Chart for Case Study 2

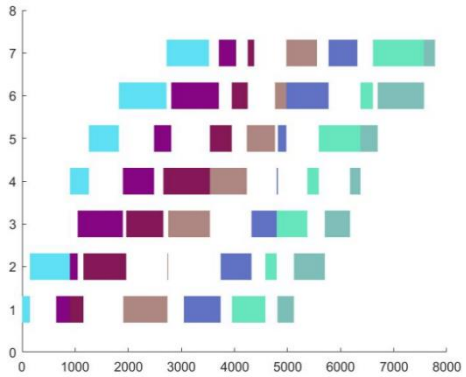



**Figure 5.** Gantt Chart for Case Study 3

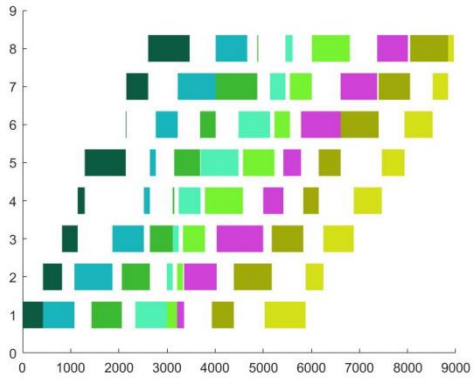



**Figure 6.** Gantt Chart for Case Study 4

Based on the visual representation of the Gantt chart, it is evident that our algorithm can effectively solve the No-Wait Flow Shop Scheduling problem. The Gantt chart clearly illustrates the scheduling plan and the allocation of jobs on different machines over time. It provides a visual confirmation that our algorithm has successfully optimized the scheduling process, minimizing the waiting time between jobs and ensuring a smooth flow of operations.

Furthermore, by analyzing the Gantt chart, we can observe the reduction in idle time and the efficient utilization of resources. The jobs are scheduled in a way that minimizes the total completion time, resulting in improved production efficiency and increased throughput. The optimized scheduling plan ensures that each job starts immediately after the previous one is completed on the preceding machine, eliminating unnecessary waiting periods and maximizing productivity.

## 5. Comparison Experiment

The initial parameter settings are shown in Table 1.

**Table 1.** Initial Parameter Values

| Parameters | Parameter values |
|---|---|
| *Temp* | 1000 |
| *inLoopNum* | 100 |
| $\partial$ | 0.8 |
| *noGoodLoop* | 100 |
| *outLoopNum* | 2000 |

Results and Running Time for different values of *Temp* are shown in Table 2.

**Table 2.** Results and Running Time for different values of *Temp*

| Parameter values | Results | Running Time |
|---|---|---|
| 10000 | 2772 | 3.4639 |
| 5000 | 2766 | 3.4008 |
| 1000 | 2766 | 3.3313 |
| 500 | 2766 | 3.3618 |
| 100 | 2766 | 3.3913 |
| 10 | 2766 | 3.4766 |

When the other parameters are kept the same, a larger value of *Temp* indicates a higher probability of accepting worse solutions. While this can help the algorithm escape local optima, it may also result in accepting too many worse solutions and impacting efficiency. In this case, we can

observe that when the value of *Temp* is set to 1000, good results can be achieved in a relatively short amount of time.

Results and Running Time for different parameter values of *noGoodLoop* are shown in Table 3.

**Table 3.** Results and Running Time for different parameter values of *noGoodLoop*

| Parameter values | Results | Running Time |
|---|---|---|
| 10000 | 2766 | 3.3139 |
| 5000 | 2766 | 3.3019 |
| 1000 | 2790 | 3.3273 |
| 500 | 2766 | 3.3266 |
| 100 | 2766 | 3.3814 |
| 10 | 2766 | 3.3747 |

In the case where other parameters remain the same, theoretically, as the value of *noGoodLoop* increases, more iterations and computational time are required. However, due to external constraints of the value of *outLoopNum*, it can be observed that the increase in running time becomes less significant beyond a certain value of *noGoodLoop*.

Modified Algorithm: The original algorithm was constrained by the maximum outer loop iterations and the value of *noGoodLoop*. Now, the termination condition is modified to stop the experiment when either the temperature decreases below 0.00001 or the number of iterations without generating an optimal solution reaches the value of *noGoodLoop*, and an experiment is conducted to compare the running results. (Other parameters remain the same) Results and Running Time for Original Algorithm and Modified Algorithm are shown in Table 4.

**Table 4.** Results and Running Time for Original Algorithm and Modified Algorithm

| Algorithm | Results | Running Time |
|---|---|---|
| Original Algorithm | 2766 | 3.3497 |
| Modified Algorithm | 2789 | 0.1449 |

Through comparison, it can be observed that the modified algorithm has significantly shorter running time compared to the original algorithm. However, the original algorithm provides superior and more stable results. This is because, in this case, the original algorithm has a sufficiently large number of outer loop iterations to achieve good results. On the other hand, the modified algorithm may terminate the loop earlier as it satisfies the condition of $Temp < 0.00001$ before completing a sufficient number of iterations, resulting in less optimal results.

## 6. Conclusion

### 6.1. Advantages of the Model

This study introduces the Simulated Annealing algorithm as a method to solve the No-Wait Flow Shop Scheduling problem. As a global optimization method, Simulated Annealing has the ability to search for optimal solutions within the search space. In the context of the No-Wait Flow Shop Scheduling problem, where multiple jobs can be processed in parallel, the solution space is complex and extensive.

Traditional local search algorithms may get trapped in local optima, resulting in imbalanced resource utilization or prolonged processing time. However, Simulated Annealing has the advantage of global search, allowing it to escape local optima and explore towards global optima. By incorporating the Simulated Annealing algorithm, it becomes possible to effectively search the solution space and find better scheduling solutions.

The focus of this study is the No-Wait Flow Shop Scheduling problem, which is more complex and challenging compared to traditional flow shop scheduling problems. The Simulated Annealing algorithm, through adjustments in job sequencing and machine allocation, leverages the advantages of parallel processing to reduce overall processing time. By optimizing for the no-wait condition, it can improve production efficiency and resource utilization.

### 6.2. Limitations of the Model

However, it should be noted that the number of test cases covered in this study is relatively small, which may have an impact on the results. Insufficient test cases may fail to capture potential issues or vulnerabilities within the system. To enhance the comprehensiveness and reliability of the testing, it is recommended to expand the test cases, including more boundary cases, exceptional situations, and diverse usage scenarios. This will provide a better assessment of the system's performance and stability, ensuring its smooth operation in various circumstances.

## References

[1] Wang, H. (2019). Optimization of Flow Shop Scheduling Problem Based on Intelligent Computing. Shanghai University of Engineering Science.

[2] Cai, G. (2022). Research on Flow Shop Scheduling Based on Improved Genetic Algorithm.Dalian University of Technology.

[3] Peng, L., Fang, L., Shen, C., et al. (2023). Integrated Textile Scheduling System Based on Simulated Annealing Algorithm. Software, 44(04), 6-10.

[4] Yang, Z., Liu, R., Tang, Y., et al. (2023). Research on Optimization Deployment of Early Warning Aircraft Based on Simulated Annealing Algorithm. Proceedings of the 2023 Annual Conference of Tianjin Electronic Industry Association, 139-143.

[5] Sun, Q. (2016). A Discrete Harmony Search Algorithm for Solving No-Wait Flow Shop Scheduling Problem. Journal of Binzhou University, 32(02), 70-74.

[6] Sun, Q., Gao, K., Dong, B. (2012). A Heuristic Algorithm for Optimizing No-Wait Flow Shop Scheduling Problem. Journal of Computer Applications Research, 29(06), 2072-2073+2101.

[7] Zhang, X. (2022). Research on Comprehensive Scheduling Hybrid Algorithm with No-Wait Operations. Harbin University of Science and Technology.

[8] Gan, L., He, M. (2018). A New Method for Support Vector Machine Training Based on Convex Optimization Theory. In Science and Engineering Research Center (Eds.), Proceedings of the 2018 2nd International Conference on Education, Management and Applied Social Science (EMASS 2018), 400-404.

[9] Wang, B., Li, T. (2012). Heuristic Algorithm for Permutation Flow Shop Scheduling with Limited Waiting Time. Journal of Management Sciences, 15(06), 22-32.

[10] Liu, C. (2014). Bat Algorithm for Solving No-Wait Flow Shop Scheduling Problem. Proceedings of the 18th Annual Conference of the Chinese Academy of Systems Science—A02 Management Science, 15-20.