Improved cat swarm optimization for permutation flow shop scheduling problem

Xiaobing Pei1, Yan Tang2*

1PhD supervisor, Tianjin University of Technology, School of management, China
2Postgraduate, Tianjin University of Technology, School of management, China
* Corresponding author: 1072755065@qq.com

Abstract: To minimize the maximum make span of permutation flow shop scheduling problem, this paper proposed a cat swarm optimization algorithm based on estimation of distribution algorithms. This article take the cat swarm optimization as the framework, embedded in the Estimation Distribution Algorithm, in the search mode, mining the excellent gene link in the solution sequence by using the probability matrix to combination blocks, speed and position tracking mode using characteristics of cat swarm optimization updates in the cat in order to update the excellent solution sequence generator group. Finally, through the simulation test and result comparison of Carlier and Reeves standard example set, the good robustness and global searching ability of the algorithm are verified.

1. INTRODUCTION
The Permutation Flow Shop Scheduling Problem (PFSP) is one of the core problems of intelligent manufacturing and has very important engineering application value. Research shows that this problem is NP-hard [1] (Non-deterministic Polynomial-time hard, NP). Common PFSP solving methods can be divided into: precise algorithm, heuristic algorithm [2] and so on. Heuristic algorithms can quickly find feasible solutions to problems, but the structure of these algorithms is relatively complex, and it is difficult to find the optimal solution.

Under limited resource conditions, the optimization of PFSP can effectively increase the profits of the enterprise. Related parties have been committed to research and development of efficient optimization technologies, hoping to quickly find the optimal solution of PFSP. Inspired by the daily behavior of cats, Shu-Chuan Chu [3] proposed Cat Swarm Optimization (CSO) in 2006. At present, the Estimation of Distribution Algorithms,(EDA) has been applied to many fields such as intelligent learning, function optimization, image recognition, and multi-objective planning [4].

2. ESTABLISH A MATHEMATICAL MODEL FOR THE REPLACEMENT FLOW SHOP SCHEDULING PROBLEM
PFSP is a simplified model of many production scheduling problems. This problem mainly studies the processing of n workpieces \( \{N_1, N_2, ..., N_n\} \) on m machines \( \{M_1, M_2, ..., M_m\} \), and each workpiece has m processing procedures. The goal is to find the optimal solution for a certain production index. The relevant constraints are as follows:

(1) All workpieces are processed on the machine in the same order, namely 1, 2, ..., m; (2) At a certain time, any machine can only process one workpiece; (3) at a certain time, any workpiece can only be
processed on one machine; (4) any workpiece cannot be interrupted during processing; (5) all workpieces can be processed at the initial moment. If the PFSP takes the maximum completion time as the goal, the problem can be expressed as \( n/m/prmu/ \max C \), where \( n \) represent the total number of workpieces, \( m \) represents the total number of machines, and \( prmu \) represents the same processing order of all workpieces on all machines. \( \max C \) representative maximum completion time.

The mathematical description of the above problem is as follows: Let \( t(\pi_i,j) \) represent the processing time of workpiece \( i \) on machine \( j \), and \( C(\pi_i,j) \) represent the completion time of workpiece \( i \) on machine \( j \). \( \Gamma_\pi = \{ \pi_1, \pi_2, ..., \pi_n \} \) represents a processing sequence of all workpieces, and \( \Gamma \) represents all sorted sets. The specific PFSP model can be expressed as:

\[
C(\pi_i, 1) = t(\pi_i, 1) \tag{1}
\]
\[
C(\pi_i, j) = C(\pi_{i-1}, j) + t(\pi_i, 1), \quad i = 2, 3, ..., n \tag{2}
\]
\[
C(\pi_i, j) = C(\pi_{i-1}, j) + C(\pi_{j-1}, j), \quad j = 2, 3, ..., m \tag{3}
\]
\[
C(\pi_i, j) = \max\{ C(\pi_{i-1}, j), C(\pi_i, j-1) \} + t(\pi_i, j), \quad i = 2, ..., n; \quad j = 2, ..., m. \tag{4}
\]
\[
C_{\max} = C(\pi_n, m) \tag{5}
\]

The goal of this problem is to obtain the optimal workpiece processing sequence \( \pi^* \), so for any other workpiece processing sequence, there are:

\[
C_{\max}(\pi^*) \leq C_{\max}(\pi). \tag{5}
\]

3. Improved catswarm optimization

The specific process of the block-based catgroup algorithm for solving PFSP is shown in Figure 1:

3.1 Initial population

The traditional cat group algorithm generates the initial population by way of roulette. Due to the low individual fitness of the initial population, the convergence speed of the algorithm is restricted to a certain extent. This study uses the greedy criterion\(^5\) to initialize the population by combining search and roulette, so as to accelerate the convergence speed and increase the diversity of the initial solutions. When using the greedy criterion to initialize the population, randomly select the first workpiece \( i \) and add it to \( \Gamma_\pi = \{ \pi_1, \pi_2, ..., \pi_n \} \), and then search among other unprocessed workpieces to find the next
workpiece, which has the shortest processing time among all unprocessed workpieces. Add to \( \Gamma_\varepsilon = \{ \pi_1, \pi_2, \ldots, \pi_n \} \), and use it as the current processing workpiece, continue to search and add the next processing workpiece \( \Gamma_\varepsilon = \{ \pi_1, \pi_2, \ldots, \pi_n \} \), until all the workpieces are added to the processing sequence set. The initial solution sequence is generated by using the greedy criterion. Therefore, the initial solution sequence needs to be transformed into a position vector in a certain interval. Specifically as formula (6):

\[
x_{i,j} = x_{min,j} + \frac{x_{max,j} - x_{min,j}}{n} \cdot (s_{i,j} - 1 + r_j) \quad j = 1, 2, \ldots, m
\]

Among them, \( x_{i,j} \) represents the position value of the cat in the j dimension; \( x_{min,j} \) represents the j-th dimension workpiece number of the initial solution sequence; and \( x_{max,j} \) represents the upper and lower bounds of the position vector of the cat in the continuous space; \( r_j \) represents a random number generated randomly in the interval \([0, 1]\).

The initial position and speed generation method is shown in the following formula:

\[
X_i = X_{min} + (X_{max} - X_{min})r_1
\]

\[
V_i = V_{min} + (V_{max} - V_{min})r_2
\]

In the above formula, \( X_i \) continuous changes in the \([X_{min}, X_{max}]\) interval, \( V_i \) continuous changes in the \([V_{min}, V_{max}]\) interval.

### 3.2 Cat behavior mode selection based on mixing ratio

The traditional cat swarm algorithm cannot reasonably allocate the proportion of local search and global search according to the number of iterations of the algorithm. If a larger mixing ratio is used to track cats in the early stage of the algorithm iteration, the global search capability of the algorithm can be increased, but the search will be made in the later stage of the algorithm iteration. The proportion of cats is larger, which can improve the search accuracy and convergence. Therefore, this study adopts a cat behavior mixture ratio selection method in the literature [10], as shown in Figure 2.

![Fig. 2. The allocation of mixed ratio](image)

The calculation formula of the linear mixing ratio is:

\[
MR = MR_i + \frac{(MR_2 - MR_1) \times T}{T_0}
\]

Among them, \( T \) represents the number of iterations; \( T_0 \) represents the maximum number of iterations.

### 3.3 Search mode

#### 3.3.1 Build a probabilistic model

The fitness function value of the initial solution is sorted from small to large, and the first excellent solution \( s \) is selected to form an excellent solution set \( \partial = \{ \Gamma_1, \Gamma_2, \Gamma_3, \ldots, \Gamma_s \} \), and the position matrix and the dependent matrix are updated as follows:


\[ B_{ij}^k = \begin{cases} 
1 & \text{if } i \text{ in the } j \\
0 & \text{otherwise} 
\end{cases} \]  

\[ i = 1, 2, \ldots, n; \ j = 1, 2, \ldots, m; k = 1, 2, \ldots, s \]  

\[ T_{ij}(t) = T_{ij}(t-1) + \sum_{k=1}^{s} B_{ij}^k, \]  

\[ i = 1, 2, \ldots, n; \ j = 1, 2, \ldots, m; k = 1, 2, \ldots, s \]  

\[ Y_{ij} = \begin{cases} 
1 & \text{If the workpiece } i \text{ is next to } l \\
0 & \text{otherwise} 
\end{cases} \]  

\[ i = 1, 2, \ldots, n; \ l = 1, 2, \ldots, m; k = 1, 2, \ldots, s \]  

\[ T_{ij}(t) = T_{ij}(t-1) + \sum_{k=1}^{s} Y_{ij}^k, \]  

\[ i, l = 1, 2, \ldots, n; j = 1, 2, \ldots, m; k = 1, 2, \ldots, s \]  

The details are shown in Figure 3 and Figure 4.

![Fig.3. Updating method of position matrix](image_url)

![Fig.4. Updating method of dependency matrix](image_url)

### 3.3.2 Combination block

In order to find blocks with high competitive advantages, this research combines blocks from two steps: block mining and block competition.

![Fig.5. The comparison of complexity](image_url)

1. Block mining

   According to the relevant information provided by the location matrix and the dependency matrix model, perform corresponding block mining. The starting position of the block is selected in a random manner, and a blank block with the minimum length is generated, and the minimum length of the block is set to 3.

   Put the most suitable workpiece in the blank area. For the convenience of calculation, the cumulative result of each workpiece in the position matrix and the dependent matrix is converted into probability.
At the same time, the probabilities of the two matrices are integrated into a combined probability (Combination Probability, CP), using roulette to select the merge probability. The calculation of the merging probability is shown in formula 14, where $i$ represents the workpiece code, $j$ represents the immediately preceding workpiece number, $k$ represents the location of the workpiece $i$, $n$ represents the total number of workpieces, $CP_{i,k}$ represents the merging probability of the workpieces $i$, $W_{dom}$ and $W_{dep}$ represents the merging weights of the current position matrix and the dependent matrix respectively. The value $P_{i,k}^{dom}$ is the probability that the workpiece $i$ is in the position in the position matrix $k$, and $P_{i,j}^{dep}$ is the probability that the workpiece $j$ is immediately before the workpiece $i$ in the dependency matrix. (Exchange the positions of $k$ and $j$)

$$CP_i = (W_{dom} \times P_{i,k}^{dom}) + (W_{dep} \times P_{i,j}^{dep})$$

$$i, j, k = 1, 2...n; i \neq j$$

(14)

Calculate the merging probability of all artifacts, and use roulette to select the second artifact and other artifacts in the block, as shown in Figure 6.

![Diagram](image)

**Fig. 6.** Mining blocks by combining probability

As the block length increases, the total probability gradually decreases, that is, the greater the probability of error. In order to ensure that the quality of the block does not decrease with the increase of the block length, a threshold is designed to screen the above-mentioned mined blocks. The threshold increases from 0.24 to 0.8 as the number of iterations increases. The blocks that meet the threshold are temporarily stored in the block library, and all the blocks that meet the threshold in this iteration are reserved in the block library.

2. Block competition

After the above block mining is completed, the block competition is used to select the block with greater competitiveness.

The calculation method of the average probability is shown in Equation 15:

$$P_{AVG}^B = \frac{P_{i,k}^{dom} + \sum_{l=2}^{n} CP_{i,k}}{n}$$

(15)

Among them, $i$ is the block number, $B_1^i$ is the first workpiece in the first block, $k$ is the position of, and $n$ is the current block length.

The specific block competition is shown in Figure 6. The newly generated 2 blocks, the block library
\{D_1, D_2\} contains 3 blocks \{D_3, D_4, D_5\}, the newly generated block \(D_1\) and the recurring artifact \(D_6\) in the block library \(D_3\), the average of the pair \(D_1\) and \(D_3\). Comparing the probabilities, the average probability of \(D_6\) is 0.26, which is greater than the average probability \(D_3\) of 0.23, so \(D_1\) instead of \(D_3\) entering the block library, \(D_3\) is eliminated and deleted. \(D_2\) overlaps with \(D_4\) at position 11, and the average probability of comparing \(D_2\) with \(D_4\), \(D_2\) is higher than that from the figure \(D_4\), so \(D_4\) is eliminated. No workpiece or position duplication occurs with any block, so it continues to be stored in the block library, as shown in Figure 7.

Fig 7 The competition of blocks

### 3.3.3 Combined artificial solution

In order to improve the quality of the solution sequence, after completing the block mining, use the blocks reserved in the block library to combine manual solutions. The specific steps are as follows:

Step 1: Start mining from the first position of the manual solution. The mining method is the same as the above block mining. The first position uses the probability in the position matrix to select the workpiece in a roulette way;

Step 2: The remaining N-1 positions are selected in the form of roulette for the combined probability;

Step 3: Each time a workpiece is selected, compare the workpiece with the starting positions of all other blocks. If the positions and workpieces of other blocks are the same, copy this block directly to the manual solution, and then the next position continues to be selected and compared until the manual solution combination is completed.

### 3.4 Tracking mode

The cat tracking mode is to make the individual close to the global optimal solution. In this mode, the cat is compared with the optimal position of the group to update the speed and position of the individual \[^7\]. The tracking mode can be carried out according to the following 2 steps.

1) Speed update

At any time, every cat has a speed, and the current speed of the i-th cat can be expressed as:

\[ V_i = \{v_{i1}, v_{i2}, \ldots, v_{in}\} \]

The speed of all cats is updated according to the following formula:

\[ V_i(n+1) = V_i(n) + w \times c \times \text{rand} \times [X_{\text{opt}(n)} - V_i(n)] \]  \hspace{1cm} (16)

\[ w = w_{\text{max}} - (w_{\text{max}} - w_{\text{min}}) \times \left(\frac{t}{t_{\text{max}}}\right) \]  \hspace{1cm} (17)

Among them, \(V_i(n+1)\) represents the speed of the i cat after the update; \(w\) is the inertia weight, and the size of \(w\) is determined by Formula 17; \(t\) represents the iteration information; \(t_{\text{max}}\) represents the maximum number of iterations; \(c\) represents the acceleration constant; \(\text{rand}\) obeys \([0, 1]\) uniform distribution.
(2) Location update
The position update of each cat is determined by the following formula
\[ X_i'(n+1) = X_i'(n) + V_i'(n+1) \]  \hspace{1cm} (18)

Among them, \( X_i'(n+1) \) represents the updated position of the i cat.

(3) If the new position of the i-th cat \( X_i'(n+1) = X_i'(n) + V_i'(n+1) \) is beyond the search space, multiply the speed by -1 and continue searching from the opposite direction.

(4) The binary race method is used to update the solution sequence, that is, the fitness of the parent population and the subgroup are compared in pairs, and the subgroup with the highest fitness replaces the parent population for the next search.

4. Simulation test

4.1 Parameter settings
In order to verify the performance of the block-based cat swarm algorithm, the test data of this research adopts Carlier’s Car class benchmark examples and tests, apply the algorithm to solve these cases, and compare with algorithms in other documents. This research program uses C++. Parameter design: The number of simulation tests is 20, the number of initial populations is 100, and the maximum number of iterations is 100. This study compares the best relative error (BRE), average relative error (ARE) and worst relative error (WRE) of each algorithm. Among them, \( C^* \) represents the optimal solution of a known calculation example, \( C_{min} \) represents the optimal value solved, \( C \) represents the average value solved, and \( C_{max} \) represents the worst value solved.

\[ BRE = \left( \frac{C_{min} - C^*}{C^*} \right) \times 100\% \]  \hspace{1cm} (19)

\[ ARE = \left( \frac{\overline{C} - C^*}{C^*} \right) \times 100\% \]  \hspace{1cm} (20)

\[ WRE = \left( \frac{C_{max} - C^*}{C^*} \right) \times 100\% \]  \hspace{1cm} (21)

4.2 Experimental test and comparison
Aiming at Carlier’s example, this research will be based on the improved cat swarm algorithm (EDA-CSO) and cat swarm algorithm (CSO) [9], standard particle swarm algorithm (PSO) [9], bat algorithm (BA) [9] based on the distribution estimation algorithm Compare. The specific comparison results are shown in Table 1.

| Examples | n, m | \( C^* \) | BRE | ARE | WRE | BRE | ARE | WRE | BRE | ARE | WRE |
|----------|------|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Car1     | 11, 5| 7038        | 0.00| 0.01| 0.20| 0.00| 0.02| 0.27| 0.00| 0.52| 2.72|
| Car2     | 13, 4| 7166        | 0.00| 1.37| 5.00| 0.00| 2.76| 6.01| 0.00| 5.22| 10.8|
| Car3     | 12, 5| 7312        | 0.00| 1.85| 3.17| 0.00| 2.03| 3.86| 0.00| 3.77| 9.04|
| Car4     | 14, 4| 8003        | 0.00| 0.37| 4.98| 0.00| 0.44| 5.25| 0.00| 3.86| 6.55|
| Car5     | 10, 7| 7720        | 0.00| 0.37| 1.09| 0.00| 0.29| 1.31| 0.00| 1.24| 2.12|
| Car6     | 8, 9 | 8505        | 0.00| 0.52| 2.12| 0.00| 0.77| 2.47| 0.00| 2.06| 5.7 |
| Car7     | 7, 7 | 6590        | 0.00| 0.15| 2.03| 0.00| 0.24| 2.47| 0.00| 1.61| 4.51|
| Car8     | 8, 8 | 8366        | 0.00| 0.09| 1.01| 0.00| 0.29| 1.33| 0.00| 3.85| 8.88|

Tab.1 Comparison of test results for EDA-CSO、CSO、PSO and BA
It can be seen from Table 1 that these four algorithms can find the optimal solution to the problem in most cases (BRE is 0). In addition, EDA-CSO is obviously smaller than the ARE and WRE of CSO, PSO, BA. At the same time, due to the adjustment of the mixing ratio of cats, a larger mixing ratio is used to track cats in the early stage of the iteration, which increases the algorithm’s global search capabilities. In the later stage of the iteration, the proportion of search cats is relatively large, which improves the search accuracy and convergence. In the later stage, the EDA-CSO algorithm is obviously better than other algorithms.

5. In conclusion
This research is based on the improvement of the cat group algorithm, and proposes an improved cat group algorithm based on the distribution estimation algorithm to solve the replacement flow shop scheduling problem. In the cat search stage, the greedy criterion is used to initialize the population to speed up the convergence; the location matrix and the dependent matrix are used to mine blocks; the block competition is used to generate artificial solutions; different mutation methods are used in different evolution stages to improve the artificial solution. Finally, the experimental results of each algorithm are compared to verify the effectiveness and robustness of the algorithm.

references
[1] Garey M R, Johnson D S, Sethi R. The Complexity of Flowshop and Jobshop Scheduling[J]. Mathematics of Operations Research, 1976, 1(2): 117-129.
[2] X B Li, Yan Bai, L X Geng. Improved genetic algorithm to solve the replacement flow shop scheduling problem[J]. Journal of Computer Applications, 2013, 33(12): 3576-3579.
[3] CHUS C, TSAIP W, PAN J S. Cat swarm optimization[J]. Lecture Notes in Computer Science, 2006, 4099: 854-858.
[4] TZENG Y RG, CHEN C L. A Hybrid EDA with ACS for Solving Permutation Flow Shop Scheduling[J]. The International Journal of Advanced Manufacturing Technology, 2012, 60(9/12): 1139-1147.
[5] Kim D, Haldar J P. Greedy Algorithms for Nonnegativity-Constrained Simultaneous Sparse Recovery[J]. Signal Processing, 2016, 125(C): 274-289.
[6] Chang P C, Chen M H. A block based estimation of distribution algorithm using bivariate model for scheduling problems[J]. Soft Computing, 2014, 18(6): 1177-1188.
[7] Rautray R, Balabantaray R C. Cat swarm optimization based evolutionary framework for multi-document summarization[J]. Physica A Statistical Mechanics & Its Applications, 2017, 477: 174-186.
[8] Carlier, Jacques. Ordonnancements à contraintes disjonctives[J]. RAIRO - Operations Research, 1978, 12(4): 333-350.
[9] B X Ma, C M Ye. Using cat swarm algorithm to solve flow shop scheduling problem[J]. Modern Manufacturing Engineering, 2014, (06): 12-15+71.