Assembly Sequence Planning Based on Discrete Bat Algorithm

CHEN Yuejun¹,a, ZHANG Li²,b and HAN Wenjie³,c

¹Lanpec Technologies Limited, Shanghai 201518, China
²Institute of Mechanical Engineering, Lanzhou University of Technology, Lanzhou 730050, China
³Institute of Mechanical Engineering, Lanzhou University of Technology, Lanzhou 730050, China

Keywords: assembly sequence planning; bat algorithm; fitness function; orthogonal test

Abstract. A Discrete Bat Algorithm (DBA) is proposed to solve the assembly sequence planning (ASP) problem. Several key technologies including the position and velocity of Bat Algorithm, corresponding operators for updating the position and the position of local search are redefined. The fitness function is established based on the geometric feasibility which is quantified, the stability, the polymerization for assembly sequence and the frequency of direction changes. The performance of the DBA is investigated through a case study of a typical assembly which contains 10 parts. The near-optimal parameters of the algorithm are determined by the combination of orthogonal test (initial positioning) and control variables (precise positioning). It is proved that the DBA, compared with Particle Swarm Optimization (PSO) which is applied frequently in assembly sequence planning field, is more effective.

Introduction

Manufacturing companies are under pressure continuously from global competitors in their product development process, which has forced them to speed up the time to market while minimizing costs, thus ensuring to stay competitive [1]. Assembly as an important manufacturing process, consumes up to 50% of total production time and accounts for more than 20% of total manufacturing [2]. Research in its optimization, can quickly speed up the assembly process and reduce consumption. Assembly sequences planning (ASP) problem is essentially NP-hard (non-deterministic polynomial-hard) problem [3], which is a typical combinatorial explosion problem with the increase in the number of components in products. The research achievements of intelligent optimization algorithms have been outstanding, which has given a new way to solve the ASP problem over the years. Bonneville F et al. [4] proposed to apply genetic algorithms (GA) to ASP problems; however, mutations and crossovers are likely to lose some features of the sequence. JM Milner et al. [5] proposed to apply simulated annealing algorithm (SA) to ASP problem; however, it has poor expansion of search space and finds the most effective area difficultly. JF Wang et al. [6] proposed to apply the ant colony optimization (ACO) to the ASP problem; however, it needs to specify the basic components in the assembly sequence planning, select the parameters in the formula difficultly, and the convergence speed is poor [7]. Wang Song et al. [8] proposed a hybrid frog leaping algorithm to solve the ASP problem. The algorithm has better global convergence ability, but the convergence rate of the algorithm is too slow in the later stage.

Bat-inspired Algorithm (BA) [9] is one of the newest swarm-intelligence-based algorithms. In recent years, the biological type of BA as a more advanced heuristic algorithm is widely used to solve a large number of different types of optimization problems, including engineering design, image processing, feature selection, path planning, etc. [10], especially scholars Yassine Saji, such as [11] be used to solve the traveling Salesman Problem (Travelling Salesman Problem, TSP) such a typical NP hard problem, but few scholars use them to solve ASP problems. In this paper, a discrete bat algorithm for ASP problem is proposed, the influence of its parameters is analyzed, and the near optimal parameters are determined. The optimal assembly sequence of assembly examples is obtained by using this algorithm, and a comparison test with particle swarm optimization (PSO) algorithm is carried out.
Bat-inspired algorithm

In 2010, Yang X.-S.et al. [9], a professor at Cambridge University, proposed a new intelligent optimization algorithm, the bat algorithm (BA), by simulating the echolocation behavior of bats during foraging. The basic BA formula is as follows [9]:

First, initialize the bat population: the updated formulas for position $x$, velocity $v$, frequency $f$, bat new position $x'_t$ and new velocity $v'_t$ at $t$ are as follows

$$f_i = f_{\text{min}} + (f_{\text{max}} - f_{\text{min}}) \beta$$  \hspace{1cm} (1)

$$v'_t = v^{-1}_t + \left(x^{-1}_t - x_*\right) f_i$$ \hspace{1cm} (2)

$$x'_t = x^{-1}_t + v'_t$$ \hspace{1cm} (3)

Where $\beta \in [0,1]$ is a random vector and $x_*$ is the current global optimal position.

Secondly, if (rand > $r_i$), a solution is obtained from the optimal solution set and a local solution is formed near it. The update formula for each bat in a local search is as follows:

$$x_{\text{new}} = x_{\text{old}} + \epsilon A'$$ \hspace{1cm} (4)

Where $\epsilon \in [-1,1]$ is a random number and $A' = <A'_i>$ is the average loudness of all bats in this generation.

Third, the loudness $A_i$ and the pulse emission rate $r_i$ are updated as the iteration proceeds. If the random number is less than the loudness $A_i$ and $f(x_i) < f(x_*)$, the updating formula of the loudness $A_i$ and the pulse emission rate $r_i$ is as follows:

$$A^{t+1}_i = \alpha A'_i - 1$$ \hspace{1cm} (5)

$$r^{t+1}_i = r_i^0 \left[1 - e^{-\gamma t'}\right]$$ \hspace{1cm} (6)

Where $\alpha$ and $\gamma$ are constants.

Evaluation indexes and objective function for ASP

In order to save cost and reduce assembly time, several factors which have great influence on product assembly are selected as evaluation indexes, including the geometric feasibility, the stability, the polymerization for assembly sequence and the frequency of direction changes. The fitness function is constructed after several indexes are weighted.

The geometric feasibility. The interference matrix was first proposed by Dini and Santochi [12] in assembly planning which can be obtained from geometric assembly relations. For an assembly of $n$ parts, $P = \{P_1, P_2, \ldots, P_n\}$, the interference matrix is $I$, where $I_{ijkl}$ represents the interference value of the $j$th part in the assembly direction of the component $d_k$ along the $k$ direction with the $i$th part. The judgment is as follows: $I_{ijkl} = 0$, indicating that the $j$th part has no interference with the $i$th part in the direction of $d_k$; $I_{ijkl} = 1$, indicating that the $j$th part interferes with the $i$th part in the direction of $d_k$, in which $\{d_k \in \{\pm X, \pm Y, \pm Z\}\}$. The interference between the part $P_j$ and the part $P_i$ along the direction of $d_k$ assembly is the same as that the part $P_1$ and the part $P_1$ along the direction of $-d_k$, namely $I_{ijkl} = I_{ij-d_k}$, it is only necessary to consider the interference matrix along the positive direction of the three axes. Let $Z_k(P_i)(k \in \{1,2,\ldots,6\})$ be the sum of the interference values of the parts $P_i$ along the $d_k$ direction and the assembled parts. The expression is as follows:


Whether the part $P_i$ can be assembled along the $d_k$ direction is as follows: $Z_k(P_i)=0$ means that the part $P_i$ can be assembled in the $d_k$ direction; $Z_k(P_i)\neq 0$ means that the part $P_i$ cannot be assembled in the $d_k$ direction.

For any assembly sequence $\{P_1, P_2, \ldots, P_n\}$, let $Z_g$ be the number of parts for which this sequence cannot be assembled. If there is no feasible assembly direction for part $P_i$, then $Z_g+1$. Therefore, $Z_g$ can be used as a quantitative indicator to measure the geometric feasibility.

**The stability for assembly sequence.** The reliability of assembly operation and the complexity of fixture and tool are embodied in the stability of operation which is composed of the stability of gravity direction of assembly and the connection relationship between parts. To quantify the stability of the assembly sequence, a connection matrix $C$ is established. $C_{ij}$ represents the connection relationship between the parts $P_i$ and $P_j$. The judgment basis is as follows: $C_{ij}=0$, $P_i$ and $P_j$ do not have a connection relationship; $C_{ij}=1$, $P_i$ and $P_j$ are connected stably $P_j$ is stable with fixtures; $C_{ij}=2$, $P_i$ and $P_j$ are stably connected and $P_i$ is stable with fixtures. $P_j$ is stable on $P_i$ under gravity. A stable connection refers to the connection of a part with a forced constraint, such as a interference connection or a fastener connection between the shaft of a hole. The quantitative expression of the stability index $Z_c$ is as follows:

$$Z_c = \sum_{i=2}^{n} U_i$$

Where $n$ is the total number of assembly parts and $U_i$ is the assembly stability of part $P_i$. If $C_{ij}$ $(1 \leq j \leq i-1)$ contains 2, $U_i=2$; if $C_{ij}$ $(1 \leq j \leq i-1)$ does not contain 2 but contains 1, then $U_i=1$; If $C_{ij}$ $(1 \leq j \leq i-1)$ only contains 0 element, then $U_i=0$, obviously $0 \leq Z_c \leq 2n-2$.

**The polymerization for assembly sequence.** The aggregation is usually measured by the number of assembly tool changes. The set assembly tool $P_i$ is set to $T(P_i)$ and the assembly tool is changed to $Z_t$. For a given assembly sequence $\{P_1, P_2, \ldots, P_m, P_{m+1}, \ldots, P_n\}$, if $\prod_{i=1}^{m} T(P_i) = T(P_i)$, when assembling $P_1, P_2, \ldots, P_m$, assembly tool does not change; if $\prod_{i=1}^{m} T(P_i) = T(P_i)$, and $\prod_{i=1}^{m+1} T(P_i) = T(P_{m+1})$, the assembly tool needs to be changed once when assembling the part $P_{m+1}$, then $Z_t+1$.

**The frequency of direction changes.** According to the interference matrix, the feasible assembly direction of parts $P_i$ is as follows:

$$D(P_i) = \{d_k | Z_k(P_i) = 0\}$$

For any feasible assembly sequence $\{P_1, P_2, \ldots, P_m, P_{m+1}, \ldots, P_n\}$, solving the minimum number of assembly direction change $Z_d$ assembly sequence, if $\prod_{i=1}^{m} D(P_i) = \Phi$, the direction of assembly is not changed when assembling $P_1, P_2, \ldots, P_m$; If $\prod_{i=1}^{m} D(P_i) \neq \Phi$, and $\prod_{i=1}^{m+1} D(P_i) = \Phi$, the assembly parts $P_{i+1}$ need to change a direction, then $Z_d+1$.

**Objective function structure.** The objective function is determined by weighting the above evaluation indexes:

$$F = \omega_g Z_g + \omega_t Z_t + \omega_d Z_d + \omega_c Z_c$$
In which, \( \omega_g \), \( \omega_c \), \( \omega_d \), \( \omega_t \) are the evaluation weight coefficients. Obviously, the lower the F value, the better the assembleability is.

The operation and steps of discrete Bat algorithm (DBA)

The DBA formula is not suitable for assembly sequence planning. In this paper, a discrete bat algorithm formula is proposed to solve the problem.

**Key technologies of DBA solution to ASP.** In the ASP problem, the DBA related operation is as follows:

- **Redefinition of position and velocity.** Each bat’s position is defined as an initialized \( n \)-dimensional vector. Each bat’s position corresponds to an assembly sequence. The \( i \)th is \( X_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T, x_{ik} \in \{1, 2, \ldots, n\} \), where \( n \) is the total number of parts and \( x_{ik} \) is the part number. The bat speed \( V_i \) is defined as a sequence for adjusting the transformation order of parts, that is, \( V_i \) stores switching sequences.

- **Subtraction operation between positions.** The speed is reduced between positions, and the expression is as follows:

\[
\text{Sub}_{xx} = X_j - X_i \tag{11}
\]

Define the number of differences between the two assembly sequences as \( r \) (initial value \( r = 0 \)), and compare the sequence elements \( X_i \) and \( X_j \) correspondingly. For the \( k \)th \( (1 \leq k \leq n) \) element, if \( x_{ik} \neq x_{jk} \) then \( r + 1 \), and \( \text{Sub}_{xx} = \{(s_{xm}, s_{jm}) | s_{xm} = x_{ik}, s_{jm} = x_{jk}, m \in [1, r]\} \), otherwise, no record. That is, \( \text{Sub}_{xx} \) inherits valid elements from \( X_i \), \( X_j \).

- **Addition operation of velocity \( V_0 \) and velocity \( \text{Sub}_{xx} \).** Let bat initial speed \( V_0 \), bat frequency \( f_i \), definition velocity \( V_0 \) and velocity \( \text{Sub}_{xx} \) get a new speed \( V_i \), the specific method is as follows: For \( V_0 \), \( \text{Sub}_{xx} \) merge \( V_i = \{V_0, \text{Sub}_{xx}\} \), choose 1 row value randomly in \( V_i \) \((l=\text{round} (\text{step} \times n \times f_i)+1, \text{round means rounding, step} > 0 \text{ means step size, } n \text{ means total number of parts})\) to constitute \( V_i \), and it is easy to know that \( V_i \) is a matrix of 1 row and 2 columns. After this operation the update of \( V_i \) can be completed.

- **Addition operation of position \( X_i \) and velocity \( V_i \).** The \( k \)th \( (1 < k < l) \) row element \((v_1, v_2) (v_1 \in [1, n], v_2 \in [1, n])\) of \( V_i \) is used as the exchange sequence number, that is, the position of the \( v_1 \)th element and the \( v_2 \)th element in the exchange assembly sequence \( X_i \). The addition of position \( X_i \) and velocity \( V_i \) completes the exchange of elements in \( X_i \) one by one with each element in \( V_i \) as an exchange order, thereby realizing the update of position \( X_i \).

- **Update of local position.** The local position updating adopts the reverse order arrangement of the assembly sequence subsequence [13], and randomly arranges any subsequence in the assembly sequence \( X_i \), which enhances the population diversity and accelerates the convergence speed. After the above operation, the core formulas Eq.2, Eq.3, Eq.4 of bat algorithm can be discretized.

**Operational steps of DBA.** The steps of applying the DBA in ASP in this paper are as follows:

**STEP1.** Get the total number of assembly parts, assembly information matrix; initialize bat population number \( \text{Popsize} \), DBA parameters pulse emission loudness \( A_i \), pulse rate \( r \), their update parameters \( \alpha \) and \( \gamma \), and algorithm maximum iteration number \( N_{\text{max}} \). The evaluation coefficient \( \omega_g \), \( \omega_c \), \( \omega_d \), \( \omega_t \) of fitness function is given. Randomly generate the position velocity of the initial population, calculate the fitness function value \( F_n \), and find the assembly sequence \( x \)-with the least fitness.

**STEP2.** Determine whether to satisfy the algorithm's end condition. If so, stop iteratively changing to step 8. Otherwise, go to step 3.

**STEP3.** According to the operation method above, follow the new \( X_i \) and \( V_i \).

**STEP4.** To determine whether rand > \( r \) is true or not, the assembly sequence subsequence is reversed and a new sequence is formed, otherwise, the assembly sequence is turned to step 5.
STEP5. Calculate the $X_{\text{new}}$, fitness value $F_{\text{new}}$, and determine if $F_{\text{new}} < F_n$ and rand $< A_i$ are both true. If it is established, then let $X_i = X_{\text{new}}$ and update $r_i$, $A_i$ according to Eq.5 and Eq.6, otherwise, go to step 6.

STEP6. When all assembly sequences complete an iteration, update the current best assembly sequence.

STEP7. Increase the number of iterations and switch to step 2.

STEP8. Output the relevant information of the optimal assembly sequence: assembly sequence, assembly direction, fitness function value.

Where, rand is a random number distributed uniformly on [0, 1].

Case study and analysis

In order to verify the validity of the algorithm, a test program is written in MATLABR2012a. The CPU model of the program is Intel Core i5 430, the main frequency range of CPU is 2.27^-2.53 GHz, the memory is 6 GB, and the Windows8 64-bit operating system is used. The assembly sequence planning of top drive blowout preventer containing 10 parts (Figure 1 and assembly tool table as shown in Table 1) is taken as an example. The weighting factor for this assembly fitness function was evaluated as $\omega_g = 0.5$, $\omega_c = 0.25$, $\omega_d = 0.15$, $\omega_t = 0.1$.

![Figure 1 An assembly consisting of 10 parts](image)

![Table 1 Tool type of each part in the assembly](table)

<table>
<thead>
<tr>
<th>Part no.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool type</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Test 1: the effects of population size. In this paper, the relationship between the optimal fitness of the assembly sequence and the number of the population is analyzed by tests on the population size of 200, 300, 400, 500 and 600 when the parameters of the algorithm are the same. The results of the algorithm are as shown in Table 2, and the optimal assembly sequence is as shown in Table 3.

![Table 2 Results of 50 times for different population size](table)

<table>
<thead>
<tr>
<th>Population size</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Number of executions</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Average running time [s]</td>
<td>4</td>
<td>12</td>
<td>25</td>
<td>39</td>
<td>60</td>
<td>83</td>
</tr>
<tr>
<td>Average optimization algebra.</td>
<td>17</td>
<td>53</td>
<td>95</td>
<td>135</td>
<td>214</td>
<td></td>
</tr>
<tr>
<td>Maximum fitness</td>
<td>5.35</td>
<td>5.1</td>
<td>4.95</td>
<td>4.85</td>
<td>4.85</td>
<td>4.85</td>
</tr>
<tr>
<td>Optimal fitness</td>
<td>4.85</td>
<td>4.85</td>
<td>4.85</td>
<td>4.85</td>
<td>4.85</td>
<td>4.85</td>
</tr>
<tr>
<td>Optimal probability of occurrence [%]</td>
<td>4</td>
<td>20</td>
<td>28</td>
<td>42</td>
<td>70</td>
<td>98</td>
</tr>
</tbody>
</table>

After several tests, the optimal fitness value of the assembly sequence is 4.85, and 6 optimal assembly sequences are found, and the 6 sets of optimal assembly sequences are all in line with the engineering practice. Table 2 shows that with the increase of population size, the optimization ability of the algorithm increases gradually, and the maximum fitness value obtained from each test
approximates to the optimal fitness value, and finally is equal. When the population size reaches 600, the probability is as high as 98%. However, with the increase of population size, the average running time increases, but even when the population size is 600, the optimization rate of the algorithm is close to 100%, and the average time of the program is only 83s. If the population size is increased, the running time of the program will be increased, which will lead to a long time consuming and lower efficiency of optimization algorithm. Therefore, when the population size of the assembly sequence is set to 600, the optimization ability of the discrete bat algorithm reaches an ideal state.

Table 3 Optimal assembly sequence

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Optimal assembly sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 1 2 5 4 6 7 9 10 8</td>
</tr>
<tr>
<td>2</td>
<td>3 5 4 1 2 6 9 7 10 8</td>
</tr>
<tr>
<td>3</td>
<td>1 3 2 5 4 6 7 9 10 8</td>
</tr>
<tr>
<td>4</td>
<td>1 3 2 5 4 6 7 10 9 8</td>
</tr>
<tr>
<td>5</td>
<td>5 3 4 1 2 6 7 9 10 8</td>
</tr>
<tr>
<td>6</td>
<td>5 3 4 1 2 6 7 10 9 8</td>
</tr>
</tbody>
</table>

Test 3: the effects of different parameter settings. In practical applications, the update parameters $\alpha$ and $\gamma$ of the loudness $A_i$ and the pulse emission rate $r$ have not been determined yet. According to experience, values are generally taken within (0, 1) [10]. The higher the frequency $f_i$ ($f_i \in [f_{\text{min}}, f_{\text{max}}]$), the shorter its wavelength and the shorter the flight distance. In the actual solution process, the value of $f_i$ can be determined according to the size of the problem domain [9]. The above four factors are divided into three levels (as shown in Table 4), and an orthogonal test table $L_9(5^6)$ is used to test the algorithm (as shown in Table 5). After analysis of the three groups are more ideal parameter values, respectively $\alpha=0.5, \gamma=0.9, f_{\text{min}}=2, f_{\text{max}}=3$ (Group 5 test), $\alpha=0.5, \gamma=0.9, f_{\text{min}}=0, f_{\text{max}}=4$ (Group 6 test) and $\alpha=0.9, \gamma=0.9, f_{\text{min}}=1, f_{\text{max}}=3$ (Group 9 test).

Table 4 Parameter factors and their corresponding levels

<table>
<thead>
<tr>
<th>Corresponding levels</th>
<th>Parameter</th>
<th>$f_{\text{min}}$</th>
<th>$f_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.9</td>
<td>0.9</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5 Orthogonal test table

<table>
<thead>
<tr>
<th>Test NO.</th>
<th>$\alpha$</th>
<th>$\gamma$</th>
<th>$f_{\text{min}}$</th>
<th>$f_{\text{max}}$</th>
<th>$T$ [s]</th>
<th>I</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>93</td>
<td>125</td>
<td>4.85</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>118</td>
<td>118</td>
<td>4.85</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>142</td>
<td>136</td>
<td>4.85</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>132</td>
<td>152</td>
<td>4.85</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>118</td>
<td>82</td>
<td>4.85</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>112</td>
<td>85</td>
<td>4.85</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>135</td>
<td>204</td>
<td>4.85</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>142</td>
<td>116</td>
<td>4.85</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>102</td>
<td>83</td>
<td>4.85</td>
</tr>
</tbody>
</table>

Note: T in the table represents the average running time, I represents the optimal solution algebra, and F represents the optimal fitness value.

Set up the test again according to this group of parameters. The test adopts control variables, increasing the number of tests, and small steps ($\alpha$ and $\gamma$ are each taken as 0.1 steps), so as to analyze the relationship between the parameter change and the algorithm's ability to search for optimization. And more ideal parameter values, the test results shown in Fig. 2. From Fig. 2, we can see that as
the value of $\alpha$ gradually increases, the number of iterations with the optimal value decreases, indicating that the DBA algorithm converges faster. Because the algorithm can find the optimal solution within the range of [0.1, 0.9] in the orthogonal test, the maximum value of $\alpha$ within this range will not cause the algorithm to converge too fast and not find the optimal value. With the increase of $\gamma$, the number of iterations with optimal value decreases, the DBA algorithm converges faster, and the same value of $\gamma$ in the range of [0.1, 0.9] makes the algorithm converge faster but does not make the algorithm premature. Therefore, it is more ideal to take $\alpha=0.9$, $\gamma=0.9$, $f_{\text{min}}=1$ and $f_{\text{max}}=3$ for the parameters of this assembly sequence DBA algorithm.

Figure 2 The relationship between $\alpha$ and $\gamma$ values and the average running algebra of optimal results

Test 4: Comparison between the DBA approach and the PSO approach. The discrete bat algorithm (DBA) is compared with the particle swarm optimization algorithm (PSO), which is commonly used in assembly sequence. The test still uses the assembly of top drive blowout preventer as an example, and the discrete bat algorithm adopts the near optimal parameter above. The parameters of PSO algorithm are set with recommended value [14] ($c_1=c_2=1.49445$). The two algorithms run 50 times each, and the results are shown in Table 6, Fig. 3.

Table 6 Comparison of algorithm test results

<table>
<thead>
<tr>
<th></th>
<th>Optimal fitness</th>
<th>Optimal probability of occurrence [%]</th>
<th>Average running time [s]</th>
<th>Average optimization algebra.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBA</td>
<td>4.18</td>
<td>98</td>
<td>83</td>
<td>260</td>
</tr>
<tr>
<td>PSO</td>
<td>4.18</td>
<td>64</td>
<td>112</td>
<td>368</td>
</tr>
</tbody>
</table>

Figure 3 The relationship between the average fitness of two algorithms and the number of iterations for assembly sequence

Table 6 shows that both algorithms find the optimal fitness value in the test, but the optimization rate of DBA is as high as 98%, which is much higher than that of particle swarm optimization algorithm, and the average running time of DBA and the average optimization algebra are lower than that of PSO algorithm. Obviously, the efficiency of DBA optimization is higher than that of PSO algorithm. From Fig 4, it can be seen that neither of the two algorithms is trapped in local
optimum, but the convergence speed of DBA is faster than that of PSO algorithm. The analysis shows that the DBA proposed in this paper is effective in solving the ASP problem. It is important to note that this paper assumes that the assembly is carried out in the positive and negative direction of the XYZ axis. If both linear and rotational motions occur in the assembly process, it will be simplified to a linear motion only along the axis of the assembly. Because bolts and other fasteners are usually assembled in parallel mode in the process of automatic assembly, the assembly body containing fasteners such as bolts is used as sub-assembly body to participate in assembly.

Conclusions

According to the characteristics of assembly sequence planning problem, this paper redefines the related operations of bat algorithm, which is often used to solve continuous space optimization problem, and then proposes a discrete bat algorithm for assembly sequence planning. By using orthogonal test and variable control method, the parameters of the algorithm for assembly sequence planning are determined, and the comparison of tests shows that the proposed algorithm is superior to the particle swarm optimization algorithm.

Acknowledgements

This work was financially supported by the Key Research and Development Program of Gansu(17YF1GA003).

References

