Improved Cat Swarm Optimization Algorithm for Assembly Sequence Planning

Jianwen Guo[1,*], Zhenzhong Sun[1], Hong Tang[2], Ling Yin[1] and Zhicong Zhang[1]

[1]*School of Mechanical Engineering, Dongguan University of Technology, Dongguan 523808, China*

[2]*Dongguan Neutron Science Center, Dongguan 523890, China*

Abstract: Assembly sequence planning (ASP) is a combinatorial optimization problem in which the order for each part and subassembly is determined. This order is then incorporated into an incrementally expanding subassembly and eventually results in a final assembly. To address this problem, we propose an improved cat swarm optimization (CSO) algorithm and redefine some basic CSO concepts and operations according to ASP characteristics. The feasibility and the stability of this improved CSO are verified through an assembly experiment. The improved CSO is also compared with particle swarm optimization. Experimental results show that the proposed algorithm effectively solves the ASP problem; thus, the application of the proposed algorithm should enhance ASP level.

Keywords: Complex production, Assembly sequence planning (ASP), Cat swarm optimization (CSO), Particle swarm optimization (PSO).

1. INTRODUCTION

Assembly sequence planning (ASP) involves the planning of the order of parts in the assembly process under certain constraint conditions. ASP considers many assembly factors, including geometric feasibility, tool and direction changes. Solving the ASP problem is crucial in manufacturing because it generates an optimal sequence to minimize manufacturing time and cost [1].

Several assembly sequences are feasible in complex production with many parts. Therefore, the determination of an optimal assembly sequence to satisfy time, cost, and reliability requirements is a combinatorial problem. The complexity of this problem is proportional to the number of parts, and the number of feasible assembly sequences increases with equipment complexity [2].

This problem is difficult and impractical to solve with human involvement because of the issue of combinatorial explosion. The amount of research in intelligence assembly sequencing has increased rapidly in recent years, and the intelligence ASP problem is regarded as a discrete search and optimization problem. In line with this research boom, different artificial intelligence approaches have been proposed recently, including graph theory [3], subassembly detection [4], motion planning [5], and evolution algorithms [6].

Evolution algorithms provide new solutions to various complex optimization problems by imitating the self-organization mechanism of natural biological communities and the adaptability of evolution [7-9]. The evolution algorithms that have been investigated for ASP include the genetic algorithms [10], ant colony [11], and particle swarm optimization [12].

The cat swarm optimization (CSO) algorithm was proposed by Chu and Tsai in 2007 [13] and imitates the natural behavior of cats. This algorithm is categorized under the group of swarm intelligence. It effectively determines global solutions. Many optimization problems have been resolved successfully using this algorithm, such as linear antenna array synthesis [14], infinite impulse response system identification [15], and travelling salesman problem [16]. The aforementioned studies suggest that CSO is a simple, robust, and fast algorithm for solving combinatorial optimization problems.

However, few studies have applied CSO as an effective method to solve the ASP problem. This paper presents an improved CSO algorithm to solve the ASP problem. Several basic concepts and operations of CSO are redefined according to ASP characteristics. The feasibility and stability of this improved CSO are then verified through an assembly experiment. The improved CSO is also compared with particle swarm optimization (PSO). The experimental results show that the proposed algorithm effectively solves the ASP problem. Therefore, the application of the proposed algorithm should enhance ASP level.

The remainder of the paper is organized as follows. Section 2 introduces related works. Section 2 describes CSO. Section 3 describes fitness function. Section 4 discusses the improved CSO for ASP. Section 5 describes the experiments and analyses. Finally, Section 6 concludes the study.

*Address correspondence to this author at the School of Mechanical Engineering, Dongguan University of Technology, Dongguan 523808, China; Tel: +86 13532853596; E-mail: guojw@dgut.edu.cn

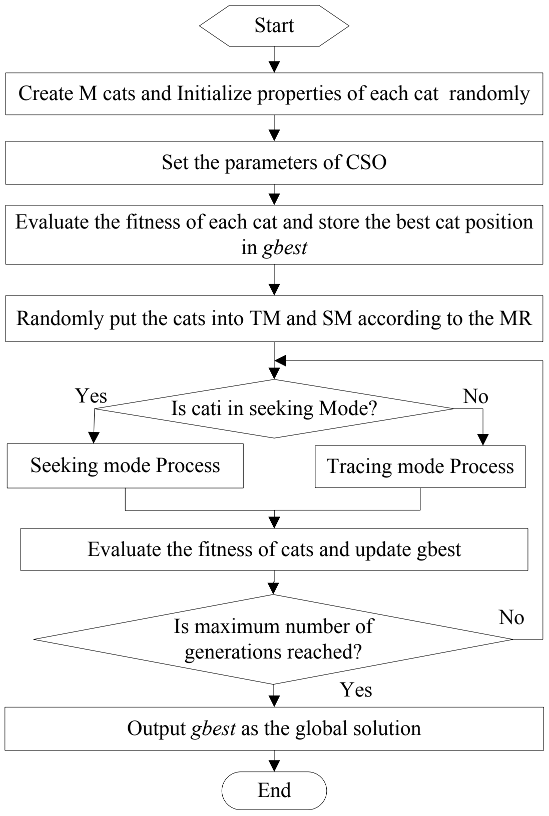
Start
Create M cats and Initialize properties of each cat randomly
Set the parameters of CSO
Evaluate the fitness of each cat and store the best cat position in *gbest*
Randomly put the cats into TM and SM according to the MR
Is cati in seeking Mode?
Yes
No
Seeking mode Process
Tracing mode Process
Evaluate the fitness of cats and update gbest
Is maximum number of generations reached?
No
Yes
Output *gbest* as the global solution
End

Fig. (1). Flowchart of CSO Algorithm.

2. CSO

CSO is a new evolutionary algorithm inspired by cat behavior. Two models simulate the strong curiosity of cats in moving objects and their hunting skills, namely, the seeking model (SM) and the tracing model (TM). The SM simulates alertness and very slow movement, whereas the TM simulates high-speed chasing behavior. The two models are mathematically modeled to solve optimization problems.

In CSO, the position of a cat corresponds to a solution to the problem for optimization. The final solution is the ideal position of one of the cats. A cat is characterized by the following properties: (1) a position composed of *n* dimensions; (2) velocities for each dimension; (3) a fitness value that represents the accommodation of the cat to the fitness function; and (4) a flag that indicates whether a cat is in the TM or the SM.

2.1. Algorithm

The optimal solution to the CSO algorithm is determined by using two groups of cats. One group contains cats in SM, whereas the other group contains cats in TM. Each cat belongs to one mode. Moreover, this study introduces a mixture ratio (MR) that defines the ratio of the number of cats in TM to that of the number of cats in SM. Most cats in the algorithm are in the SM while the rest are in the TM, which is consistent with the real-life behavior of cats.

Global and local search processes are performed in SM and TM, respectively. In the process, local and global searches can be combined in CSO to accelerate convergence and to enhance the quality of optimization problem solutions.

The flowchart of the CSO algorithm is shown as in Fig. (**1**). The steps followed by the algorithm are as follows. (1) A population with *M* cats is generated as the solution in *n* dimensions. (2) CSO algorithm parameters are selected. (3) The fitness function of each individual is calculated, and the ideal position of the cat is determined. This position is recorded as *gbest*. (4) Cats are selected at random from the population according to the MR. Their flags are set to either SM or TM. (5) The cats in the SM undergo the SM process, whereas the cats in the TM are subject to the TM process. (6) The new fitness function value of each cat is calculated. If this value is lower than that of *gbest*, then the former replaces the latter. (7) If the above condition is unsatisfied, then steps (5) to (7) are repeated to determine whether the end condition is met. Otherwise, the program is ended and the global optimal solution is saved.

2.2. SM

SM simulates a cat in a state of rest and determining its next move. Three parameters are introduced. **Seeking memory pool (SMP)**: the size of the seeking memory of a cat as denoted by the number of copies of a cat captured in seeking mode. **Seeking range of a selected dimension (SRD)**: the maximum difference between the new and old values in the selected dimension. If a dimension is selected for mutation, this difference may remain within the SRD range. **Counts of dimension to change (CDC)**: the number of dimensions for mutation that indicates how many dimensions are to be altered.

Cats in the SM are inputted into the memory pool by copying their positions. Each individual in the memory pool is subject to a mutation operator to generate a new position. The position is updated by calculating the fitness of all of the cats in the memory pool and by selecting a candidate point with the highest fitness value as the position to which the cats will move.

The SM process is performed as follows:

(1) An *N* copy of the *j*th cat is produced. The copies are inputted into the SMP, whose size is represented by *N*.

(2) The mutation operator is utilized. The individuals in the SMP are distributed at random based on the SRD and CDC values.

(3) The fitness value of each cat in the SMP is calculated.

(4) The executive selection operator selects the best cat from the SMP to replace the original cat.

2.3. TM

TM simulates the tracking of a target by a cat. Once a cat goes into TM, it moves according to its own velocities in each dimension. As with PSO, the global optimum position replaces the current position of the cat, and the cat approaches the optimal solution step by step. The TM can be described by the following steps.

(1) The best position of the whole cat swarm is denoted by $gBest^{(d)}(t)$. Each cat has a velocity represented by $V_i = \{v_i^1, v_i^2, \cdots, v_i^L\}$. This velocity is updated according to Formula (1):

$$\begin{cases} v_k^{(d)}(t+1) = v_k^{(d)}(t) + vr_k^{(d)}(t) \\ vr_k^{(d)}(t) = c \times rand \times (gBest^{(d)}(t) - x_k^{(d)}(t)) \\ d = 1, 2, \cdots, L \end{cases} \tag{1}$$

where $v_k^{(d)}(t+1)$ is the velocity of the *k*th cat in *d*th dimension; L is the total dimension length; $gBest^{(d)}(t)$ is the *d*th dimension of the position of the cat $gBest\ (t)$ with the highest fitness; $x_k^{(d)}(t)$ represents the *d*th dimension of the position of cat $x_k(t)$; *c* represents a constant, which is set according to a specific scene; and *rand* represents a random number between [0, 1].

(2) Each cat in the TM updates its position using Formula (2):

$$x_k^{(d)}(t+1) = x_k^{(d)}(t) + v_k^{(d)}(t+1) \tag{2}$$

where $x_k^{(d)}(t+1)$ is the *d*th dimension of the position of a cat after updating.

3. FITNESS FUNCTION

Four evaluation indicators are introduced to analyze the assembly sequence, namely, assembly stability, the changing times of the assembly tool, and the changing times of assembly direction.

(1) Geometric feasibility. The first step in solving ASP problems involves the geometric feasibility of assembly sequences. This factor represents the uninterrupted assembly operation process.

(2) Assembly stabilities. In the actual assembly process, parts may be rendered unstable by gravity. In this case, several assembly operations known as stable operations must maintain stability using a jig or auxiliary tool. A stable assembly sequence operation results in inefficient assembly; therefore, the stability of the assembly sequence should be evaluated.

(3) Changing times of the assembly tool. Changing the assembly tool during assembly increases assembly time and costs. Therefore, the changing times of the assembly tool should be as short as possible.

(4) Changing times of the assembly direction. The reduced changing times of the assembly direction shortens assembly time and enhance assembly efficiency.

This study introduces n_f, n_s, n_t, and n_d to quantify the influence of the factors above on assembly sequence.

n_f is the overall period of geometric feasibility for an assembly sequence.

n_s is the period of stable assembly sequence operation. A small value indicates a stable assembly sequence.

n_t is the changing times of an assembly tool for an assembly sequence.

n_d is the changing times of assembly direction for an assembly sequence.

The values of the parameters above are based on our previous research work [17].

In a real assembly problem, the geometrically infeasible assembly sequence should be eliminated first. A penalty function $c_f n_f$ is then applied to accelerate the algorithm convergence rate during assembly sequence generation. The weighted fitness function is expressed as follows:

```
Step (1): Set $j = 1$
Step (2): IF $x_{a,j} = x_{b,j}$ THEN
              $vo_{ab,j} = (j, j)$
          ELSE
              $vo_{ab,j} = (j, k)$
              $X_b = X_b + vo_{ab,j}$
          /* k : number of $x_{a,j}$ dimensions in $X_b$ */
          ENDIF
Step (3): $j = j + 1$
Step (4): IF $j < n$ THEN
              Proceed to Step (2)
          ENDIF
Step (5): END
```

Fig. (2). Operational rules for position subtraction.

$$f = c_s n_s + c_t n_t + c_d n_d + c_f n_f \tag{3}$$

where c_s, c_t, c_d, and c_f are the weighting factors for each evaluating indicator and c_f must generally be larger than the other three weighting factors (i.e., $c_f \geq \frac{n}{2}\max\{c_s, c_t, c_d\}$). In this study, a small fitness function value indicates a good frog position and a good assembly sequence.

4. IMPROVED CSO FOR ASP

The ASP is a combinatorial optimization problem in which every solution dimension is discrete. The CSO algorithm is suitable for continuous optimization. Relevant algorithm operations and concepts must be redefined when applied to ASP, which can be treated as a discrete search and optimization problem as mentioned previously. Therefore, this paper proposes an improved cat swarm algorithm for ASP. The relevant operations and concepts in assembly sequence planning are redefined as follows.

(1) Position of cat *i*

The position of a cat is an *n*-dimension vector that represents an assembly sequence. To maintain the diversity of the cat swarm, the initial position of each cat is randomly initialized as an *n*-dimension vector as follows.

$$\begin{cases} X_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,j}, \cdots x_{i,n})^T \\ \quad x_{i,j} \in \{1, 2, \cdots, n\} \end{cases} \tag{4}$$

where *n* represents the number of parts and each component of $x_{i,j}$ varies in the location vector of the same cat.

(2) Permutation factor

The permutation factor $vo(s,k)$ in the location vector of cat *i* indicates that the positions of the *s* and *k* components in the vectors have switched ($s, k \in \{1, 2, \cdots, n\}$). If $s = k$, the function of the permutation factor on the position vector does not change and the permutation factor is invalid.

(3) Velocity of cat *i*

In assembly planning given *n* parts, the velocity of cat *i* is defined as the ordered sequence with *n-1* permutation factors:

$$V_i = (vo_{i,1}, vo_{i,2}, \cdots, vo_{i,j}, \cdots, vo_{i,n-1})^T \tag{5}$$

(4) Position subtraction. The final output of position subtraction is a velocity vector.

$$X_a - X_b = V_{ab} \tag{6}$$

Its operational rules are as shown in Fig. (**2**).

(5) Scalar multiplication of velocity

The output of the scalar multiplication of velocity remains a velocity vector.

$$c \times rand \times V_i = V_j \tag{7}$$

where $c \in (0,1)$ is a fixed value, and *rand* is a random sequence of *n* dimensions. Each element r_k follows a uniform random distribution between 0 and 1. The value of the permutation factor in V_j is determined as follows:

$$vo_{j,k} = \begin{cases} vo_{i,k}, & r_k \geq c \\ (k,k), & r_k < c \end{cases} \qquad k \in [1, n-1] \tag{8}$$

(6) Addition of position and velocity

The output of the addition of position and velocity is a position vector. Its corresponding rule acts on the position vectors according to the ordered sequence of permutation factors in the velocity vector. Nonetheless, the addition of position and velocity cannot satisfy the commutative law.

(7) The number of genes changed by an individual in the search model.

A permutation factor is defined as a gene in the assembly sequence and is represented by a uniformly distributed random integer between 0 and n-1.

(8) Modification range of each gene

The modification range of each gene is defined as a permutation factor that participates in an operation with a certain probability. When the random number is greater than this probability, this permutation factor is invalid and does not participate in the operation.

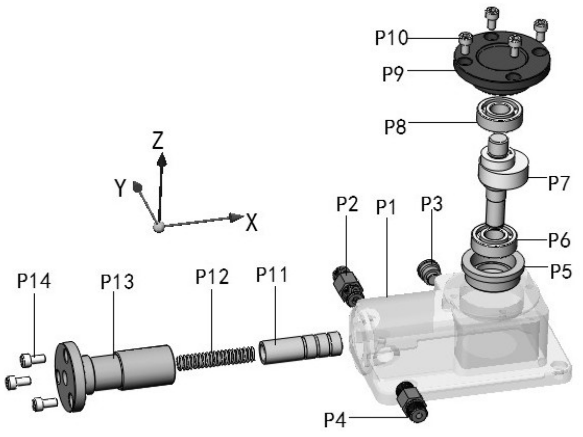

Fig. (3). Detailed View of the Plunger Pump.

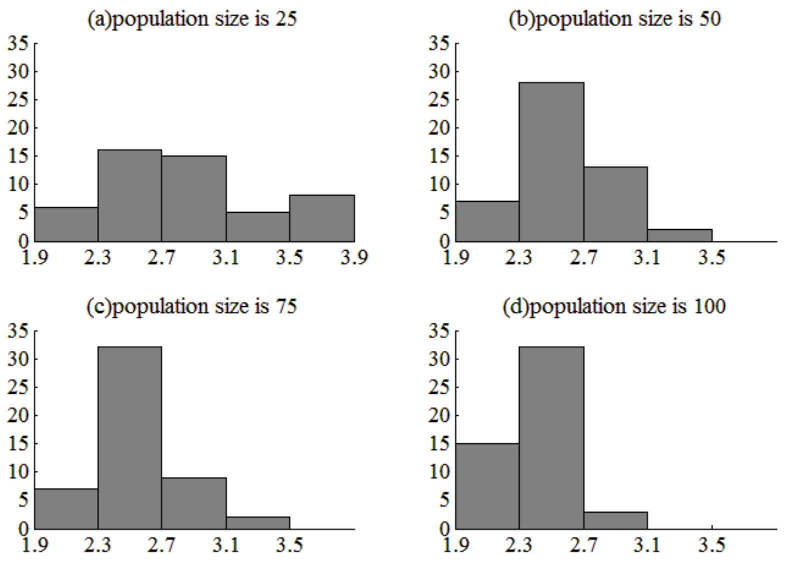

Fig. (4). Distribution of Local Optimal Fitness.

Based on the redefinition above, the update formula for cat velocity and position in the TM can be modified into the following:

$$\begin{cases} x_k^{(d)}(t+1) = x_k^{(d)}(t) + v_k^{(d)}(t) + vr_k^{(d)}(t) \\ vr_k^{(d)}(t) = c \times rand \times (gBest^{(d)}(t) - x_k^{(d)}(t)) \\ d = 1,2,\cdots,L \end{cases} \tag{9}$$

$$v_k^{(d)}(t+1) = x_k^{(d)}(t+1) - x_k^{(d)}(t) \tag{10}$$

Specifically, we first update the position of the cat before its velocity. The value of constant *c* in Formula (10) is set to 0.5.

5. EXPERIMENTS AND ANALYSIS

A plunger pump that consists of 14 parts is used for the ASP experiment. Fig. (**3**) displays a detailed view of this pump.

5.1. Algorithm Test

Following an orthogonal experiment on plunger pump assembly, the algorithm quickly identifies an optimal assembly sequence when the weighting factors of the evaluation indicator in the fitness function are denoted by $cf = 4$, $cs = 0.5$, $ct = 0.3$, $ct = 0.3$, and $cd = 0.2$. Based on multiple comparison experiments, the algorithm optimization capability is optimized when MR is 0.04, SMP is 5, and SRD is 0.2.

The ASP experiment is conducted with population sizes of 25, 50, 75, and 100. The value of each parameter of the improved CSO and the weighting factors of the evaluation indicators in the fitness function are constant. The optimal assembly sequence was determined according to the distribution of fitness function value and the results of the ASP experiment. The analysis results show that the number of algorithm iterations is 200 and that of reiterated operation times is 50, as shown in Fig. (**4**) and Table **1**.

Table 1. Comparison of the 50 Optimal Asp Results Given Different Population Sizes.

Population Size	**25**			**50**			**75**			**100**		
	S	***D***	***T***	***S***	***D***	***T***	***S***	***D***	***T***	***S***	***D***	***T***
Information Related to Assembly Sequence	1	-Y	T1	1	+Y	T1	1	-Z	T1	1	-Z	T1
	3	-Y	T2	4	+Y	T2	5	-Z	T2	5	-Z	T2
	2	-Y	T2	11	+X	T2	6	-Z	T2	6	-Z	T2
	5	-Z	T2	12	+X	T2	7	-Z	T2	7	-Z	T2
	6	-Z	T2	5	-Z	T2	8	-Z	T2	8	-Z	T2
	7	-Z	T2	6	-Z	T2	11	+Y	T2	11	+X	T2
	8	-Z	T2	7	-Z	T2	12	+X	T2	12	+X	T2
	4	+Y	T2	8	-Z	T2	13	+X	T1	13	+X	T1
	11	+X	T2	9	-Z	T1	9	+X	T1	9	-Z	T1
	12	+X	T2	13	+X	T1	10	-Z	T3	10	-Z	T3
	13	+X	T1	14	+X	T3	14	-Z	T3	14	+X	T3
	9	-Z	T1	3	-Y	T3	3	-Y	T3	3	-Y	T3
	10	-Z	T3	2	-Y	T3	2	-Y	T3	2	-Y	T3
	14	+X	T3	10	-Z	T3	4	+X	T3	4	+Y	T3
Changing Times of Assembly Direction	5			5			5			5		
Changing Times of the Assembly Tool	3			3			3			3		
Unstable Operation Times	0			0			0			0		
Single Execution Time/s	6.21			12.13			18.18			23.97		
Fitness	1.9			1.9			1.9			1.9		

Note: *S* represents the assembly sequence; *D* represents the assembly direction; and *T* represents the assembly tool.

The results of many experiments indicate that the fitness value of the global optimal assembly sequence *F* is 1.9. Table **2** suggests that the CSO algorithm can determine an optimal assembly sequence in a small population scale; therefore, its optimizing capacity is strong. Fig. (**4**) shows that the distribution of the local optimal fitness value at 50 times is within the following ranges: 1.9 to 2.2, 2.3 to 2.6, 2.7 to 3.0, 3.1 to 3.4, and >= 3.5. Thus, an increase in population scale can effectively improve the planning effect of the algorithm.

Fig. (**5**) presents the changes in the mean and optimal average fitness values with the increase in iteration times when the population scale is 75. The average optimal fitness value steadily decreases as algorithm iteration increases; this finding indicates the capability of the algorithm to optimize stability. Moreover, the mean fitness value remains high in current iteration periods. This occurrence is attributed to the fact that the MR is small, in accordance with real-life cat behavior. Most cats are in the SM and utilize mutation operators, which can enhance population diversity and reduce the convergence rate of the algorithm.

5.2. Experimental Comparison

The performance of the improved CSO algorithm is compared with PSO for verification. Many experimental results suggest that the performance of PSO is superior when the inertia weight of PSO is 0.6. Table **2** presents the planning results with different population scales.

This table indicates that in a small population scale, the improved CSO algorithm can obtain an optimal assembly sequence with a fitness function value of 1.9. Therefore, its optimizing capacity is strong. Furthermore, the improved

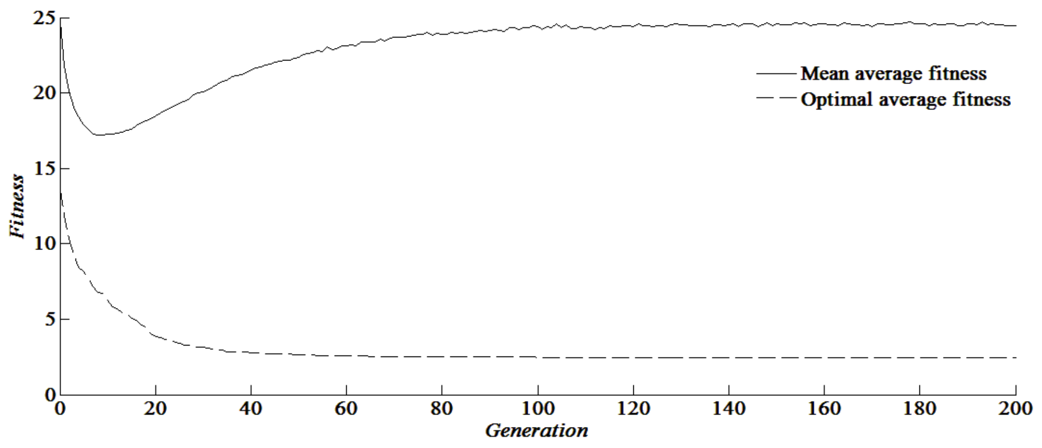

Fig. (5). Mean Fitness and Optimal Fitness.

Table 2. Comparison of the Improved CSO And PSO.

Parameter	CSO				PSO			
Population Size	25	50	75	100	25	50	75	100
Iteration Times	200	200	200	200	200	200	200	200
Feasible Assembly Sequence	43	50	50	50	27	39	43	48
Single Execution Time/s	6.21	12.13	18.18	23.97	5.54	10.98	16.54	21.95
Fitness of Optimal Assembly Sequence	1.9	1.9	1.9	1.9	2.1	1.9	1.9	1.9

CSO identifies more feasible assembly sequences than PSO in the same population scale. The mean fitness function value of the final planning result is also lower than that of PSO. These results are attributed to the fact that PSO cannot easily jump out of local optimization, whereas the cat in the SM can copy and mutate in the improved CSO; hence, it does not slip into local optimization easily. The mean single-run time of the improved CSO is longer than that of PSO. Its operation efficiency is slightly lower but remains within an acceptable range.

CONCLUSION

Assembly process optimization significantly reduces manufacturing time and cost and is widely applicable in complex production manufacturing. This study introduces ASP as a component of assembly process optimization. Moreover, the evolution algorithm is a useful tool for ASP, which is considered a combinatorial optimization problem. To address the ASP issue, this study develops an advanced evolution algorithm called the improved CSO. CSO is an evolution algorithm that is used to calculate the global optima of several combinatorial problems. It effectively determines solutions. Several concepts and operations are redefined with respect to the discreteness characteristic of ASP, and the conclusions of this study are as follows:

(1) Algorithm tests show that the optimizing capacity of the algorithm and the quality of sequence planning results improve with the increase in population scale.

(2) A small MR value can ensure population diversity. However, it also reduces the convergence speed of the algorithm.

(3) Experiment results indicate that the improved CSO is effective when used for ASP. Furthermore, it generates a better assembly sequence than PSO. Thus, the application of the proposed algorithm should enhance ASP level.

Nonetheless, the experimental results also suggest that the mean single-run time of the improved CSO is longer than that of PSO. Therefore, additional studies should be conducted to improve the proposed algorithm and to enhance its efficiency.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

The study was supported by the National Natural Science Foundation of China (Grant No.71201026), the Project of

Department of Education of Guangdong Province (No. 2013KJCX0179), the Development Program for Excellent Young Teachers in Higher Education Institutions of Guangdong Province (No.Yq2013156), the China Spallation Neutron Source Electromechanical Technology R & D Joint Laboratory Foundation (No. ZD120512), and the Dongguan Social Science and Technology Development Project (NO. 2013108101011).

REFERENCES

[1] L. Wang, S. Keshavarzmanesh, H. Y. Feng, and R. O. Buchal, "Assembly process planning and its future in collaborative manufacturing: a review," *The International Journal of Advanced Manufacturing Technology*, vol. 41, no. 1-2, pp. 132-144, 2009.

[2] P. Jiménez, "Survey on assembly sequencing: a combinatorial and geometrical perspective," *Journal of Intelligent Manufacturing*, vol. 24, no. 2, pp. 235-250, 2013.

[3] A. R. Selva, R.C. Castro, and V.G.F. Fríasa, "Design of disassembly sequences using search strategies. Application of IDA* in state diagrams," *International Journal of Production Research*, vol. 49, no. 11, pp. 3395-3403, 2010.

[4] J. Ko, E. Nazarian, H. Wang, and J.Abell, "An assembly decomposition model for subassembly planning considering imperfect inspection to reduce assembly defect rates," *Journal of Manufacturing Systems*, vol. 32, no. 3, pp. 412-416, 2013.

[5] C. Morato, K. N. Kaipa, and S. K. Gupta, "Improving assembly precedence constraint generation by utilizing motion planning and part interaction clusters," *Computer-Aided Design*, vol. 45, no. 11, pp. 1349-1364, 2013.

[6] M. Fadzil, F. Rashid, W. Hutabarat, A. Tiwari, "A Review on Assembly Sequence Planning and Assembly Line Balancing Optimization using Soft Computing Approaches," *International Journal of Advanced Manufacturing Technology*, vol. 59, no. 1-4, pp.335-349, 2011.

[7] R.B. Xiao, W.M. Chen, and T.G. Chen, "Modeling of Ant Colony's Labor Division for the Multi-project Scheduling Problem and Its Solution by PSO, *Journal of Computational and Theoretical Nanoscience*, vol. 9, no.2, pp.223-232, 2012.

[8] R.B. Xiao, Z.W. Tao, and T.G. Chen, "An Analytical Approach to the Similarities between Swarm Intelligence and Artificial Neural Network," *Transaction of the Institute of Measurement and Control*, vol. 34, no.6, pp.736-745, 2012.

[9] R.B. Xiao, and T.G. Chen, "Relationships of swarm intelligence and artificial immune system," *International Journal of Bio-Inspired Computation*, vol. 5, no. 1, pp.35-51, 2013.

[10] H. S. Wang, Z. H. Che, and C. J. Chiang, "A hybrid genetic algorithm for multi-objective product selection problem with ASP and ALB," *Expert Systems with Application*, vol. 39, no. 5, pp. 5440-5450, 2012.

[11] H. B. Shan, S. H. Zhou, and Z. H. Sun, "Research on assembly sequence planning based on genetic simulated annealing algorithm and ant colony optimization algorithm," *Assembly Automation*, vol. 29, no. 3, pp. 249-256, 2009.

[12] M. Li, B. Wu, P. Yi, C. Jin, Y. Hu, and T. Shi, "An improved discrete particle swarm optimization algorithm for high-speed trains assembly sequence planning," *Assembly Automation*, vol. 33, no. 4, pp. 360-373, 2013.

[13] S.C. Chu, and P.W. Tsai, "Computational intelligence based on the behavior of cats," *International Journal of Innovative Computing, Information and Control*, vol. 3, no. 1, pp.163-173, 2007.

[14] L. Pappula, and D. Ghosh, "Linear antenna array synthesis using cat swarm optimization," *International Journal of Electronics and Communications*, vol. 68, no. 6, pp. 540-549, 2014.

[15] G. Panda, P. M. Pradhan, and B. Majhi, "IIR system identification using cat swarm optimization," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12671-12683, 2011.

[16] S. Wang, J.W. Guo, Z.Z. Sun, H.B. Chen, and R.Y. Li. "A hybrid cat swarm optimization algorithm for TSP," *Energy Education Science and Technology Part A: Energy Science and Research*, vol. 32, no. 6, pp. 5739-5742, 2014.

[17] J.W. Guo, S. Wang, H.B. Chen, Z.Z. Sun, and Z.C. Zhang, "An intelligent approach for remote handling maintenance sequence planning in radiation environment," *Journal of Chemical and Pharmaceutical Research*, vol. 6, no. 4, pp. 543-552, 2014.

Received: June 02, 2015 Revised: August 02, 2015 Accepted: September 05, 2015