DOI: 10.21062/mft.2023.056 http://www.journalmt.com

# Assembly Sequence Intelligent Planning based on Improved Particle Swarm Optimization Algorithm

Wei Zhang (0000-0002-3176-9520)
School of intelligent manufacturing, Xiamen City University, Xiamen, Fujian 361008, China. E-mail: zhangw@stu.xmu.edu.cn

**Combinatorial explosion and limited efficiency when solving complex products with multiple parts are two issues that traditional assembly sequence solution methods frequently run into. To improve the level of assembly sequence planning (ASP), an interference matrix is constructed to convey the fundamental assembly information of a product. Taking the stability of the assembly sequence, the number of assembly direction changes, and the number of assembly tool changes as evaluation indicators, a fitness function is constructed. On the basis of the unique characteristics of the ASP problem, an improved particle swarm optimization (IPSO) approach is devised. Redefining particle positions, velocities, and their update operations, and introducing mutation operators in genetic algorithm (GA) to improve the ability of PSO algorithms to jump out of local optima. Additionally, the algorithm's convergence speed is enhanced by adjusting the value of the inertia weight. Finally, an example is provided to demonstrate the IPSO algorithm's usefulness and efficiency.**

**Keywords:** Assembly Sequence Planning, Improved Particle Swarm Optimization, Interference Matrix, Genetic Algorithm

## 1 Introduction

Efficiency and cost of the product assembly process are the key factors to increasing the competitiveness in the manufacturing sector. According to statistics, the cost of assembly makes up more than 40% of the production cost, and the workload for assembly makes up 20% to 70% of the overall labor required to manufacture a product [1]. As the core of assembly planning, the quality of ASP is directly related to the assembly cost of products. A suitable assembly sequence might help to save assembly time and workload, thus providing higher productivity and improving product assembly quality [2]. As the number of product parts and components increases, the conceivable assembly sequence also increases exponentially [3-5]. Engineers need to spend a lot of time determining their assembly sequence. Because traditional empirical design is difficult to ensure the correctness and consistency of ASP, the optimal assembly sequence is easily overlooked in the design process. To reduce the negative impact of assembly sequence in the process of assembly, it is vital to optimize and study the ASP problem in the process of product assembly and provide a support for the feasibility and optimality of the product in the real assembly process.

In recent years, using the intelligent assembly method to solve ASP problem has gained a lot of attention in the field of assembly process planning. Many scholars have studied ASP problem via different intelligent methods. Such as GA, ant colony optimization (ACO), PSO, and simulated annealing algorithm (SA) provide powerful methods for solving assembly sequences of complex products. As an illustration, Wang et al. [6] solve the ASP problem in the assembly of antenna reflectors by GA. Mishra et al. [7] introduced an intelligent assembly sequence optimization approach based on the flower pollination algorithm (FPA), which automatically generates multiple feasible assembly sequences by minimizing the number of direction changes and tool changes under various priority constraints. Wu et al. [8] completed the ASP of eccentric milling machine using PSO algorithm. In order to improve the global search ability of genetic algorithms, Li et al. [9] adopted a new coding method to optimize the ASP of satellite partial structures. Bala et al. [10] proposed a new hybrid artificial intelligence technology, which combines GA to realize artificial immune system (AIS), so as to find an optimal and feasible algorithm to extract assembly sequences from possible assembly sequences. Gunji et al. [11] sugguested an assembly sub-detection approach based on teaching learning based optimization algrithm to optimize robot ASP.

GA is a meta heuristic method that imitates the principle of natural selection [12]. Due to its global search and gradient information independent optimization capabilities, it has been widely used in many combinatorial optimization problems [13,14], including ASP problems [15-19] because of its global search and gradient information independent optimization ability. However, the GA is highly dependent on the quality and size of the initial population, which necessitatess a large proportion of feasible assembly sequences in the initial population. The feasible assembly sequences need to be set manually, which takes a lot of time. Inappropriate assembly sequences in the initial population may often lead the search process to evolve in the direction of poor, and ultimately may not get the optimal assembly sequence, or even may not converge. Simulated annealing algorithm can solve some nonlinear problems and obtain the optimal solution with a high probability [3], which has defect that the global search ability is poor. When a product has a large number of parts to assemble, the algrithm can only obtain an approximate solution. PSO originates from the study of bird predation behavior [15]. Compared with GA, its convergence speed is faster and has a memory function, but it is easy to fall into local optimization [16,20]. The immune algorithm [12] is a relatively new algorithm in the field of intelligent algorithms, which has numerous issues in system modeling and other areas.

In order to solve the ASP problem of complex products, an IPSO algorithm is proposed to solve the above problems by setting the objective function based on the reorientation times of parts, the number of assembly tool switching, and the assembly stability in the assembly process. The IPSO algorithm solves the problem that the PSO algorithm is easy to fall into the local optimum. The algorithm is verified and analyzed by an example. The verification results show that the algorithm can effectively solve the ASP problem of complex products.

The rest of this paper is organized as follows. In section 2, we analyze the assembly sequence problem. In section 3, we propose an improved particle swarm optimization algorithm and problem-solving method. Section 4 verifies the rationality of the model theory and the effectiveness of the algorithm through case studies. Finally, we put forward the conclusions with limitations and describe the future work in section 5.

## 2 Analysis of assembly sequence problems

### 2.1 Interference matrix

An effective assembly sequence must first satisfy the geometric constraints of the assembly. According to the interference matrix of all parts in the product to be assembled in each assembly direction, the feasible assembly sequence of parts $p_i$ and parts $p_j$ without interference can be found. The interference matrix is used to judge the geometric feasibility of the parts in the product in each assembly direction. Generally, the interference matrix is established according to the six directions of space $\pm x, \pm y, \pm z$. Assuming that the assembly model is composed of *n* parts, the interference matrix can be expressed as:

$$IM = (I_{ijk})_{n\times n} \tag{1}$$

Where: $I_{ijk}$ represents the interference between part $p_j$ and part $p_i$ when it is assembled along the k direction, which is one of the $\pm x, \pm y, \pm z$ directions. The value of $I_{ijk}$ can be expressed as:

$$I_{ijk} = \begin{cases} 0, \text{When } p_i \text{ is assembled along } k \text{ direction without interference with } p_j \\ 1, \quad \text{When } p_i \text{ is assembled along the } k \text{ direction and interferes with } p_j \end{cases} \tag{2}$$

If the sequence $AP = (p_1, p_2, \cdots, p_{i-1})$ is the assembled part sequence and $p_i$ is the part to be assembled, the feasible assembly direction of the part $p_i$ can be determined by the following formula (3).

$$K = \sum_{j=1}^{i-1} I_{ijk} \tag{3}$$

*K* is the six directions of the Cartesian axis. Judge the assembly interference of the parts in each direction. If the formula is 0, it means that the parts $p_i$ are assembled in this direction without interference. If the formula is not 0, interference will occur and this assembly sequence is not feasible.

### 2.2 Fitness function

In the assembly process of complex products, an excellent assembly sequence should have lower assembly cost, shorter assembly time, and higher assembly quality. Therefore, according to the features of product assembly, the evaluation index and objective function of the assembly sequence are defined, which are constituted of the assembly tool changes times, the assembly direction changes times, and the assembly stability. Among them, assembly stability directly affects assembly quality, and the assembly direction change times and the assembly tools change times affect the assembly time and assembly cost.

1) Stability of assembly sequence. In order to increase the stability and reliability of the assembly, we need to establish a contact matrix to guarantee the normal operation of the assembly process [1]. Set the stability parameter as $C_{ij}$, and the stability judgment rule of the assembly sequence: set $AP = (p_1, p_2, \cdots, p_{i-1})$ as the assembled part sequence and the parts $p_i$ to be assembled. If the two parts are in direct contact and maintain a stable connection relationship $C_{ij} = 1$, if the two parts are only in direct contact $C_{ij} = 0.4$, and if the two parts

are not connected, then $C_{ij} = 0$. The overall assembly stability of the product is:

$$V_s = \sum_{i=2}^{n} C_{ij} \tag{4}$$

2) Assembly direction change times. If the direction of assembly changes too frequently, it will greatly increase the cost and difficulty of assembly, so we need to minimize the amount of redirection of assembly. Set the direction change parameter as $D_{ij}$. If the direction of two parts does not change, then $D_{ij} = 0$, otherwise $D_{ij} = 1$.

For any feasible assembly sequence $(p_1, p_2, \cdots, p_n)$, the solution steps for the assembly direction changes times Vd of this assembly sequence are as follows.

- (1) Let $i$=0, $V_d$=0.
- (2) Then when assembling parts Pi+1, the assembly direction needs to be changed to make $V_d$=$V_d$+1, otherwise, $V_d$ = $V_d$; Skip to step (3);
- (3) Let $i$=$i$+1, if $i$+1<$n$, repeat step (2); Otherwise, skip to step (4).
- (4) End.

Obviously, the smaller the $V_d$, the smaller the assembly direction changes times and the lower the assembly cost.

Assembly tools change times. Each part has an assembly tool. If different assembly tools are used for assembling several parts in succession, the assembly cost will also increase. Set the continuity parameter as $T_{ij}$. If the two-part assembly tool does not change, $T_{ij}$=0, otherwise $T_{ij}$=1.

For any feasible assembly sequence $(p_1, p_2, \cdots, p_n)$, the solving steps of assembly tool change times Vt of this assembly sequence are similar to those of assembly direction change times.

Therefore, the fitness function values are constructed as follows:

$$\text{Max} \quad J = \omega_1 * V_s / (\omega_2 * V_d + \omega_3 * V_t) \tag{5}$$

## 3 Operation steps of IPSO algorithm

### 3.1 Definition of PSO algorithm considering assembly sequence planning

PSO algorithm is mainly used for optimization of continuous functions. In order to enable PSO algorithm to be applied to models established in discrete space, the location, speed, and update operations of particles are redefined based on the characteristics of ASP models.

Definition 1: The position of particles. The position vector of the ith particle is expressed as $P^i = (p_1^i, p_2^i, \cdots, p_n^i)$, indicating that the assembly is performed in the order of $p_1^i, p_2^i, \cdots, p_n^i$, and n is the number of parts of the product.

Definition 2: particle speed. The velocity vector of the ith particle is expressed as $V^i = (v_1^i, v_2^i, \cdots, v_{n-1}^i)$. The function of the speed operator $V_{(x,y)}$ is to exchange the position of the x-th part and the y-th part in the assembly sequence to generate a new assembly sequence.

Definition 3: Addition of position and speed. The result of a particle's position vector plus its velocity vector is a new position vector. The formula is expressed as $P^{i+1} = P^i + V^i$.

Definition 4: Subtraction between positions. The result of subtracting two position vectors is a velocity vector.

Set $P^s = (p_1^s, p_2^s, \cdots, p_n^s)$, $P^k = (p_1^k, p_2^k, \cdots, p_n^k)$. $P^s - P^k = V^{(s,k)} = (v_1^{(s,k)}, v_1^{(s,k)}, \cdots, v_1^{(s,k)})$.

Definition 5: Multiplication of speed. Let the velocity vector of a particle be $V^1 = (v_1^1, v_2^1, \cdots, v_{n-1}^1)$ and the coefficient c, $c \in [0,1]$. Define the number of multiplication of velocity vector and coefficient as $V^2 = c * V^1 = (v_1^2, v_2^2, \cdots, v_{n-1}^2)$. Take values $V^2$ according to the following rules:

$$v_i^2 = \begin{cases} v_i^2 & r \geq c \\ 0 & r < c \end{cases} \tag{6}$$

Where $r$ is a random number evenly distributed between 0 and 1.

Definition 6: Addition of speed. The result of the addition of two velocity vectors is a new velocity vector. In order to facilitate calculation, the velocity vector is not directly added, but added with the particle position vector in order, and then subtracted with the new and old position vectors to obtain the desired velocity.

Through the above redefinition, the particle position and velocity update formula applicable to solving the ASP problem of discrete space model is:

$$V^{k+1} = \omega * V^k + \left\{ \left[ c_1 * gBest - P^k \right] + \left[ c_2 * pBest - P^k \right] \right\} \tag{7}$$

$$P^{k+1} = P^k + V^{k+1} \tag{8}$$

### 3.2 Algorithm improvement and implementation steps

IPSO algorithm can improve the local search effect by adding mutation operator; On the other hand, it can maintain the individual differences of the population, prevent the occurrence of premature convergence, and increase the probability of the particle finally converging to the global optimal solution.

(1) Particle initialization. The solution of ASP is a feasible assembly sequence matrix AS, which is composed of assembly part sequence AP, assembly direction sequence AD and assembly tool sequence AT. Random initialization generates AP sequence, which determines the optimal AD and AT sequence.

(2) Initial fitness calculation. According to formula (4), the fitness function value of each particle can be directly calculated, and the initial individual optimal sequence and the initial global optimal sequence can be determined.

(3) Inertia weight calculation. Inertia weight is taken according to formula (9):

$$\omega = m * C_{dt} + n \tag{9}$$

Where $\omega \in [0,1]$, in this paper, m=0.6，n=0.3, and $C_{dt}$ are the target distance factors, which are taken according to the following formula (10):

$$C_{dt} = \begin{cases} 1, f_{gb} \geq f_d \\ |f_{gb} - f_d| / f_d, else \end{cases} \tag{10}$$

Where $f_{gb}$ is the fitness function value of the currently found global optimal assembly sequence, and $f_d$ is the expected fitness function value of the global optimal assembly sequence.

(4) Particle update. The assembly part sequence AP is updated according to formula (5) and formula (6), while the assembly direction sequence AD and assembly tool sequence AT are obtained from the updated AP sequence. The AD and AT sequences are the optimal sequence of the updated AP sequence.

(5) Adaptability update. The fitness value of particle swarm is updated by formula (4), and the individual optimal sequence and global optimal sequence of each particle are updated.

(6) Renewal of diversity factors. The standard deviation of the population fitness value is used as the index to measure the population diversity, which is taken according to the following formula (11):

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f_i - f_a)^2} \tag{11}$$

Where n is the number of particles in the particle swarm, the fitness of the ith particle, and the average fitness of the particle swarm:

$$f_a = \textstyle\sum_{i=1}^{n} f_i / n \tag{12}$$

(7) Variation. In order to avoid precocity, a mutation operator is introduced to make the current global optimal assembly part sequence gBest mutate. The mutation probability (* *) is calculated as follows:

$$p_m = \begin{cases} k, \sigma < \sigma_d, f_{db} \geq f_d, i \neq run \\ 0, else \end{cases} \tag{13}$$

Among them, $k \in [0.1, 0.3]$ and $\sigma_d$ are the critical standard deviation of population convergence, and their values are related to the actual problem, generally far less than the maximum value of $\sigma$, $f_d$ is the expected optimal fitness, i is the current iteration number, and run is the maximum iteration number. When the above mutation conditions are met and $f_{gb} \geq c_f$ is met, the new population randomly generated will replace the old population.

(8) If $i < run$, go to step 3; Otherwise, go to step 9.

(9) Output the best sequence gBest found.

## 4 Comparative analysis of examples

### 4.1 ASP test based on IPSO algorithm

In order to validate the proposed assembly planning algorithm, a product assembly planning oriented to part assembly features was constructed based on Matlab. We select a robotic arm as the object of analysis. The view of the robotic arm assembly is shown in Fig.1, and the assembly tool and direction for each component of the assembly is shown in Tab. 1.

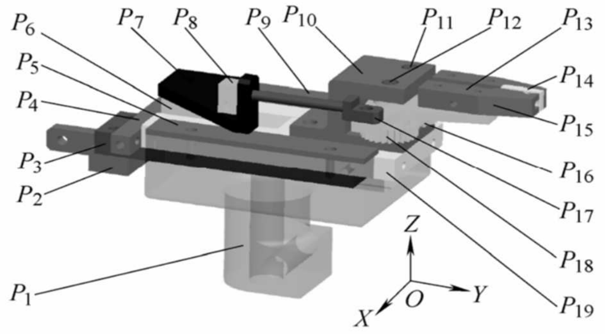

***Fig. 1*** *Structural Diagram of the Mechanical Arm*

***Tab. 1*** *Assembly Tools and Orientation of Parts*

| No. | Direction | Tool |
|---|---|---|
| 1 | *y,z* | T3 |
| 2 | *x,–x* | T2 |
| 3 | *y,z* | T2,T3 |
| 4 | *x,y* | T2 |
| 5 | *z,-x* | T2 |
| 6 | *y* | T2 |
| 7 | *–x* | T2 |
| 8 | *–x* | T2 |
| 9 | *y,z* | T2 |
| 10 | *y,z* | T2,T3 |
| 11 | *y* | T1 |
| 12 | *y* | T1 |
| 13 | *x,y* | T2 |
| 14 | *y,z* | T2 |
| 15 | *x,y,z* | T2 |
| 16 | *x,z* | T2 |
| 17 | *x,y* | T1 |
| 18 | *x,z* | T2 |
| 19 | *y,z* | T2,T3 |

Three algorithms are used to optimize the above problems. Tab.2 shows the allocation results obtained by the three algorithms. For example, the assembly sequence scheme obtained by GA is 1→3→19→6→10→16→2→4→18→13→15→17 →12→11→5→9→14→7→8. According to this scheme, the assembly direction change times is 4, the assembly tool change times is 3, and the assembly stability value is 16.6. Its fitness function value is 2.37. The fitness function values of PSO and IPSO are 2.87 and 2.97, respectively, which are superior to the GA optimization scheme. It can be seen from Tab.1 that IPSO has achieved optimal results in terms of assembly stability, assembly direction change times, and assembly tool change times.

***Tab. 2*** *ASP Scheme Obtained by Three Algorithms*

| | No. | Direction | Tool | No. | Direction | Tool | No. | Direction | Tool |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | *y* | T3 | 1 | *z* | T3 | 1 | *z* | T3 |
| | 3 | *y* | T3 | 19 | *z* | T3 | 3 | *z* | T3 |
| | 19 | *y* | T3 | 10 | *z* | T3 | 19 | *z* | T3 |
| | 6 | *y* | T2 | 16 | *z* | T2 | 5 | *z* | T2 |
| | 10 | *y* | T2 | 3 | *z* | T2 | 10 | *z* | T2 |
| | 16 | *x* | T2 | 18 | *x* | T2 | 9 | *z* | T2 |
| | 2 | *x* | T2 | 6 | *y* | T2 | 18 | *z* | T2 |
| | 4 | *x* | T2 | 11 | *y* | T1 | 16 | *x* | T2 |
| Assembly Information | 18 | *x* | T2 | 17 | *y* | T1 | 2 | *x* | T2 |
| | 13 | *x* | T2 | 12 | *y* | T1 | 13 | *x* | T2 |
| | 15 | *x* | T2 | 13 | *y* | T2 | 4 | *x* | T2 |
| | 17 | *x* | T1 | 14 | *y* | T2 | 14 | *y* | T2 |
| | 12 | *y* | T1 | 4 | *y* | T2 | 6 | *y* | T2 |
| | 11 | *y* | T1 | 15 | *y* | T2 | 17 | *y* | T1 |
| | 5 | *z* | T2 | 9 | *y* | T2 | 11 | *y* | T1 |
| | 9 | *z* | T2 | 2 | –*x* | T2 | 12 | *y* | T1 |
| | 14 | *z* | T2 | 7 | –*x* | T2 | 15 | *y* | T2 |
| | 7 | –*x* | T2 | 5 | –*x* | T2 | 7 | –*x* | T2 |
| | 8 | –*x* | T2 | 8 | –*x* | T2 | 8 | –*x* | T2 |
| Direction Change Times | | 4 | | | 3 | | | 3 | |
| Tool Change Times | | 3 | | | 3 | | | 3 | |
| Stability Value | | 16.6 | | | 17.2 | | | 17.8 | |
| Fitness Function Value | | 2.37 | | | 2.87 | | | 2.97 | |
| Execution Time | | 0.35 | | | 0.26 | | | 0.25 | |

## 4.2 Comparative analysis of IPSO, GA and PSO

In order to verify the effectiveness and superiority of the IPSO algorithm, the IPSO algorithm is compared with the GA and PSO algorithms, which are widely used in the field of ASP. The fitness function and the weight coefficient of each evaluation index remain unchanged. The application program is also written in Matlab, and the program running environment remains unchanged.

The comparison of test results of the IPSO, PSO, and GA are shown in Tab.3. It can be seen from Tab.3 that in bigger population sizes, these three algorithms can produce more feasible assembly sequences. However, when the population size is small, GA will get fewer feasible assembly sequences, and the population size is positively connected with the number of feasible sequences. Furthemore, the IPSO algorithm finds much more global optimal solutions than PSO, indicating that the IPSO method has the strongest ability to find global optimal solutions.

***Tab. 3*** *Comparison of Experimental Results of Three Algorithms*

| | GA | | | | PSO | | | | IPSO | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Population | 20 | 50 | 100 | 200 | 20 | 50 | 100 | 200 | 20 | 50 | 100 | 200 |
| Iterations | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Number of Runs | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| Number of Feasible Assembly Sequences | 6 | 19 | 36 | 62 | 22 | 46 | 65 | 65 | 42 | 65 | 65 | 65 |
| Optimal Fitness Function Value | 1.68 | 2.14 | 2.37 | 2.97 | 1.81 | 2.64 | 2.87 | 2.97 | 1.78 | 2.92 | 2.97 | 2.97 |

IPSO and PSO identify better optimal assembly sequences than GA with the same population size. When the population size is less than 200, GA has never identified the global optimal solution, while PSO and IPSO can find it effectively. In addition, with the same population capacity, the IPSO's running time is substantially shorter than that of the GA. It is clear that the IPSO outperforms the GA in terms of performance and efficiency.

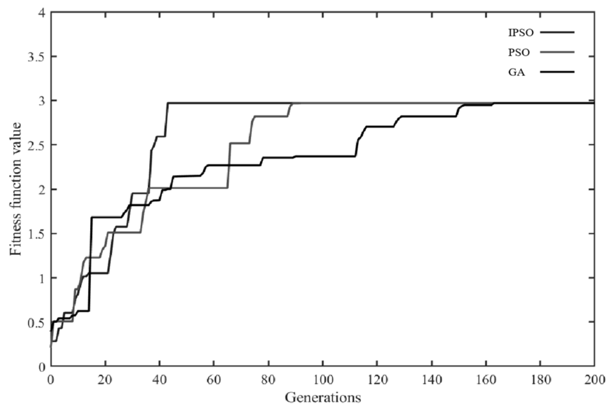

***Fig. 2*** *Average value of optimal fitness of three algorithms*

In order to validate the effectiveness and superiority of the IPSO, several experiments were carried out to compare the IPSO with GA and PSO. Fig. 2 depicts how the average optimal function value changes as the number of iterations rises. It can be seen from Fig. 2 that GA has a slower search speed and require a longer algebra to locate the ideal answer. Compared to PSO and GA, IPSO can identity the optimal solution in a relatively short time, and the shortest path searched by IPSO is better than GA. The IPSO, on the other hand, can yield greater results.

Take the standard deviation of the fitness function value as an indicator to measure population diversity, and compare the change of the mean standard deviation of the fitness function value of the three algorithms with the increase in the number of iterations. The comparison curve of population diversity of the three algorithms is shown in Fig. 3 when the population size is 100 and the number of iterations is 50.

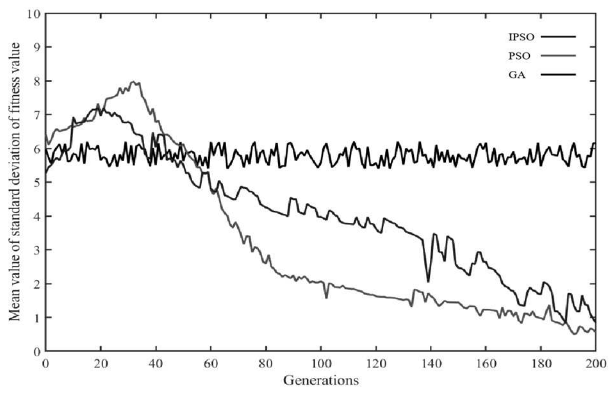

***Fig. 3*** *Comparison of population diversity*

As can be seen from Fig. 3, the diversity of GA has not changed significantly, which indicates that its convergence pace is modest. In the initial iteration stage of the algorithm, IPSO can achieve a faster convergence rate than PSO, but in the later iteration stage, it fluctuates greatly due to the addition of a mutation factor to the IPSO algorithm. When a population falls into a local optimum, the mutation mechanism of the population can improve the diversity of the population, thereby increasing the probability of the algorithm converging to the global optimal solution. From a global perspective, the population diversity of IPSO and PSO algorithms has steadily decreased.

## 5 Conclusion

In this paper, an interference matrix is used to describe information such as a product's viable assembly orientation. On this basis, objective optimization functions are established from assembly stability, the quantity of changes in assembly direction, and the quantity of changes in assembly tools to solve the assembly sequence.

Based on the fundamental PSO algorithm, the particle position, velocity, and their update operations are redefined. The mutation operator in GA is introduced to improve the ability of the algorithm to jump out of the local optimum, and an ASP method based on IPSO is proposed. According to the simulation results, this method has an advantage over GA and basic PSO in that it converges quickly, has significant optimization capabilities, and can successfully avoid approaching the local optimal solution. It can better search the optimal assembly sequence and effectively improve the efficiency and quality of product ASP.

## Acknowledgement

***This work was supported by the Natural Science Foundation of Xiamen (3502Z20227432), Xiamen City University High Level Talent Research Funding Project (G2Q2022-7).***

## References

[1] SU, Q. (2007). Computer aided geometric feasible assembly sequence planning and optimizing. *International Journal of Advanced Manufacturing Technology*, Vol.33, no.1-2, pp.48-57. Springer, London.

[2] B, C. K. A., ANDRÁS KOVÁCS A, & JÓZSEF VÁNCZA A B. (2020). A constraint model for assembly planning. Journal of Manufacturing Systems, Vol.54, pp.196-203. Elsevier B.V., Netherlands

[3] WANG Y., LIU J.H., LI L.S. (2009). Assembly sequence merging based on assembly unit partitioning. International Journal of Advanced ManufacturingTechnology, Vol.45, no.7-8, pp.808-820. Springer, London.

[4] E SUJOVÁ, E STIHAVKOVÁ, & IERNA, H. (2018). An analysis of the assembly line modernization by using simulation software. *Manufacturing Technology*, Vol.18, no.5, pp.839-845.

[5] SAGAN, M. (2018). Importance of holistic approach of assembly production transformation in manufacturing with value stream mapping. *Manufacturing Technology*, Vol.18, no.1, pp.112-116.

[6] WANG, D., SHAO, X., LIU, S. (2017). Assembly sequence planning for reflector panels based on genetic algorithm and ant Colony optimization. *International Journal of Advanced Manufacturing Technology*, Vol.91, no.1-4, pp.987-997.

[7] MISHRA, A. & DEB, S. (2016) Assembly sequence optimization using a flower pollination algorithm-based approach. *Journal of Intelligent Manufacturing*, Vol.30, no.2, pp.461-482. Springer Science and Business Media, LLC

[8] WU, Y., CAO, Y., WANG, Q. (2019), Assembly sequence planning method based on particle swarm algorithm. *Cluster Computing*, Vol.22, pp.835-846. Springer Science and Business Media, LLC

[9] XIN, L., JIANZHONG, S., YUJUN, C. (2017). An efficient method of automatic assembly sequence planning for aerospace industry based on genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, Vol.90, no.5-8, pp.1307-1315. Springer, London.

[10] GUNJI, B., DEEPAK, B., BAHUBALENDRUNI, M., et al. (2017). Hybridized genetic-immune based strategy to obtain optimal feasible assembly sequences. *International Journal of Industrial Engineering Computations*, Vol.3, no.333, pp.333-346. Growing science, Canada.

[11] GUNJI, A. B., DEEPAK, B. B. B. V. L., BAHUBALENDRUNI, C., et al. (2018). An optimal robotic assembly sequence planning by assembly subsets detection method using teaching learning-based optimization algorithm. *IEEE Transactions on Automation Science & Engineering*, pp.1-17. Institute of Electrical and Electronics Engineers Inc., United States

[12] MARTOWIBOWO, S.Y., & DAMANIK, B. K. (2021). Optimization of material removal rate and surface roughness of aisi 316l under dry turning process using genetic algorithm. *Manufacturing Technology,* Vol.21, no.3, pp.373-380.

[13] WU, Y., XU, Y., LUO, L., et al. (2018). Research on evolution balancing for product family assembly line in big data environment. *Manufacturing Technology*, Vol.18, no.2, pp.337-342.

[14] WU, Y., LIN, S., DAI, L., ET AL. (2017). Dynamic balancing for mixing assembly line based on pso - ga cooperative optimization. *Manufacturing Technology*, Vol.17, no.4, pp.622-628.

[15] LAZZERINI, B., & MARCELLONI, F. (2000). Genetic algorithm for generating optimal assembly plans. *Artificial Intelligence in Engineering*, Vol.14, no.4, pp.319-329. Elsevier Science Ltd, Exeter, United Kingdom

[16] CHEN, R.S., LU, K.Y., & YU, S.C. (2002). A hybrid genetic algorithm approach on multi-objective of assembly planning problem. *Engineering Applications of Artificial Intelligence*, Vol.15, no.5, pp.447-457. Elsevier B.V., Netherlands.

[17] ZHAO, L. (2011). Assembly sequence concomitant planning method based for the variation of constraint. *Journal of Mechanical Engineering*, Vol.47, no.5, pp.149-155. Editorial Office of Chinese Journal of Mechanical Engineering, China.

[18] MARIAN, R. M., LUONG, L., & ABHARY, K. (2006). A genetic algorithm for the optimisation of assembly sequences. *Computers & Industrial Engineering*, Vol.50, no.4, pp.503-527. Elsevier B.V., Netherlands.

[19] LUONG, L., MARIAN, R. M., & ABHARY, K. (2005). Assembly sequence optimization using genetic algorithms. Springer, US.

[20] ZHANG, W., LIANG, H., GAN, Y., ET AL. (2019). Parallel optimization of the balancing and sequencing for mixed-model assembly lines. *Manufacturing Technology*, Vol.19, no.3, pp.537-544.