mathematics

MDPI

Article

An Assembly Sequence Planning Method Based on Multiple Optimal Solutions Genetic Algorithm

Xin Wan, Kun Liu *, Weijian Qiu and Zhenhang Kang

School of Naval Architecture and Ocean Engineering, Jiangsu University of Science and Technology, Zhenjiang 212003, China; wanxin_1001@163.com (X.W.); moxin_221@163.com (W.Q.); kangzh0127@163.com (Z.K.)
* Correspondence: kunliu@just.edu.cn; Tel.: +86-135-1169-2085; Fax: +86-0511-8444-6543

Abstract: Assembly sequence planning (ASP) is an indispensable and important step in the intelligent assembly process, and aims to solve the optimal assembly sequence with the shortest assembly time as its optimization goal. This paper focuses on modular cabin construction for large cruise ships, tackling the complexities and challenges of part assembly during the process, based on real engineering problems. It introduces the multiple optimal solutions genetic algorithm (MOSGA). The MOSGA analyzes product constraints and establishes a mathematical model. Firstly, the traditional genetic algorithm (GA) is improved in the case of falling into the local optimum when facing complex problems, so that it can jump out of the local optimum under the condition of satisfying the processing constraints and achieve the global search effect. Secondly, the problem whereby the traditional search algorithm converges to the unique optimal solution is solved, and multiple unique optimal solutions that are more suitable for the actual assembly problem are solved. Thirdly, for a variety of restrictions and emergencies that may occur during the assembly process, the assembly sequence flexible planning (ASFP) method is introduced so that each assembly can be flexibly adjusted. Finally, an example is used to verify the feasibility and effectiveness of the method. This method improves the assembly efficiency and the diversity of assembly sequence selection, and can flexibly adjust the assembly sequence, which has important guiding significance for the ASP problem.

Keywords: assembly sequence planning; multiple unique optimal solutions; multi-objective optimization; flexible planning

MSC: 90-04

Citation: Wan, X.; Liu, K.; Qiu, W.; Kang, Z. An Assembly Sequence Planning Method Based on Multiple Optimal Solutions Genetic Algorithm. *Mathematics* **2024**, *12*, 574. https://doi.org/10.3390/math12040574

Academic Editor: Carlos Conceicao Antonio

Received: 17 January 2024
Revised: 7 February 2024
Accepted: 9 February 2024
Published: 14 February 2024

1. Introduction

Assembly sequence planning (ASP) holds paramount significance in the realm of manufacturing and production. It aims to determine the optimal assembly sequence, seeking to minimize the time required for the assembly process. Thoughtful ASP can effectively reduce production cycles and enhance productivity, consequently lowering costs and increasing production capacity [1]. Rational ASP contributes to optimized resource utilization, ensuring the efficient scheduling and utilization of parts and tools during the assembly process [2]. This aids in reducing unnecessary wait times and resource wastage, thereby improving resource utilization efficiency. Well-planned assembly sequences also play a crucial role in lowering the probability of assembly errors, enhancing accuracy and consistency, and ultimately elevating product quality, reducing defect rates, and strengthening product competitiveness [3].

The design and construction of large cruise ship cabins have always been a focus of research in the field of high-tech passenger ship construction. For the construction process of prefabricated cabins, the main task of ASP is to find all feasible assembly sequences and optimize the feasible sequences according to certain assembly objectives, ultimately providing the best sequence. This paper studies the ASP problem based on the actual engineering problems of constructing modular cabins for large cruise ships. ASP

is a complex NP-hard problem [4], particularly challenging when dealing with a large number of product parts, leading to a phenomenon known as "combinatorial explosion", which significantly increases the difficulty of solving the problem [5]. Numerous scholars have employed various search algorithms, such as genetic algorithms (GA), ant colony optimization (ACO), heuristic algorithms (HA), simulated annealing (SA), and particle swarm optimization (PSO), to tackle the ASP [6–11]. Tseng, H.E. et al. [12] proposed a hierarchical classification method that restructures the initial assembly relationships into a new hierarchical structure, aiming to achieve efficient assembly planning. They also introduced an improved ACO based on three assembly principles to search for the optimal assembly sequence of parts. However, their hierarchical classification method is only suitable for assembly products that are easily divided into hierarchical structures. In contrast, complex assembly structures pose challenges in hierarchical division, making this method unsuitable for solving ASP problems in intricate scenarios. Mishra, A. et al. [13] presented an assembly sequence optimization method based on the flower pollination algorithm (FPA). While it can yield multiple optimal solutions, it fails to achieve stable convergence. After thousands of population iterations, it may not necessarily converge to the optimal value, with instances of the fitness function continuing to increase at the end of the iteration. This inability to ascertain global optimality, coupled with a slower convergence speed and significant computational costs, diminishes the effectiveness of the method. Wang, Z.Y. et al. [14] proposed an ASP method tailored for discrete manufacturing, ensuring an optimal assembly sequence. They utilized a non-dominated sorting GA with a mixed chromosome coding mechanism for the solution. However, their consideration of a limited number of constraint variables results in poor convergence performance, making it less applicable to real-world machining scenarios.

ASP is a complex optimization challenge that typically entails a vast array of possible sequences and constraints. In solving such problems, GAs hold several advantages over other approaches like HA, ACO, PSO, and SA: Firstly, GAs sustain population diversity through crossover and mutation operations, enhancing the comprehensive exploration of the solution space and reducing the likelihood of converging to local optima [15]. Secondly, GAs do not directly use the actual values of decision variables for optimization calculations but operate on them in encoded form. It is through this encoding approach that GAs draw inspiration from biological concepts such as chromosomes and genes, mimicking the genetic and evolutionary mechanisms to solve problems [16]. Thirdly, GAs boast strong adaptability: they are not constrained by the specific mathematical formulation of a problem, rendering them applicable to a wide array of complex optimization scenarios, including nonlinear, multi-objective, and multi-constraint types [17]. Fourthly, the operations of GAs, such as selection, crossover, and mutation, can be tailored to the specific attributes of a problem, offering a high degree of flexibility and customization [18]. Lastly, GAs are robust: they exhibit minimal dependency on initial parameters and solutions, demonstrating substantial robustness and stability [19,20].

In the construction process of modular cabins for large cruise ships, GAs can adapt to changes in problem specifications or constraints without requiring significant modifications to the algorithm structure. This characteristic is particularly valuable in engineering problems where design parameters or constraints may evolve over time [21]. Moreover, many practical engineering problems involve optimizing multiple conflicting objectives simultaneously. GAs can be extended to multi-objective optimization, enhancing their scalability. This scalability is crucial for addressing the multidimensionality and multifaceted nature of engineering problems.

However, GAs also have some limitations. The selection operation in GAs includes three methods: roulette wheel selection, tournament selection, and an elite preservation strategy [22]. During the selection operation, the number of selected optimal assembly sequences only represents a small fraction of the population size [23]. Retaining too many assembly sequences with high fitness values can impact genetic diversity. This may lead to a situation where a local optimal individual is not easily eliminated, reducing the global

search efficiency of the algorithm. In complex data input scenarios, the occurrence of local optima is highly likely [24–27]. This paper first establishes the hierarchical model of the product, illustrating the constraints and corresponding relationships between assembly parts, as shown in Figure 1a. Subsequently, a mathematical model is constructed based on the constraints of the product, with the product's image and mathematical model depicted in Figure 1b. Details of the construction of the mathematical model are presented in Section 2. Through enhancements to the GA, the search space is expanded, improving search efficiency and enabling the solution to complex assembly problems to break free from local optima, achieving global optimality. The improved solving process is depicted in Figure 1c.

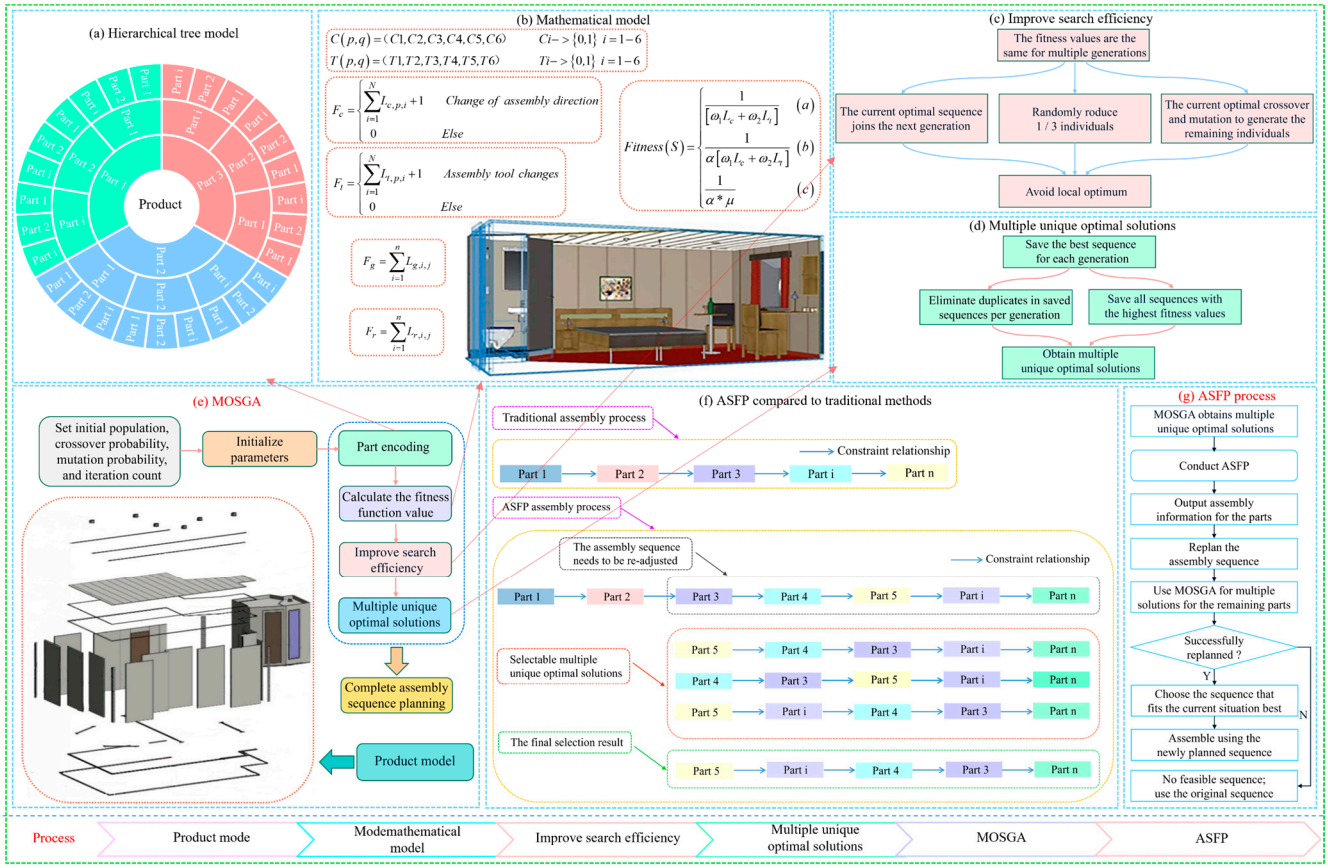


Figure 1. The overall framework.

Furthermore, GAs, ACO and other search algorithms have certain shortcomings. These algorithms often converge to a single optimal solution, lacking diversity in the solutions [28–33]. Products with multiple parts have a vast solution space during the assembly process, where there may be more than one optimal assembly sequence. Various unexpected situations may arise during assembly, and a single assembly sequence may not be sufficient to handle these contingencies. Therefore, it is not suitable for complex assembly scenarios. This paper addresses these limitations by ensuring the generation of multiple unique optimal solutions in a single solving process. This improvement guarantees the diversity of solutions, as depicted in Figure 1d. This is particularly crucial in the assembly process, where there are multiple optimal assembly sequences to choose from, aligning more closely with real-world assembly scenarios. Figure 1e illustrates the solving process of the MOSGA and the model diagram of the product.

In practical production processes, the production workshop is not an idealized production environment. Assembly parts are diverse, and the processes are complex. There is a significant probability of uncertainties such as delayed material deliveries, labor short-

ages, and equipment damage [34–36]. Therefore, ensuring the continuous fulfillment of production tasks in the workshop becomes a crucial issue. If a partial shortage of assembly parts occurs after assembling some parts of the assembly, it is necessary to flexibly plan and adjust the optimal assembly sequence based on the constraints of the already assembled parts. This involves constructing an adjustable flexible assembly sequence to cope with these uncertainties and complete production tasks within the specified time. Hence, there is a need for an assembly sequence flexible planning (ASFP) method, designed to tackle practical re-optimization issues. This approach adjusts and plans the assembly of subsequent parts based on the parts that have already been partially installed, effectively accommodating changes and updates in the assembly process. Figure 1f compares ASFP with traditional algorithms, demonstrating its advantages. Figure 1g delineates the detailed solving process of ASFP.

To address these issues, the development of an efficient and reliable ASP method aimed at solving the construction problems of modular cabins for large cruise ships is proposed. This method also seeks to reduce product construction costs and improve construction quality and efficiency. The objectives of this study are as follows: (1) The MOSGA is proposed to improve the computational efficiency of solving complex assembly problems, increase the search space, avoid falling into the local optimum, and determine the global optimum of the assembly sequence. (2) In the assembly sequence planning process, we aim to obtain multiple unique optimal assembly sequences in a single solving process. (3) The introduction of the AFSP method effectively addresses the challenge of reconfiguring assembly sequences when encountering additional constraints. This approach adeptly manages uncertainties within the assembly process, thereby solving the practical re-optimization issues in assembly operations.

2. Mathematical Model of Assembly Sequence

ASP is the process of solving optimization problems that meet spatial geometric relationships, physical relationships, and mechanical condition constraints. Its goal is to avoid interference between parts while minimizing the waste of resources and time as much as possible [37–39]. In the complex task of assembly, the establishment of the assembly information mathematical model is crucial, as it directly affects whether the subsequent generated assembly sequence is applicable to practical production issues. The design of the mathematical model is essential for ensuring the orderliness and efficiency of the assembly process, and its quality directly determines the applicability of the generated assembly sequence in actual production. When establishing the mathematical model, it is necessary to clearly reflect the information of the product's parts, including their relationships [40,41]. Additionally, consideration of information about assembly tools is essential to ensure the feasibility of the planned sequence in practice. Furthermore, coverage of information regarding various assembly operations is also necessary to ensure the rationality and smoothness of the entire assembly process [42–44].

2.1. Design of Constraint Matrix

Assume that the parts of the product undergo horizontal motion in six directions in three-dimensional space, corresponding to the six coordinate axes of the Cartesian coordinate system [45]. The variables $C(p,q)$ and $T(p,q)$ are introduced to represent the contact and motion interference relationship between two arbitrary parts in the six directions in space, denoted as $(x, y, z, -x, -y, -z)$. The defining expressions for the two variables are as follows:

$$\begin{array}{ll} C(p,q) = (C1, C2, C3, C4, C5, C6) & Ci- > \{0,1\}\ i = 1-6 \\ T(p,q) = (T1, T2, T3, T4, T5, T6) & Ti- > \{0,1\}\ i = 1-6 \end{array} \quad (1)$$

In Equation (1), Ci represents the contact relationship between two parts in the i direction. If $Ci = 1$, it indicates that part q is in contact with part p in the i direction of p. Otherwise, if $Ci = 0$, it signifies that there is no contact between the two parts in that

direction. The motion contact relationships are illustrated in Table 1. Similarly, Ti represents the interference relationship when one part moves relative to another in the i direction. If $Ti = 1$, it means that part q can move in the i direction relative to p without interference from p. Conversely, if $Ti = 0$, it implies that when q moves in that direction to complete the assembly operation, it will encounter interference from p. The motion interference relationships are depicted in Table 2.

Table 1. Motion contact relationship.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
A	(0,0,0,0,0,0)	(0,1,0,0,1,0)	(0,0,0,0,0,0)	(0,0,0,0,0,0)	(0,0,1,0,0,0)	(0,0,0,1,0,0)	(0,0,1,0,0,0)	(0,1,0,0,0,0)
B	(0,1,0,0,1,0)	(0,0,0,0,0,0)	(0,0,0,1,0,0)	(0,0,0,0,0,1)	(0,0,0,0,0,0)	(0,0,1,0,0,0)	(0,1,0,0,0,0)	(0,0,0,0,0,0)
C	(0,0,0,1,0,0)	(1,0,0,0,0,0)	(0,0,0,0,0,0)	(0,0,0,0,0,0)	(0,0,0,0,0,0)	(0,0,0,0,0,1)	(0,0,0,0,0,1)	(0,0,1,0,0,0)
D	(0,0,0,0,0,0)	(0,0,0,0,1,0)	(0,0,0,0,0,0)	(0,0,0,0,0,0)	(0,0,0,0,0,0)	(0,0,0,0,0,0)	(0,0,0,0,0,0)	(0,0,0,0,0,0)
E	(0,0,1,0,0,0)	(0,0,0,0,0,0)	(0,0,1,0,0,0)	(0,0,0,1,0,0)	(0,0,0,0,0,0)	(1,0,0,0,0,0)	(0,0,0,0,0,0)	(0,1,0,0,0,0)
F	(0,0,0,0,1,0)	(0,0,0,0,0,0)	(1,0,0,0,0,0)	(0,1,0,0,0,0)	(0,0,0,1,0,0)	(0,0,0,0,0,0)	(0,1,0,1,0,0)	(0,1,0,1,1,1)
G	(0,0,0,0,0,0)	(0,0,0,0,0,0)	(0,0,0,0,0,0)	(0,0,0,0,0,1)	(0,1,0,0,1,0)	(1,1,1,0,0,0)	(0,0,0,0,0,0)	(1,1,0,0,0,0)
H	(0,0,0,0,0,0)	(0,0,0,0,0,0)	(0,0,0,0,0,1)	(0,0,0,0,0,0)	(1,0,0,1,0,1)	(0,1,0,1,0,0)	(0,0,0,0,0,0)	(0,0,0,0,0,0)

Table 2. Motion interference relationship.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
A	(1,1,1,1,1,1)	(0,0,1,0,0,1)	(1,0,1,1,1,0)	(1,1,1,1,1,0)	(1,1,0,0,1,1)	(1,1,1,0,1,1)	(1,1,0,1,1,1)	(1,0,1,1,1,1)
B	(0,0,1,0,0,0)	(1,1,1,1,1,1)	(1,1,1,0,0,0)	(1,1,1,1,0,0)	(1,0,1,1,1,1)	(1,1,0,1,1,1)	(1,0,1,1,1,1)	(1,1,1,1,1,1)
C	(0,0,0,0,0,0)	(0,1,0,1,1,1)	(1,1,1,1,1,1)	(1,1,1,1,0,0)	(1,1,0,1,1,1)	(1,1,1,1,1,0)	(1,1,1,1,1,0)	(1,1,0,1,1,1)
D	(0,0,0,0,0,0)	(1,1,1,0,0,0)	(0,1,1,0,0,0)	(1,1,1,1,1,1)	(0,1,1,1,1,1)	(1,1,1,1,1,1)	(1,1,1,1,1,1)	(1,1,1,1,1,1)
E	(0,0,0,1,0,0)	(0,1,1,1,1,0)	(1,1,0,0,1,0)	(1,0,1,0,0,1)	(1,1,1,1,1,1)	(0,1,1,1,1,1)	(1,1,1,1,1,1)	(1,0,1,1,1,1)
F	(1,1,1,0,0,0)	(0,1,0,1,1,1)	(0,1,1,1,1,1)	(1,0,0,1,1,1)	(1,1,0,0,1,1)	(1,1,1,1,1,1)	(1,0,1,0,1,1)	(1,0,1,0,0,0)
G	(1,1,1,0,1,1)	(0,1,0,1,0,1)	(1,1,1,1,0,1)	(1,1,1,0,0,0)	(0,0,1,1,0,1)	(1,0,0,1,1,1)	(1,1,1,1,1,1)	(0,0,1,1,1,1)
H	(0,1,1,1,1,1)	(1,1,1,0,1,1)	(1,1,1,0,1,0)	(1,1,1,0,1,0)	(0,1,1,0,1,0)	(1,0,1,0,1,1)	(1,1,1,0,1,1)	(1,1,1,1,1,1)

2.2. Number of Assembly Direction Changes

This section introduces several mathematical symbols to aid in describing the established mathematical model. The symbols are as follows:

N: The quantity of parts in the product.

L: The assembly sequence of the product.

P_i: The i-th assembly part in the assembly sequence.

$L_{c,p,i}$: The number of direction changes when assembling P_i parts.

F_c: The number of assembly direction changes.

$$F_c = \begin{cases} \sum_{i=1}^{N} L_{c,p,i} + 1 & Change\ of\ assembly\ direction \\ 0 & Else \end{cases} \quad (2)$$

In Equation (2), during the assembly of a product, the assembly directions of the parts are determined based on the information provided in Table 1. When the assembly direction changes between two adjacent assembly parts, it incurs a certain amount of time and labor cost [46–48]. Each change in assembly direction increases the reversal count $L_{c,p,i}$ of the assembly. The reversal count reflects the complexity of the assembly sequence operations. The initial F_c value is 0. When the assembly changes direction once, the value of $L_{c,p,i}$ increases by 1. A smaller value of F_c indicates lower time costs.

2.3. Number of Assembly Tool Changes

$L_{t,p,i}$: The number of assembly tool changes when assembling P_i parts.
F_t: The number of assembly direction changes.

$$F_t = \begin{cases} \sum_{i=1}^{N} L_{t,p,i} + 1 & Assembly\ tool\ changes \\ 0 & Else \end{cases} \tag{3}$$

In Equation (3), within the assembly sequence, when considering two consecutive assembly parts, denoted as P_i and P_j, the utilization of the same assembly tool for both parts contributes to improved assembly efficiency [49,50]. If the assembly tool changes, the efficiency decreases. By tracking the number of tool changes $L_{t,p,i}$, we can reflect the efficiency of the assembly sequence. The initial value of F_t is 0. When the assembly tool changes between two adjacent assembly parts, the value of $L_{t,p,i}$ increases by 1. A smaller value of F_t indicates higher assembly efficiency.

2.4. Geometric Constraints of Assembly

$L_{g,i,j}$: Whether the parts P_i and P_j satisfy the geometric constraint relationship.
F_g: Geometric constraints of assembly.

$$L_{g,i,j} = \begin{cases} 1 & Mounting\ part\ P_j\ after\ part\ P_i\ does\ not\ satisfy\ the\ geometric\ constra\text{int} \\ 0 & Else \end{cases} \tag{4}$$

$$F_g = \sum_{i=1}^{n} L_{g,i,j} \tag{5}$$

In Equations (4) and (5), within the assembly process, as indicated by the information presented in Tables 1 and 2, if assembly parts P_i and P_j cannot be assembled in practice due to mutual interference between the assembly bodies, the feasibility conditions for assembly are not met [51,52]. As long as there is one situation where the geometric constraints are not satisfied, the entire assembly becomes unfeasible. In the calculation process, penalty coefficients need to be introduced to account for such situations. The feasibility of the assembly sequence can be assessed by tracking whether adjacent parts satisfy the geometric constraints, denoted as $L_{g,i,j}$. The initial value of F_g is set to 0. When there are geometric constraints between two adjacent parts, the value of $L_{g,i,j}$ increases by 1. It is then checked whether F_g is greater than 0; if so, geometric constraints are not satisfied, and a penalty coefficient is applied to the assembly sequence.

2.5. Sequence Constraints of Assembly

$L_{r,i,j}$: Whether the parts P_i and P_j satisfy the sequence constraint relationship.
F_r: Geometric constraints of assembly.

$$L_{r,i,j} = \begin{cases} 1 & Mounting\ part\ P_j\ after\ part\ P_i\ does\ not\ satisfy\ the\ sequence\ constra\text{int} \\ 0 & Else \end{cases} \tag{6}$$

$$F_r = \sum_{i=1}^{n} L_{r,i,j} \tag{7}$$

In Equations (6) and (7), if adjacent parts P_i and P_j do not satisfy the assembly sequence constraint, assembly cannot proceed. Taking the example of assembling compartments in a ship, when installing a sanitation unit and the compartment floor, the sanitation unit cannot be installed before the compartment floor, which does not meet the actual installation requirements. As long as there is one instance of a sequence constraint violation, a penalty coefficient is introduced during the calculation process to prevent the progression to the

next step in the optimal population selection. By assessing whether adjacent parts satisfy the sequence constraint $L_{r,i,j}$, the feasibility of the assembly sequence can be reflected. The initial value of F_r is set to 0, and when two adjacent parts do not meet the sequence constraint, the value of $L_{r,i,j}$ is incremented by 1. It is then checked whether the value of F_r is greater than 0. If it is, the sequence constraint is not satisfied, and a penalty coefficient is applied to the assembly sequence.

2.6. Objective Function and Fitness Function

According to the established mathematical model, the effectiveness of an assembly sequence is influenced by the number of changes in assembly direction, the number of changes in assembly tools, geometric constraints, and sequential constraints. Subject to both geometric and sequential constraints, the objective function F_1 for the assembly sequence can be expressed as Equation (8). Although changes in assembly direction and changes in assembly tools both affect the total assembly time, changing the assembly tools requires more time compared to changing the assembly direction. Therefore, the weight coefficients ω_1 and ω_2 are added, with $\omega_1 = 0.4$ and $\omega_2 = 0.6$.

$$F_1 = \frac{1}{[\omega_1 F_c + \omega_2 F_t]} \tag{8}$$

If the geometric constraints are not met, the assembly process will be hindered by interference between parts, affecting the assembly efficiency. To address this, a penalty coefficient α is introduced, with α taking a value greater than 1, set to $\alpha = 1.8$. This adjustment will result in a lower overall value of the objective function, thus diminishing the competitiveness of the assembly sequence in question. The objective function F_2 is as presented in Equation (9).

$$F_2 = \frac{1}{\alpha[\omega_1 F_c + \omega_2 F_t]} \tag{9}$$

When the sequence constraints are not met, this sequence cannot proceed with assembly and must be directly eliminated. A penalty coefficient μ is added, $\mu = 8$. When α and μ are multiplied, the result of the objective function calculation is an extremely small value, eliminating it from the population evolution. The objective function F_3 is as shown in Equation (10).

$$F_3 = \frac{1}{\alpha \times \mu} \tag{10}$$

In summary, the fitness function is shown in Equation (11). When the assembly sequence satisfies all the constraint conditions, formula (a) is used. Formula (b) is used when the geometric constraints are not satisfied, and formula (c) is used when the sequence constraints are not met. $Fitness(S)$ represents the fitness function of the assembly sequence.

$$Fitness(S) = \begin{cases} \frac{1}{[\omega_1 F_c + \omega_2 F_t]} & (a) \\ \frac{1}{\alpha[\omega_1 F_c + \omega_2 F_t]} & (b) \\ \frac{1}{\alpha \times \mu} & (c) \end{cases} \tag{11}$$

3. Improvement Principle

This section elaborates on how the MOSGA enhances computational efficiency, expands the search space, and avoids falling into local optima when solving complex assembly problems. This approach aims to achieve the global optimum of assembly sequences during the solving process. This section also illustrates the methodology for obtaining multiple unique optimal assembly sequences in a single solving process. Additionally, it outlines the approach of AFSP in implementing flexible planning for assembly sequences.

3.1. Principle Analysis

The traditional GA process includes first setting the population size, and then, selecting the method of selection to be applied from among three selection methods: roulette wheel selection, tournament selection, and elitism retention strategy. After selecting the initial population, a given number of iterations are performed. During these iterations, crossover, mutation, selection, and the evolution of the population occur until the iteration count is reached. A flowchart of the traditional GA process is shown in Figure 2.

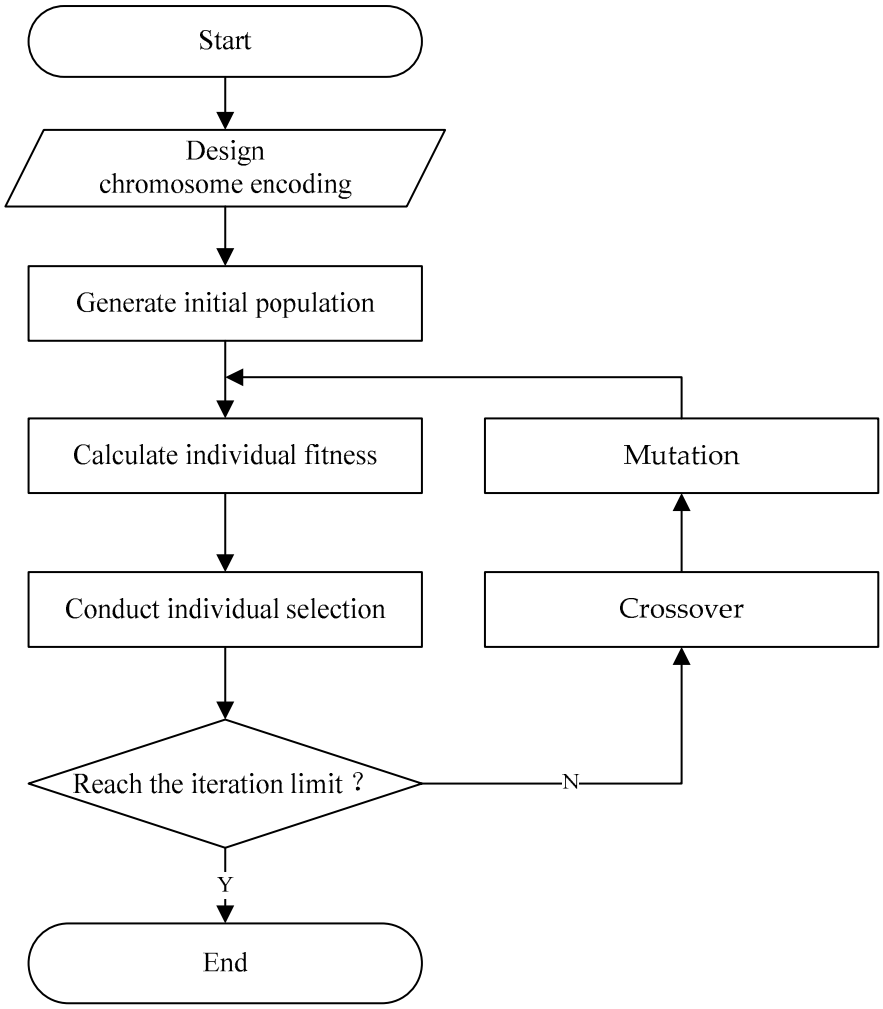


Figure 2. GA flowchart.

1. Feasibility Analysis of Improving Computational Efficiency

Through analysis, when dealing with complex problems, the reasons for traditional GAs getting stuck in local optima include the following: (1) For large-scale problems, there are fewer excellent individuals. The sparse distribution makes it difficult to search. Increasing the number of iterations does not bring significant improvement. This phenomenon occurs due to the complexity of the problem, leading to a larger search space for solutions. (2) When the optimization iterations of the GA reach a certain optimal value, the fitness value is already very high. Individuals with better fitness are sparsely distributed in the search space, making it difficult to conduct further searches. (3) In the process of population evolution, the GA tends to converge, leading to a reduction in population diversity. This decrease in diversity makes it difficult for GAs to mutate and surpass the current local optimum.

The advantages of the MOSGA compared to traditional GAs are as follows: (1) The MOSGA adds new judgment criteria. If the highest fitness function value remains the same for several consecutive generations during the evolution process, optimization is carried out according to the optimization criteria. (2) During optimization, the current best sequence is first preserved to prevent it from being compromised. The best assembly sequence is saved and added to the next generation of the population. (3) Random assembly sequences are introduced to increase the diversity of the solution space. Two individuals are randomly generated as parents for crossover and mutation, and 1/3 of the population is randomly generated to enhance diversity. These individuals are added to the next generation of the population. (4) Evolution is carried out using dominant individuals, increasing the probability of mutation while enhancing global search capability. The current best assembly sequence is used as the parental sequence. In the operations of crossover and mutation, the probability of chromosome crossover and mutation is increased compared to the original probabilities of 0.3 and 0.1, raising the crossover probability to 0.8 and the mutation probability to 0.3, to generate the individuals of the remaining population. This enhances the overall search capability while preventing local optima.

2. Theoretical Feasibility of Multiple Unique Optimal Solutions

GA and other search methodologies often fall short by converging on a single optimal solution, lacking diversity in outcomes. This limitation becomes particularly evident in the assembly of complex parts, where the vast solution space may contain multiple effective assembly sequences. Should the algorithm identify only a singular optimal solution, it may prove insufficient in practical scenarios, especially when unexpected changes or challenges arise. The significance of possessing multiple unique optimal solutions is manifold: (1) Enhanced Adaptability to Complexity: In intricate assembly scenarios, the availability of multiple optimal assembly sequences affords greater flexibility. This implies that in practical operations, should complications arise or specific parts become unavailable, an alternative optimal sequence can be selected to complete the assembly, bypassing the need to start anew or dismantle partially assembled sections in search of a new solution. (2) Increased System Robustness: By ensuring a selection from multiple solutions, the system's adaptability to external changes is bolstered, enabling a more robust response to emergencies. (3) A Closer Reflection of Real-World Conditions: Real-world problems seldom have a single solution. More often, multiple viable approaches exist to achieve a goal. Generating multiple optimal solutions allows algorithms to more accurately mirror the complexity and uncertainty of the real world, offering a method that is both more accurate and practical. (4) Application in Flexible Planning: The concept of multiple unique optimal solutions can also be applied to the ASFP method discussed herein, solving practical re-optimization problems.

In conclusion, the MOSGA presents a significant advancement over traditional GAs by addressing their key limitations, particularly in terms of computational efficiency and the ability to find multiple unique optimal solutions. These improvements make the MOSGA more suitable for complex problem-solving scenarios, offering enhanced adaptability, robustness, and a more accurate reflection of real-world conditions.

3.2. Improving Search Efficiency and Avoiding Local Optima

To address the issue of traditional GAs falling into local optima, the MOSGA employs a new optimization criterion in its selection process. This method aims to avoid entrapment in local optima and further expand the search space for solutions. In the process of solving, the assembly sequence of the highest fitness function value of each generation is saved. However, if the optimal sequence for an assembly remains the same for several consecutive generations, there is a risk of falling into a local optimum. To address this issue, the following optimization steps are implemented:

(1) Save the optimal assembly sequences and incorporate them into the next-generation population.
(2) Randomly generate two individuals as parents, perform crossover and mutation, and randomly generate 1/3 of the population to enhance diversity. Add these individuals to the next-generation population.
(3) Use the current optimal assembly sequence as a parent. In the crossover and mutation operation, increase the probability of chromosome crossover and mutation to generate the remaining population. Increase the crossover probability to 0.8 and mutation probability to 0.3, compared to the original probabilities of 0.3 and 0.1. This enhances the overall search capability of the population, preventing local optima.

Upon completion of these operations, the population size reaches the quantity required for the next-generation evolution. By expanding the search space, the current population's optimal assembly sequences are retained. Experimental validation shows that this approach helps prevent local optima during subsequent evolution. A flow chart of this process is shown in Figure 3.

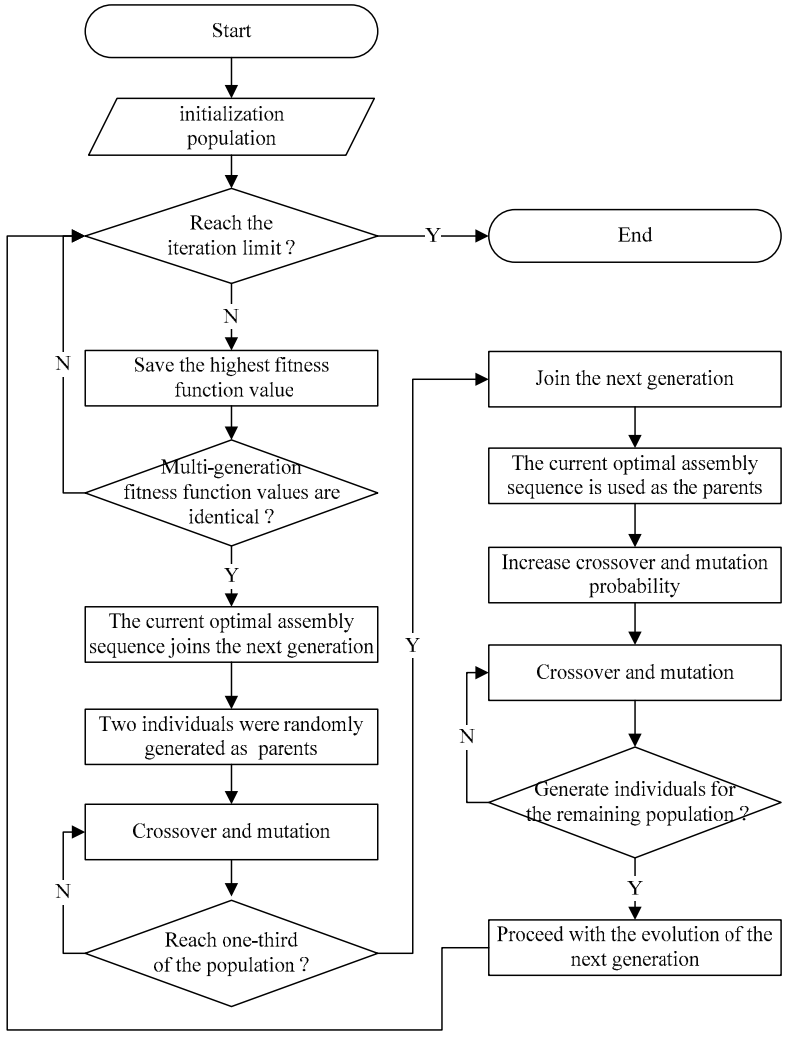


Figure 3. Improving search efficiency and avoiding local optima.

3.3. Establishment of Multiple Unique Optimal Solutions

Establishing multiple unique optimal solutions is not as simple as running the algorithm multiple times to obtain several optimal solutions. Running the algorithm multiple times often leads to convergence towards the same optimal sequence, failing to achieve the uniqueness of optimal solutions and resulting in a substantial waste of computational

resources. The method employed in this study involves saving the assembly sequence with the highest fitness after the completion of each generation's selection. Assuming the number of iterations is denoted as P_n, this results in the preservation of P_n optimal assembly sequences. After the iterations, duplicate assembly sequences are removed from the saved optimal sequences to maintain uniqueness. Subsequently, based on the converged highest fitness function value, sequences below this value are excluded. The final remaining assembly sequences constitute multiple unique optimal solutions. A flow chart of this process is shown in Figure 4.

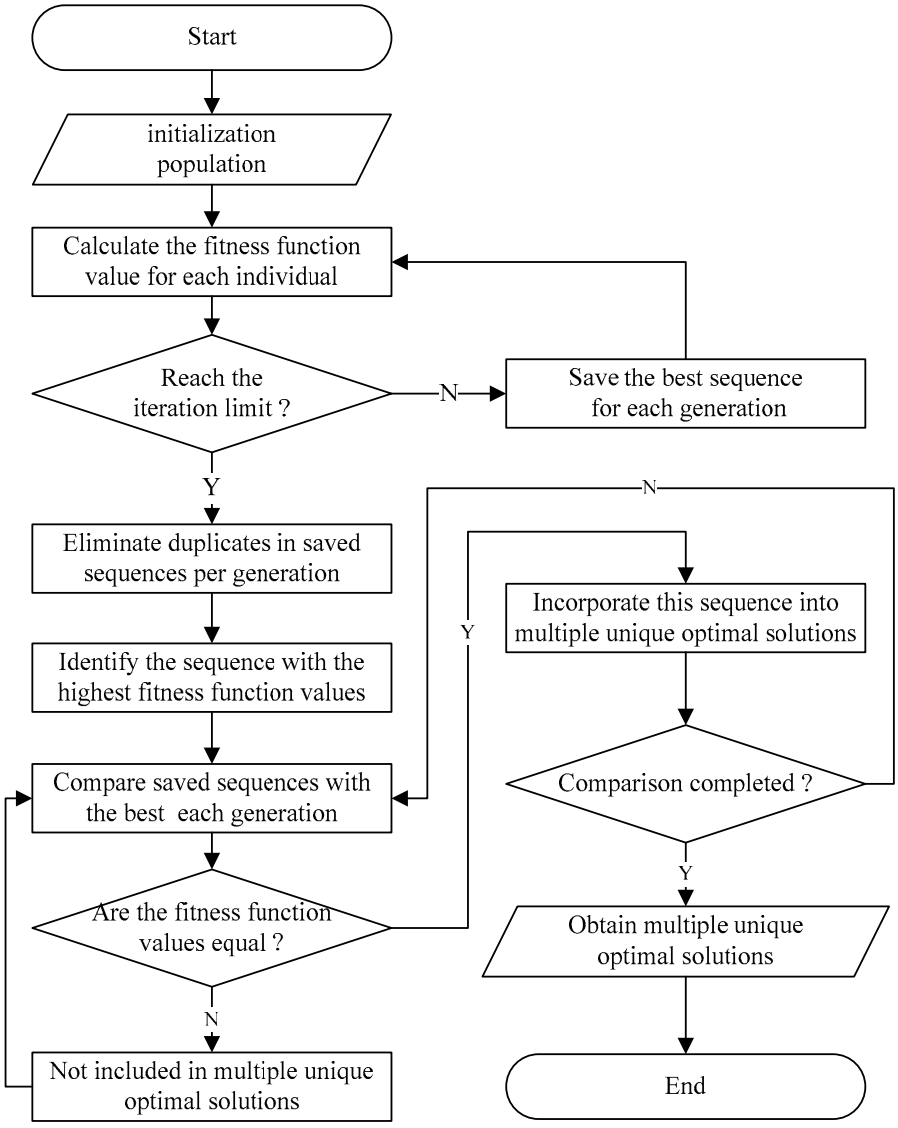


Figure 4. Establishment of multiple unique optimal solutions.

3.4. Assembly Sequence Flexible Planning

3.4.1. Problem Analysis

In the ASP problem, many scholars often focus on finding an optimal assembly sequence for use when production starts in a factory. However, they overlook a crucial issue—adopting a single assembly sequence for production does not provide a one-stop solution to all problems. In actual assembly processes, various issues may arise, such as insufficient part capacity, delivery delays, transportation damage, and occupied assembly tools, preventing the continuation of assembly according to the original sequence.

This results in increased waiting time and total assembly time, leading to manufacturing resource waste.

Addressing the aforementioned challenges, this paper proposes the ASFP method, which tackles the need to readjust assembly sequences when encountering different constraints during the assembly process. The ASFP method can further plan the assembly of remaining parts based on the already installed parts. This planning ensures that under various constraints, an optimal assembly sequence is devised for assembling the remaining parts.

Suppose that at the start of assembly, the initially planned optimal assembly sequence 1, 2, 3, 4, 5, 6, 7, 8, 9 is chosen. However, during the assembly of part 5, certain constraints prevent the continuation of assembly according to the sequence 5, 6, 7, 8, 9. In such a scenario, re-planning of the assembly sequence 5, 6, 7, 8, 9 is required. ASFP can, based on the already installed parts 1, 2, 3, 4, and under various constraints, devise an optimal assembly plan for parts 5, 6, 7, 8, 9. The newly planned assembly sequence satisfies the constraints imposed by the already assembled parts. Among the multiple optimal solutions generated, the one most suitable for practical production is selected.

For example, three optimal assembly sequences are generated: 6, 9, 5, 8, 7; 6, 5, 8, 7, 9; 6, 8, 9, 5, 7. In actual production, due to insufficient capacity for part 5, the sequence 6, 8, 9, 5, 7, where part 5 is assembled later, is selected as the final solution. This addresses engineering challenges and improves assembly efficiency.

3.4.2. Process of ASFP

Assembly sequence flexibility planning poses two main challenges:

1. Constraints of installed parts

The existing assembly sequence has already assembled some parts. The planning of the remaining assembly sequence pertains to the unassembled parts. Therefore, planning the assembly sequence for the remaining parts must satisfy the optimal conditions while adhering to constraints imposed by the already installed parts. For example, if the initially planned optimal assembly sequence is 1, 2, 3, 4, 5, 6, 7, 8, 9 and re-planning is needed during the assembly of part 5, resulting in a sequence like 7, 8, 9, 5, 6, this new sequence must not only satisfy its own constraints but also meet the constraints from the already installed parts 1, 2, 3, 4. These constraints include assembly geometry constraints and sequence constraints.

2. Selecting the optimal assembly sequence

The generated optimal assembly sequences need further optimization under various constraints to make them more suitable for practical applications. For instance, if the initially planned optimal assembly sequence is 1, 2, 3, 4, 5, 6, 7, 8, 9 and flexibility planning is performed during the assembly of part 5, yielding four optimal sequences like 6, 9, 5, 8, 7; 6, 5, 8, 7, 9; 6, 8, 9, 5, 7; 6, 7, 8, 9, because 6, 7, 8, 9 is the original assembly sequence, it is initially excluded. In actual production, if there is a shortage of supply for part 5, then the sequence 6, 8, 9, 5, 7, where part 5 is assembled later, would be selected as the final optimized result.

Based on these challenges, this paper proposes the following solutions:

Assume some parts have already been assembled, making it impossible to follow the original assembly sequence.

1. Utilize the MOSGA: Employ the MOSGA to plan assembly sequences for the remaining parts and generate multiple optimal solutions for further selection.
2. Evaluate constraint satisfaction: Assess the constraint satisfaction of the generated optimal solutions and retain sequences that satisfy the constraints imposed by the already assembled parts.
3. Make a selection: Opt for sequences that align most effectively with the factory's production scenario.

4. Update original assembly sequence: Update the original assembly sequence and proceed with assembly.

A flexibility planning flowchart for ASFP is illustrated in Figure 5.

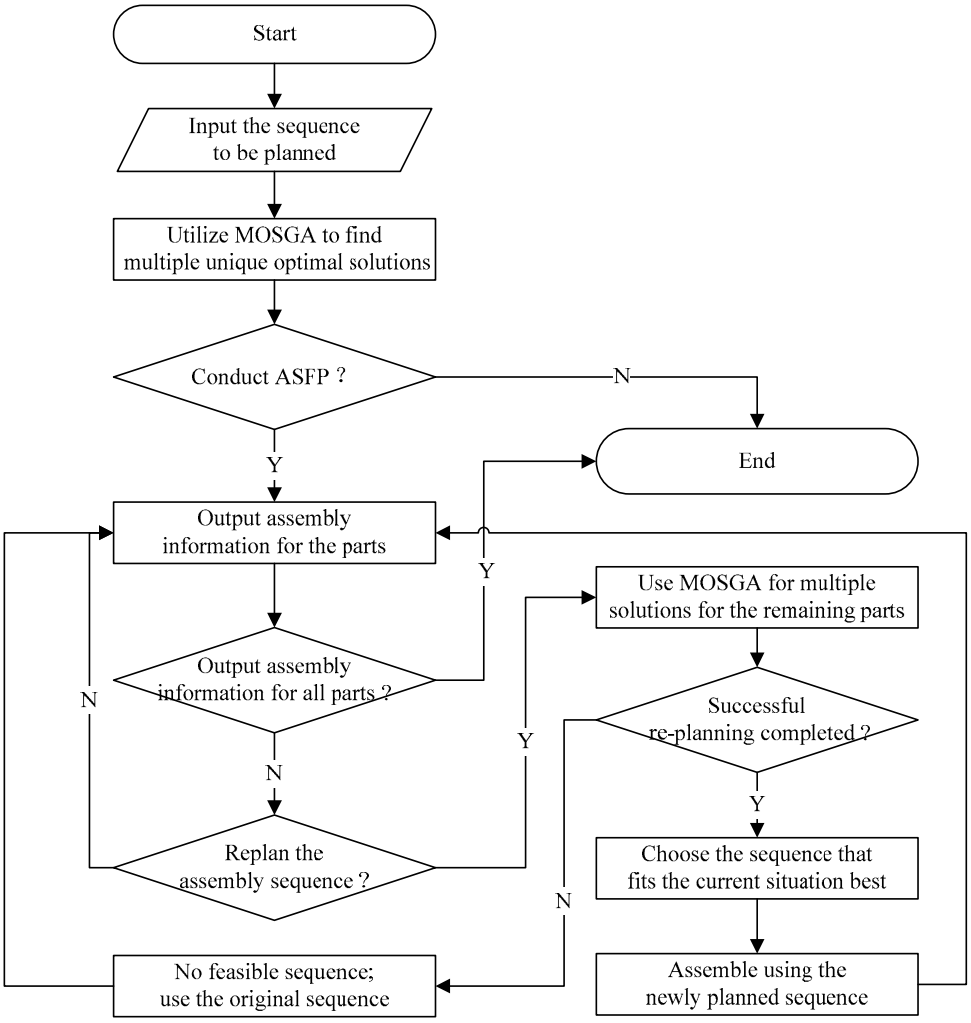


Figure 5. ASFP flow chart.

4. MOSGA and ASFP

The MOSGA is divided into the following steps: Firstly, the parts of the assembly are encoded using a genome-based encoding method in GA. Subsequently, crossover, mutation, and selection are performed to generate multiple unique optimal solutions. The iteration concludes with the completion of the calculations.

4.1. Encodings

In a GA, the method of transforming candidate solutions of a problem from their solution space to the search space that the GA can handle is referred to as encoding. Common encoding techniques include binary encoding, real-valued encoding, and permutation encoding. In this paper, the decimal method in real number coding is used for coding, and each digital coding represents the corresponding parts. Assuming there are N parts in the product, the encoding length is N.

Taking the compartments of a certain ship as an example, the compartments include parts, sanitation units, furniture, ceilings, electrical equipment, fire doors, cables, and more. Each of these parts is assigned a specific code ranging from 1 to N, facilitating an organized and efficient encoding strategy of the MOSGA in this paper.

4.2. Crossover and Mutation

(1) Crossover

The chromosomes were crossed using the following methods. Two father chromosomes, A and B, were randomly selected to cross. We randomly selected the crossover position, assuming that the crossover position selected by A and B was 5–9:

A = 1 2 3 4 | 5 6 7 8 9 | 10 11 12 13 14 15

B = 3 2 4 1 | 9 14 11 15 5 | 10 7 12 13 6 8

The cross-sections 5,6,7,8,9 of A were crossed with the cross-sections 9,12,11,15,5 of B, and the following results were obtained:

A = 1 2 3 4 | 5 6 7 8 9 9 14 11 15 5 | 10 11 12 13 14 15

B = 3 2 4 1 | 9 14 11 15 5 5 6 7 8 9 | 10 7 12 13 6 8

We removed the duplicate parts in A and B in turn, and obtained the following result after crossing:

A = 1 2 3 4 5 6 7 8 9 14 11 15 10 12 13

B = 3 2 4 1 9 14 11 15 5 6 7 8 10 12 13

(2) Mutation

The chromosome was mutated by the reverse mutation method. Two chromosomes, A and B, were selected for variation. We randomly selected the mutation position of 4–9:

A = 1 2 3 | 4 5 6 7 8 9 | 10 11 12 13 14 15

B = 3 2 4 | 1 9 14 11 15 5 | 10 7 12 13 6 8

The mutation part 4, 5, 6, 7, 8, 9 of A was reversed to obtain 9, 8, 7, 6, 5, 4. The variation part 1, 9, 14, 11, 15, 5 of B was inverted to obtain 5, 15, 11, 14, 9, 1. The variation results are as follows:

A = 1 2 3 9 8 7 6 5 4 10 11 12 13 14 15

B = 3 2 4 5 15 11 14 9 1 10 7 12 13 6 8

4.3. Overall Solution and Selection Steps

Step 1: Set the initial population size, crossover probability, and mutation probability. Generate the initial population through random number generation, with the generated population serving as the parent population.

Step 2: Analyze the interference matrix of the assembly, considering the number of assembly reorientations, tool change frequency, geometric constraints of the assembly, and impact of sequence constraints on the assembly. Design the fitness function calculation formula, and calculate the fitness function values of the population individuals according to the fitness calculation formula.

Step 3: Retain the assembly sequence with the highest fitness in each generation to establish multiple unique optimal solutions.

Step 4: Select chromosomes, perform crossover to generate new chromosomes based on crossover probability, and mutate the new chromosomes based on mutation probability to produce further diversity.

Step 5: Evaluate the change in fitness function values over multiple generations. If the fitness values remain unchanged for consecutive generations, optimize the algorithm according to predefined criteria to expand the search space and avoid local optima.

Step 6: Based on the highest fitness function value reached during convergence, discard assembly sequences with fitness values lower than the highest value from the saved

sequences. Retain only those sequences equal to the highest fitness function value to obtain multiple unique optimal solutions.

Step 7: Choose an assembly sequence for assembly from multiple unique optimal solutions.

Step 8: If constraints arise preventing assembly according to the original sequence, utilize ASFP to plan the assembly sequence.

Step 9: Employ the MOSGA to plan multiple unique optimal solutions, select the best one among them, and update the assembly sequence.

Step 10: Complete the product assembly.

An algorithm flowchart is illustrated in Figure 6.

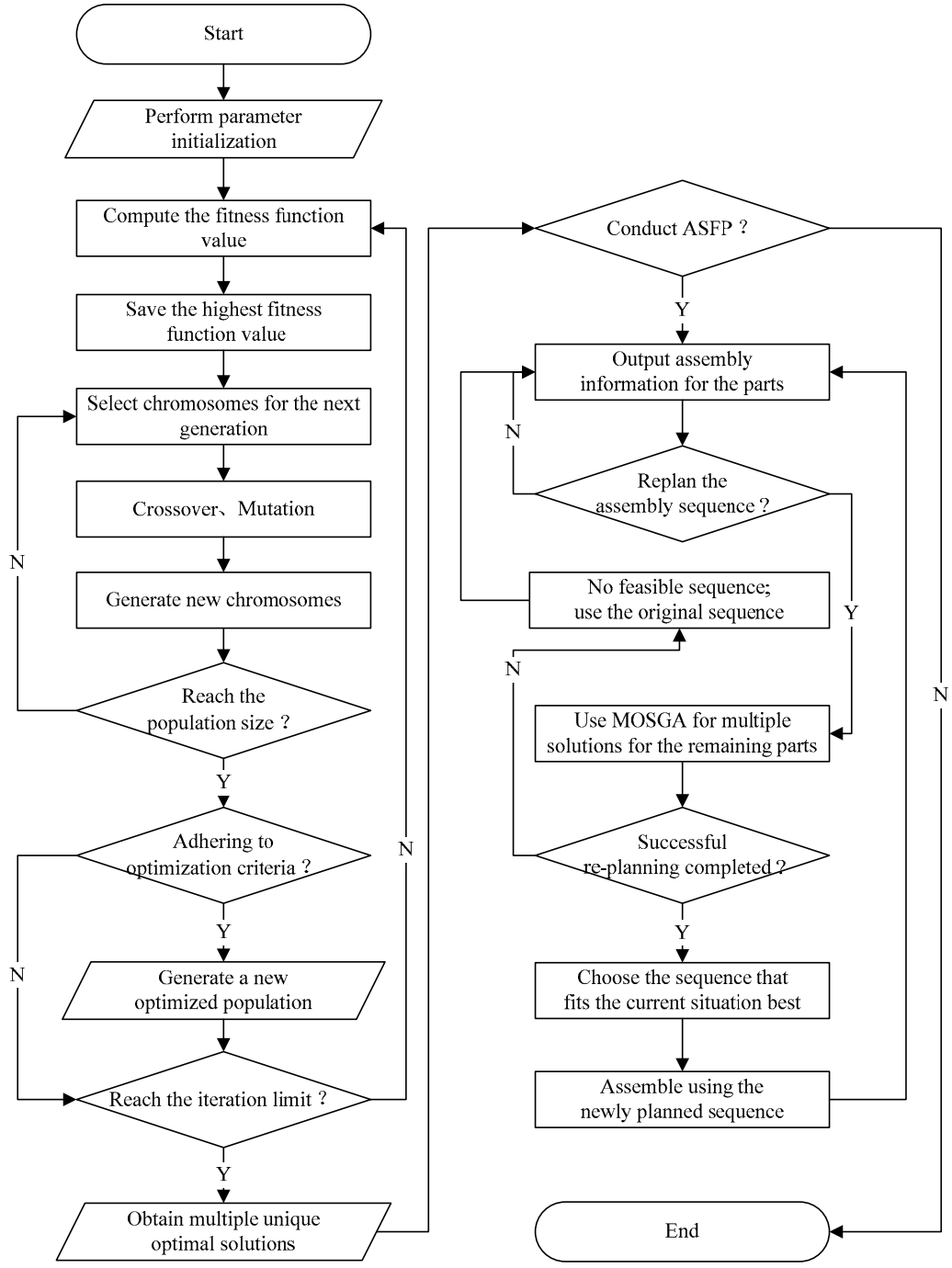
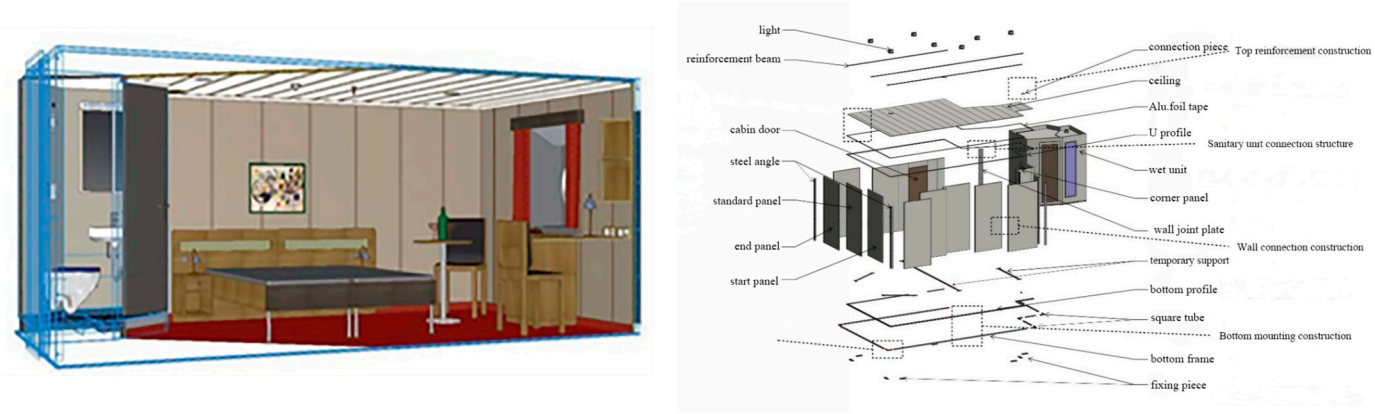


Figure 6. Overall solution and selection steps.

5. Experimental Case Study

Taking a cruise cabin as a case, our analysis reveals that the cabin unit is mainly composed of sanitary units, wall panel systems, ceiling systems, fire doors, furniture systems, ventilation systems, sprinkler systems, and electrical systems. Schematic and model diagrams of the cabin are presented in Figure 7a,b. The installation requires tools such as a cutting machine, a welding machine, an electric drill, a curve saw, an electrician's diagonal pliers, a wire stripper, a screwdriver, an electrician's pen, a multimeter, a cable-bundling tool, an aluminum ladder, a pry bar, a laser level, and a 20 m tape measure. According to the cabin assembly information, there are a total of 15 parts.



(**a**) (**b**)

Figure 7. Cabin pictures. (**a**) Schematic diagram; (**b**) model diagram.

To ensure reliable experimental results, the parameters of both algorithms were standardized. The population size for the GA and MOSGA was set to 200, with a crossover rate of 0.3 and a mutation rate of 0.1, utilizing the tournament selection method, with each tournament's size being half of the population number. Geometric constraints, sequence constraints, number of directional changes, and number of tool changes were all set to identical values. The same fitness function value evaluation strategy was applied for solving. In the small-scale experiment, a simplified cabin composed of 9 parts was used. The large-scale experiment involved a complete cabin comprised of 15 parts. By maintaining consistent parameter settings, data, and evaluation strategies, the aim is to conduct a fair and comparative analysis between the traditional GA and the MOSGA.

5.1. Small-Scale Experiments

In our small-scale experiment, a simplified cabin with nine parts was utilized. The results of solving the GA ten times with 100 iterations and 200 iterations were compared with solving the MOSGA once, all under the same parameters.

5.1.1. GA Solved 10 Times

From Figure 8a, it can be observed that at 100 iterations, when solving small-scale problems, the GA finds the highest fitness function value of 0.2155 in six out of ten attempts, while the rest fall into local optima, as shown in Table 3. Figure 8b indicates that at 200 iterations, among the ten runs of the GA, three runs end up in local optima, as detailed in Table 4. This indicates the instability of the GA when solving simplified small-scale cabin problems. Furthermore, by observing Figure 9a,b, which describe the convergence graphs during the GA solving process, it can be noted that the GA's convergence is not only slow but also unstable, often converging to local optima multiple times.

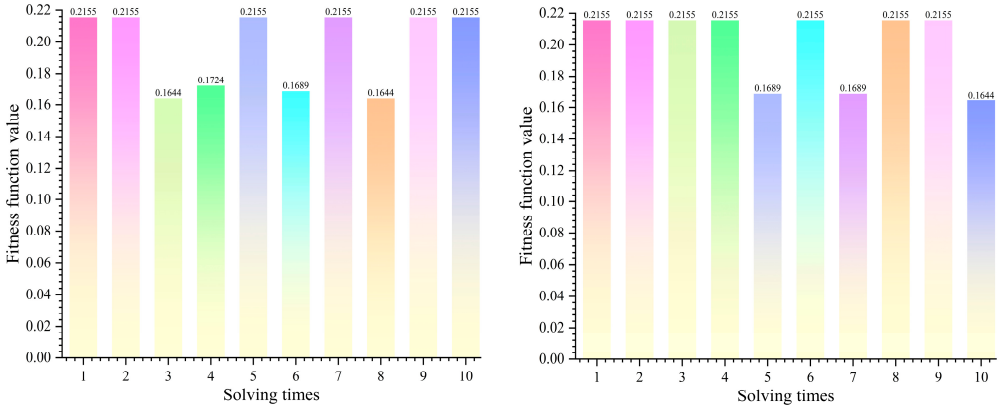


Figure 8. GA solves for the highest fitness function value 10 times, with (**a**) 100 iterations; (**b**) 200 iterations.

Table 3. GA with 100 iterations solving for 10 results.

Number of Times	Optimal Sequence	Fitness Function Value	Number of Direction Changes	Number of Tool Changes	Sequence Constraints	Geometric Constraints	Global Optimal
1	1,2,4,8,7,6,3,9,5	0.2155	3	2	Yes	Yes	Yes
2	1,4,2,6,7,8,3,9,5	0.2155	3	2	Yes	Yes	Yes
3	1,2,8,3,9,7,6,4,5	0.1644	5	3	Yes	Yes	No
4	4,1,2,8,6,7,3,9,5	0.1724	5	4	Yes	Yes	No
5	1,2,4,8,7,6,3,9,5	0.2155	3	2	Yes	Yes	Yes
6	1,2,8,6,7,3,9,5,4	0.1689	4	3	Yes	Yes	No
7	1,2,4,8,6,7,3,9,5	0.2155	3	2	Yes	Yes	Yes
8	1,2,3,9,7,6,8,4,5	0.1644	5	4	Yes	Yes	No
9	1,2,4,8,6,7,3,9,5	0.2155	3	2	Yes	Yes	Yes
10	1,2,4,8,6,7,3,9,5	0.2155	3	2	Yes	Yes	Yes

Table 4. GA with 200 iterations solving for 10 results.

Number of Times	Optimal Sequence	Fitness Function Value	Number of Direction Changes	Number of Tool Changes	Sequence Constraints	Geometric Constraints	Global Optimal
1	1,4,2,8,6,7,3,9,5	0.2155	3	2	Yes	Yes	Yes
2	1,4,2,8,6,7,3,9,5	0.2155	3	2	Yes	Yes	Yes
3	1,4,2,8,7,6,3,9,5	0.2155	3	2	Yes	Yes	Yes
4	1,4,2,7,6,8,3,9,5	0.2155	3	2	Yes	Yes	Yes
5	1,2,7,6,8,3,9,5,4	0.1689	4	3	Yes	Yes	No
6	1,4,2,8,7,6,3,9,5	0.2155	3	2	Yes	Yes	Yes
7	1,2,8,6,7,3,9,5,4	0.1689	4	3	Yes	Yes	No
8	1,4,2,6,7,8,3,9,5	0.2155	3	2	Yes	Yes	Yes
9	1,2,4,6,7,8,3,9,5	0.2155	3	2	Yes	Yes	Yes
10	1,2,8,3,9,7,6,4,5	0.1644	5	3	Yes	Yes	No

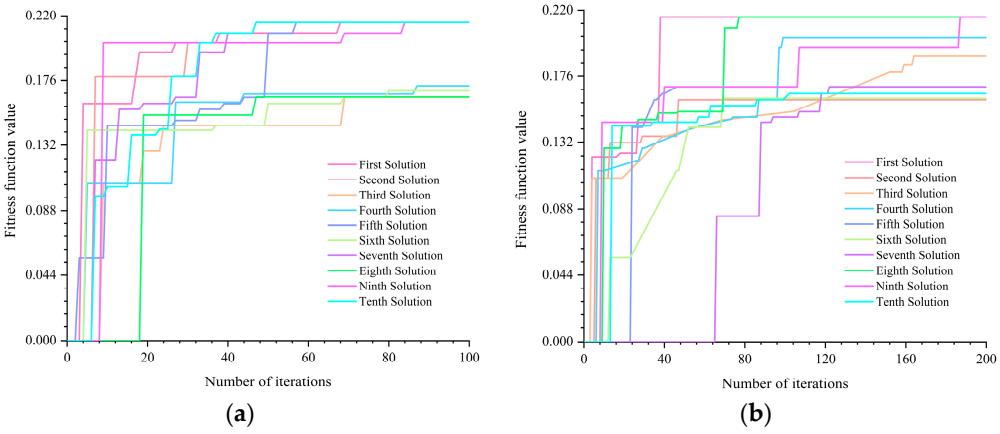


Figure 9. Convergence graphs of GA for solving small-scale problems, with (**a**) 100 iterations; (**b**) 200 iterations.

5.1.2. MOSGA Solved Once

From Figure 10a, it is evident that at 100 iterations, the MOSGA yielded six unique optimal solutions in just one run, as shown in Table 5. Similarly, Figure 10b illustrates that at 200 iterations, the MOSGA produced eight unique optimal solutions in only one run, as detailed in Table 6. The convergence plots in Figure 11a,b demonstrate that the MOSGA rapidly converges to the global optimum. Compared to the GA, the MOSGA not only converges faster but also is capable of finding multiple unique global optimal solutions in a single run, highlighting its advanced capabilities.

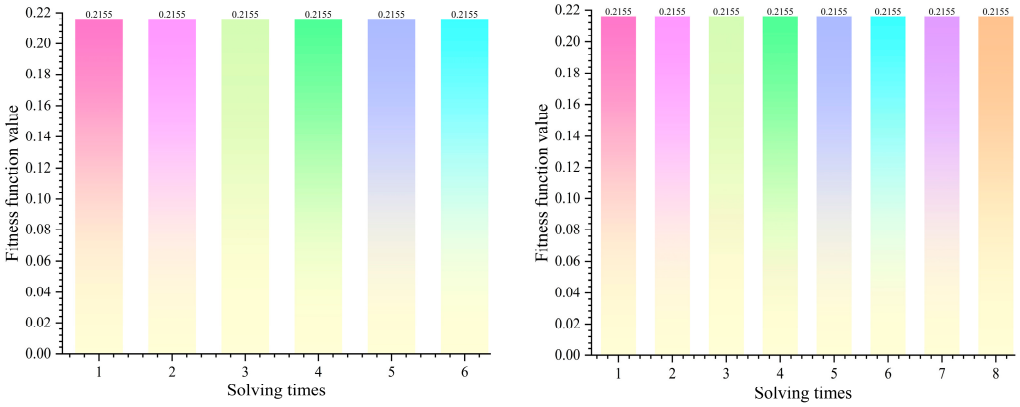


Figure 10. MOSGA solving for the highest fitness function value 1 time, with (**a**) 100 iterations; (**b**) 200 iterations.

Table 5. MOSGA with 100 iterations to solve once.

Number of Times	Optimal Sequence	Fitness Function Value	Number of Direction Changes	Number of Tool Changes	Sequence Constraints	Geometric Constraints	Global Optimal
1	1,2,4,8,7,6,3,9,5	0.2155	3	2	Yes	Yes	Yes
2	1,2,4,7,6,8,3,9,5	0.2155	3	2	Yes	Yes	Yes
3	1,2,4,8,6,7,3,9,5	0.2155	3	2	Yes	Yes	Yes
4	1,4,2,8,6,7,3,9,5	0.2155	3	2	Yes	Yes	Yes
5	1,4,2,8,7,6,3,9,5	0.2155	3	2	Yes	Yes	Yes
6	1,2,4,6,7,8,3,9,5	0.2155	3	2	Yes	Yes	Yes

Table 6. MOSGA with 200 iterations to solve once.

Number of Times	Optimal Sequence	Fitness Function Value	Number of Direction Changes	Number of Tool Changes	Sequence Constraints	Geometric Constraints	Global Optimal
1	1,2,4,6,7,8,3,9,5	0.2155	3	2	Yes	Yes	Yes
2	1,2,4,8,7,6,3,9,5	0.2155	3	2	Yes	Yes	Yes
3	1,4,2,7,6,8,3,9,5	0.2155	3	2	Yes	Yes	Yes
4	1,4,2,6,7,8,3,9,5	0.2155	3	2	Yes	Yes	Yes
5	1,4,2,8,7,6,3,9,5	0.2155	3	2	Yes	Yes	Yes
6	1,4,2,8,6,7,3,9,5	0.2155	3	2	Yes	Yes	Yes
7	1,2,4,7,6,8,3,9,5	0.2155	3	2	Yes	Yes	Yes
8	1,2,4,8,6,7,3,9,5	0.2155	3	2	Yes	Yes	Yes

5.2. Large-Scale Experiments

In the large-scale experiment, a complete cabin with 15 parts was used. Compared to a small-scale experiment with only 9 parts, the experiment with 15 parts presented a significantly larger search space and increased the difficulty of finding solutions. The complexity of solving a nine-part problem is the factorial of 9, requiring optimization of the best solution from a total of 362,880 assembly sequences. On the other hand, solving a problem composed of 15 parts requires exploring a vast solution space, with a complexity of the factorial of 15, resulting in 1,307,674,368,000 assembly sequences. Solving the optimal sequence from these solution spaces is a tremendous computational challenge. Under the same parameters, the results of solving the GA five times with 60 iterations and 100 iterations were compared with solving the MOSGA once.

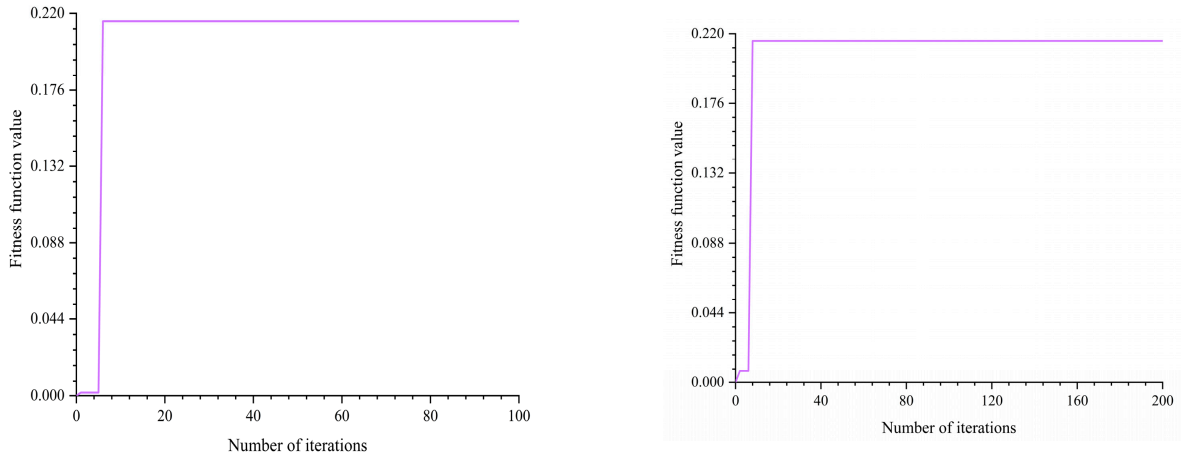


Figure 11. Convergence graphs of MOSGA for solving small-scale problems, with (**a**) 100 iterations; (**b**) 200 iterations.

5.2.1. GA Solved Five Times

From Figure 12a, it can be observed that, at 60 iterations, the GA converged to the optimal value of 0.1157 only in the fifth run, while the other four runs got trapped in local optima, as shown in Table 7. Figure 12b indicates that at 100 iterations, among the five runs of the GA, three runs ended up in local optima, as shown in Table 8. This demonstrates the instability of the GA when dealing with complex assembly sequence planning problems. Additionally, by observing Figure 13a,b, which depict convergence graphs during GA solving, it can be noted that the convergence speed of the GA is slow and unstable, often converging to local optima multiple times.

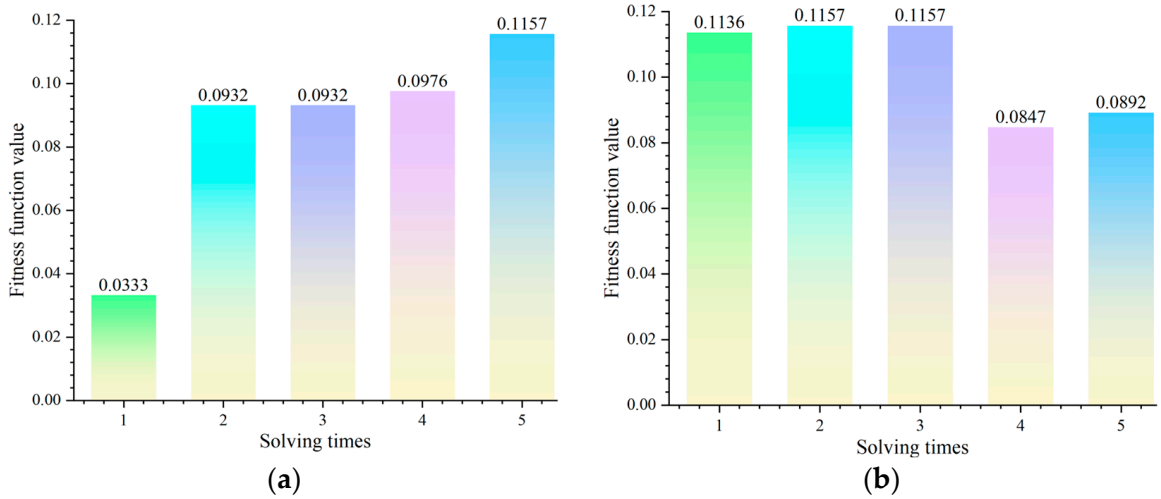


Figure 12. GA solving for the highest fitness function value 5 times, with (**a**) 60 iterations; (**b**) 100 iterations.

Table 7. GA with 60 iterations solving for 5 results.

Number of Times	Optimal Sequence	Fitness Function Value	Number of Direction Changes	Number of Tool Changes	Sequence Constraints	Geometric Constraints	Global Optimal
1	6,11,4,13,1,2,3,5,9,8,14,10,15,12,7	0.0333	11	8	Yes	Yes	No
2	1,2,6,4,8,9,3,5,10,11,7,12,13,14,15	0.0932	7	6	Yes	Yes	No
3	1,4,7,2,3,5,11,10,9,12,13,8,6,14,15	0.0932	7	6	Yes	Yes	No
4	1,2,6,11,9,3,14,15,5,4,10,7,12,13,8	0.0976	9	4	Yes	Yes	No
5	1,2,4,8,11,9,3,13,12,7,6,14,15,5,10	0.1157	6	3	Yes	Yes	Yes

Table 8. GA with 100 iterations solving for 5 results.

Number of Times	Optimal Sequence	Fitness Function Value	Number of Direction Changes	Number of Tool Changes	Sequence Constraints	Geometric Constraints	Global Optimal
1	1,2,4,14,8,3,5,15,10,11,9,12,13,7,6	0.1136	7	3	Yes	Yes	No
2	1,4,2,7,6,8,3,5,9,10,11,13,12,14,15	0.1157	5	4	Yes	Yes	Yes
3	1,2,4,9,3,14,15,5,11,10,7,6,13,12,8	0.1157	6	3	Yes	Yes	Yes
4	8,4,1,2,13,6,7,3,9,11,5,10,12,14,15	0.0847	6	5	Yes	Yes	No
5	1,4,6,2,3,13,8,11,12,15,5,10,9,7,14	0.0892	11	5	Yes	Yes	No

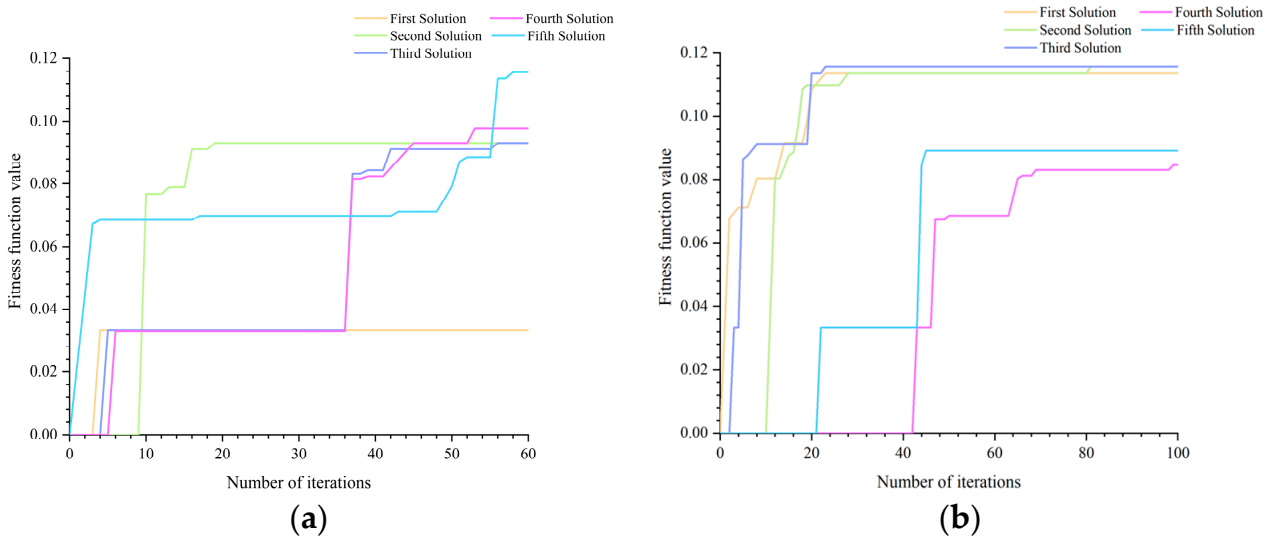


Figure 13. Convergence graphs of GA for solving large-scale problems, with (**a**) 60 iterations; (**b**) 100 iterations.

5.2.2. MOSGA Solved Once

From Figure 14a, it is evident that at 60 iterations, the MOSGA yielded four unique optimal solutions in just one run, as shown in Table 9. Figure 14b illustrates that at 100 iterations, the MOSGA produced 11 unique optimal solutions in only one run, as shown in Table 10. At the same number of iterations, the number of global optimal solutions obtained by the GA in multiple runs is significantly less than the number obtained by the MOSGA in a single run. This clearly indicates the superiority of the MOSGA over the GA. The convergence graphs in Figure 15a,b demonstrate that the MOSGA rapidly converges to the global optimum.

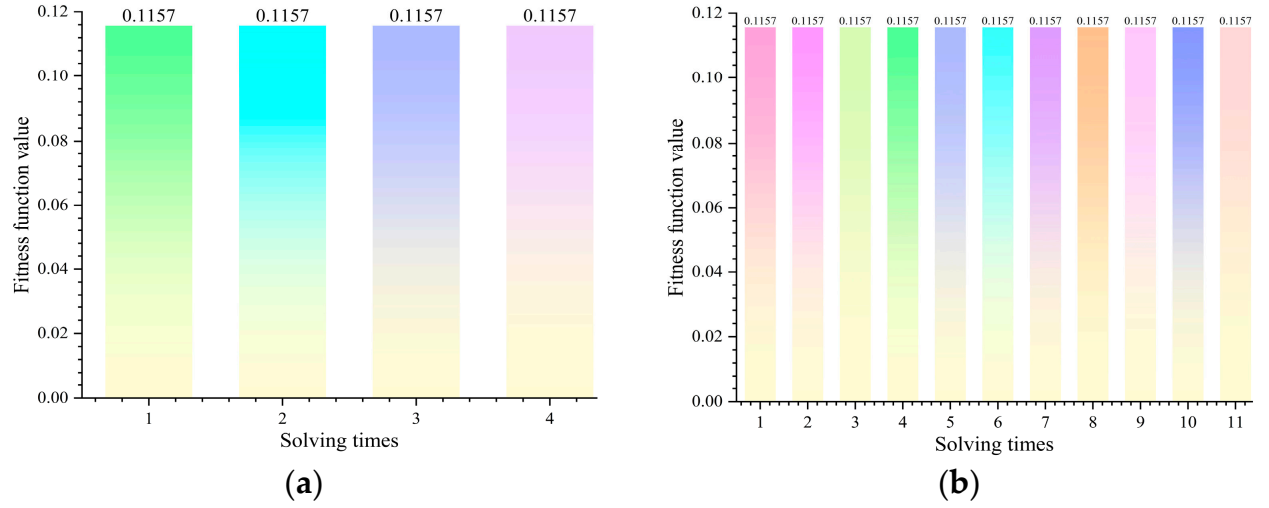


Figure 14. MOSGA solving for the highest fitness function value 1 time, with (**a**) 60 iterations; (**b**) 100 iterations.

Table 9. MOSGA with 60 iterations to solve once.

Number of Times	Optimal Sequence	Fitness Function Value	Number of Direction Changes	Number of Tool Changes	Sequence Constraints	Geometric Constraints	Global Optimal
1	1,2,4,6,7,8,3,9,14,15,5,11,10,13,12	0.1157	6	3	Yes	Yes	Yes
2	1,2,4,8,3,14,15,5,9,11,10,12,13,7,6	0.1157	6	3	Yes	Yes	Yes
3	1,2,4,8,3,5,15,14,10,9,11,13,12,7,6	0.1157	6	3	Yes	Yes	Yes
4	1,2,4,8,3,14,15,5,11,9,10,7,6,12,13	0.1157	6	3	Yes	Yes	Yes

Table 10. MOSGA with 100 iterations to solve once.

Number of Times	Optimal Sequence	Fitness Function Value	Number of Direction Changes	Number of Tool Changes	Sequence Constraints	Geometric Constraints	Global Optimal
1	1,4,2,3,9,11,5,15,14,8,10,13,12,7,6	0.1157	6	3	Yes	Yes	Yes
2	1,4,2,8,3,9,11,6,7,13,12,14,15,5,10	0.1157	6	3	Yes	Yes	Yes
3	1,4,2,8,3,9,5,15,14,6,7,11,10,13,12	0.1157	6	3	Yes	Yes	Yes
4	1,4,2,3,5,15,14,8,7,6,9,11,10,13,12	0.1157	6	3	Yes	Yes	Yes
5	1,4,2,6,7,3,5,15,14,8,10,11,9,13,12	0.1157	6	3	Yes	Yes	Yes
6	1,4,2,6,7,3,5,15,14,11,10,9,13,12,8	0.1157	6	3	Yes	Yes	Yes
7	1,4,2,6,7,3,5,15,14,11,10,9,8,13,12	0.1157	6	3	Yes	Yes	Yes
8	1,4,2,6,7,3,5,15,14,8,10,9,11,13,12	0.1157	6	3	Yes	Yes	Yes
9	1,4,2,7,6,11,9,3,5,15,14,8,10,13,12	0.1157	6	3	Yes	Yes	Yes
10	1,4,2,3,6,7,14,15,5,10,11,9,8,13,12	0.1157	6	3	Yes	Yes	Yes

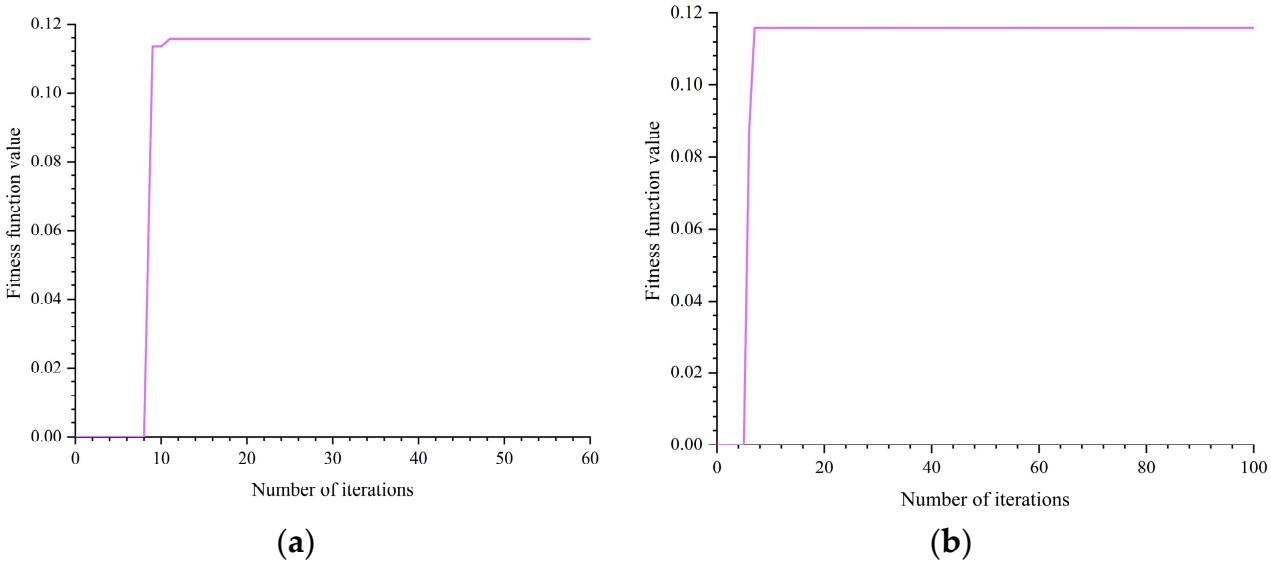


Figure 15. Convergence graphs of MOSGA for solving large-scale problems, with (**a**) 60 iterations; (**b**) 100 iterations.

6. ASFP Case Study

To verify the feasibility of ASFP method based on the MOSGA, experiments were conducted using a cabin. This shipyard completed the basic layout of the workshop in 2021 and commenced production at the beginning of 2022. Based on the cabin conditions, the optimal assembly sequence planned by the MOSGA is 1, 2, 4, 9, 3, 5, 11, 10, 14, 8, 13, 12, 6, 7, 15, as shown in the first row of Table 11. Using ASFP for its flexible assembly, when the factory assembles part 11, it is observed that there is a capacity shortage. Therefore, it is necessary to adjust the assembly sequence and conduct flexible planning. The planned result is shown in the second row of Table 7, and the optimized assembly sequence is as follows: 1, 2, 4, 9, 3, 5, 15, 14, 6, 7, 8, 10, 11, 12, 13. This sequence meets the constraints imposed by the already assembled parts 1, 4, 9, 3, 5 and represents the optimal assembly sequence for the next assembly steps. When assembling part 8, it was damaged during transportation, necessitating an adjustment to the assembly sequence. Once again, flexible planning was conducted, and the resulting sequence is shown in the third row of Table 7. The optimized assembly sequence is as follows: 1, 2, 4, 9, 3, 5, 15, 14, 6, 7, 11, 10, 13, 12, 8.

This sequence meets the constraints imposed by the already assembled parts, 1, 4, 9, 3, 5, 15, 14, 6, 7, and ensures that part 8 is assembled at a later time to address the damage issue. The completed cruise cabin is shown in Figure 16.

Table 11. Flexible planning of optimal assembly sequences.

Number of ASFP	Optimal Sequence	Fitness Function Value	Number of Direction Changes	Number of Tool Changes	Sequence Constraints	Geometric Constraints	Global Optimal
0	1,2,4,9,3,5,11,10,14,8,13,12,6,7,15	0.1157	6	3	Yes	Yes	Yes
1	1,2,4,9,3,5,15,14,6,7,8,10,11,12,13	0.1157	6	3	Yes	Yes	Yes
2	1,2,4,9,3,5,15,14,6,7,11,10,13,12,8	0.1157	5	4	Yes	Yes	Yes

Figure 16. The completed cruise cabin.

7. Discussion

This paper conducted experiments on a cabin with 15 parts in a shipyard, comparing and analyzing GAs and MOSGAs with different iteration numbers. The experimental results show that in small-scale problems, the GA exhibits instability and convergence issues at both 100 and 200 iterations, getting trapped in local optima. In Figure 17, the GA is compared with the MOSGA. In 10 runs, the GA achieved the global optimum six times in 100 iterations, and seven times in 200 iterations. In contrast, the MOSGA, in just one run, obtained six global optima at 100 iterations and eight global optima at 200 iterations.

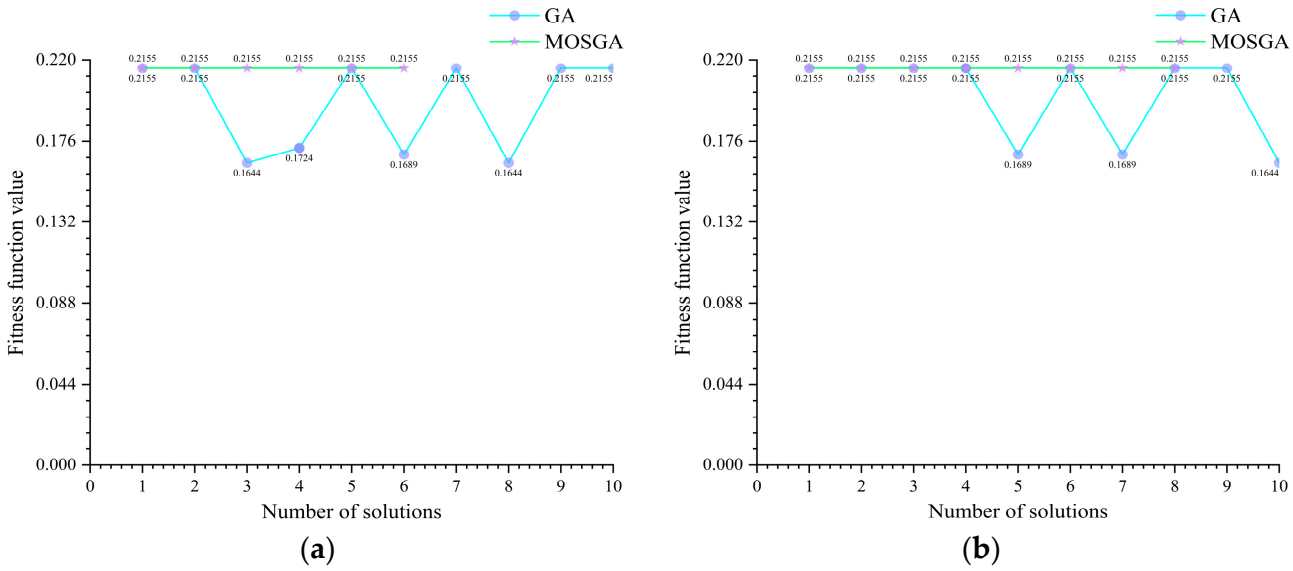


Figure 17. Comparison between MOSGA and GA, with (**a**) 100 iterations; (**b**) 200 iterations.

In large-scale problems, the advantages of the MOSGA are further highlighted. The experimental results indicate that the GA exhibits instability and convergence issues at 60 and 100 iterations, falling into local optima. In Figure 18, compare the GA with the MOSGA. Among the five runs, the GA at 60 iterations achieves a global optimum once, and at 100 iterations, it obtains two global optima, with the rest falling into local optima. In contrast, the MOSGA, with only one run, attains 4 global optima at 60 iterations and 11 global optima at 100 iterations. The proposed MOSGA overcomes the limitations of the GA in solving complex problems, further enhancing search efficiency. Our experimental results demonstrate that the MOSGA outperforms the GA significantly, whether in terms of convergence speed or the quantity of optimal solutions obtained.

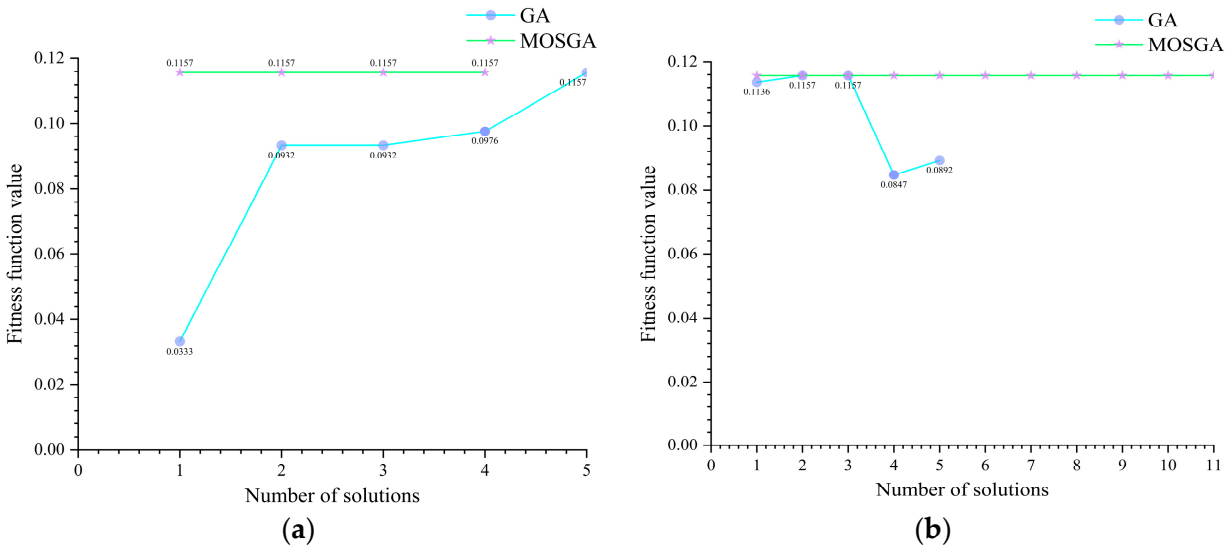


Figure 18. Comparison between MOSGA and GA, with (**a**) 60 iterations; (**b**) 100 iterations.

Observing the convergence plot reveals that the GA quickly falls into a local optimum, and increasing the number of iterations fails to bring about significant improvements. This phenomenon arises due to the complexity of the problem, leading to a larger search space for solutions. When the optimization iteration of the GA reaches a certain optimal value, the fitness value is already high. The individuals with better fitness are less distributed in the search space and it is difficult to carry out a further search. Additionally, during the population evolution process, the GA tends to converge, resulting in a reduction in population diversity. This decrease in diversity makes it difficult for the GA to undergo mutations and explore beyond the current local optimum. Therefore, further enhancements are needed to address the limitations of GAs.

Table 11 clearly demonstrates that the ASFP method proposed in this paper effectively addresses the practical re-optimization issues encountered during the assembly process. This approach is utilized in situations where assembly sequences need to be readjusted due to unforeseen constraints encountered during the assembly process, thereby avoiding the waste of manufacturing resources and improving assembly efficiency. Initially, the MOSGA is used to plan the global optimum solutions for the products requiring assembly. Throughout each step of the assembly process, flexible planning is applied based on the actual conditions in the factory. If a sudden problem arises, the assembly sequence must be readjusted. In cases of installation issues with any part, the assembly sequence can be re-planned, allowing for the development and optimization of multiple unique solutions to select the optimal sequence that aligns most closely with the actual assembly situation. By doing so, the factory continues its assembly operations with reduced waiting and total assembly times, consequently lowering manufacturing costs.

During the planning process using ASFP, if there is no new optimal assembly sequence that satisfies the constraints of the already installed parts, the planning is considered

unsuccessful, and the assembly proceeds according to the original assembly sequence. The experiments in this study addressed situations where the assembly of factory parts encountered issues twice, and flexible planning was employed to adjust the assembly sequence. If more unexpected situations are encountered, in the worst case, one or more instance of flexible planning can be carried out for each part installed until the optimal assembly sequence in line with the actual production situation is planned. The experimental results presented in this study confirm the practical feasibility of ASFP, highlighting its significant guiding implications for assembly sequence planning problems.

8. Conclusions

This paper introduces a novel ASP method suitable for the construction of modular cabins in large cruise ships. Utilizing four constraint conditions, including geometric constraints, sequence constraints, assembly changeover times, and tool changeover times, the method conducts multi-objective optimization. It generates multiple unique optimal assembly sequences that satisfy specified constraints and improves search efficiency. Simultaneously, this paper proposes the ASFP method to tackle practical re-optimization issues in factory assembly processes, such as material delivery delays, equipment breakdowns, and transportation damage. When encountering these situations, using ASFP for flexible planning can lead to the creation of a new assembly plan for workshop production.

The paper provides a detailed description of the algorithm's steps and core processes, comparing it with the GA. Illustrated by examples, the innovation of the proposed optimization algorithm is demonstrated. The key contributions of this paper are as follows:

- Overcoming Local Optima in GA: This paper resolves the issue of the GA converging to local optima when dealing with complex assembly problems. It introduces the MOSGA method, an improved approach that determines global optimal assembly sequences.
- Diverse Optimal Solutions: Addressing the problem of multiple search algorithms converging to a single optimal solution, the MOSGA not only identifies global optimal solutions but also produces multiple unique optimal solutions in a single solving process.
- ASFP for Flexible Planning: Regarding the ASFP method, this approach, designed to solve practical re-optimization issues in assembly processes, becomes crucial when unforeseen circumstances hinder adherence to the original assembly sequence, necessitating the re-planning of the optimal assembly sequence. The application of ASFP proves effective in reducing assembly costs and significantly enhancing assembly efficiency.
- Validation through Experiments: Through experiments with different parameters using a cabin example, this paper compares the GA and MOSGA, proving the advanced and innovative nature of the MOSGA. Additionally, flexible planning based on the cabin example verifies the feasibility of ASFP in solving assembly flexibility planning problems, achieving the expected results.

Author Contributions: Algorithm and experimental design X.W.; writing—original draft, X.W.; writing—review and editing, K.L. and Z.K.; visualization, W.Q. All authors have read and agreed to the published version of the manuscript.

Funding: Project supported by the National Natural Science Foundation of China (Grant No. 52171311; 52271279).

Data Availability Statement: The datasets generated or analyzed during this study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Li, S.; Tang, D.; Xue, D.; Wang, Q.; Zhu, H. Assembly sequence planning based on structure cells in open design. *Adv. Eng. Informatics* **2022**, *53*, 101685. [CrossRef]
2. Xiao, Y.; Zhou, J.; Xing, S.; Zhu, X. Research on Assembly Sequence Optimization Classification Method of Remanufacturing Parts Based on Different Precision Levels. *Processes* **2023**, *11*, 383. [CrossRef]

3. Shi, X.; Tian, X.; Gu, J.; Wang, G.; Zhao, D.; Ma, L. A hybrid approach of case- and rule-based reasoning to assembly sequence planning. *Int. J. Adv. Manuf. Technol.* **2023**, *127*, 221–236. [CrossRef]
4. Zheng, Y.; Chen, L.; Wu, D.; Jiang, P.; Bao, J. Assembly sequence planning method for optimum assembly accuracy of complex products based on modified teaching–learning based optimization algorithm. *Int. J. Adv. Manuf. Technol.* **2023**, *126*, 1681–1699. [CrossRef]
5. de Giorgio, A.; Maffei, A.; Onori, M.; Wang, L. Towards online reinforced learning of assembly sequence planning with interactive guidance systems for industry 4.0 adaptive manufacturing. *J. Manuf. Syst.* **2021**, *60*, 22–34. [CrossRef]
6. Wu, W.; Huang, Z.; Zeng, J.; Fan, K. A decision-making method for assembly sequence planning with dynamic resources. *Int. J. Prod. Res.* **2021**, *60*, 4797–4816. [CrossRef]
7. Wan, W.; Harada, K.; Nagata, K. Assembly sequence planning for motion planning. *Assem. Autom.* **2017**, *38*, 195–206. [CrossRef]
8. Ma, H.; Peng, Q.; Zhang, J.; Gu, P. Assembly sequence planning for open-architecture products. *Int. J. Adv. Manuf. Technol.* **2017**, *94*, 1551–1564. [CrossRef]
9. Watson, J.; Hermans, T. Assembly Planning by Subassembly Decomposition Using Blocking Reduction. *IEEE Robot. Autom. Lett.* **2019**, *4*, 4054–4061. [CrossRef]
10. De Winter, J.; Beckers, J.; Van de Perre, G.; El Makrini, I.; Vanderborght, B. Single assembly sequence to flexible assembly plan by Autonomous Constraint Generation. *Robot. Comput. Manuf.* **2023**, *79*, 102417. [CrossRef]
11. Wang, Y.; Wang, J.; Feng, J.; Liu, J.; Liu, X. Integrated task sequence planning and assignment for human–robot collaborative assembly station. *Flex. Serv. Manuf. J.* **2022**, *35*, 979–1006. [CrossRef]
12. Tseng, H.-E.; Chang, C.-C.; Lee, S.-C.; Huang, Y.-M. Hybrid bidirectional ant colony optimization (hybrid BACO): An algorithm for disassembly sequence planning. *Eng. Appl. Artif. Intell.* **2019**, *83*, 45–56. [CrossRef]
13. Mishra, A.; Deb, S. Assembly sequence optimization using a flower pollination algorithm-based approach. *J. Intell. Manuf.* **2016**, *30*, 461–482. [CrossRef]
14. Wang, Z.-Y.; Lu, C. An integrated job shop scheduling and assembly sequence planning approach for discrete manufacturing. *J. Manuf. Syst.* **2021**, *61*, 27–44. [CrossRef]
15. Zhu, X.; Xu, Z.; Wang, J.; Yang, X.; Fan, L. Graph-based assembly sequence planning algorithm with feedback weights. *Int. J. Adv. Manuf. Technol.* **2023**, *125*, 3607–3617. [CrossRef]
16. Qian, J.; Zhang, Z.; Shi, L.; Song, D. An assembly timing planning method based on knowledge and mixed integer linear programming. *J. Intell. Manuf.* **2021**, *34*, 429–453. [CrossRef]
17. Han, Z.; Wang, Y.; Tian, D. Ant colony optimization for assembly sequence planning based on parameters optimization. *Front. Mech. Eng.* **2021**, *16*, 393–409. [CrossRef]
18. Gao, B.; Zhang, S.; Sun, H.; Ma, C. Assembly sequence planning based on adaptive gravitational search algorithm. *Int. J. Adv. Manuf. Technol.* **2021**, *115*, 3689–3700. [CrossRef]
19. Cheng, H.; Li, Y.; Zhang, K.-F. Efficient method of assembly sequence planning based on GAAA and optimizing by assembly path feedback for complex product. *Int. J. Adv. Manuf. Technol.* **2009**, *42*, 1187–1204.
20. Ab Rashid, M.F.F.; Hutabarat, W.; Tiwari, A. Multi-objective discrete particle swarm optimisation algorithm for integrated assembly sequence planning and assembly line balancing. *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* **2016**, *232*, 1444–1459. [CrossRef]
21. Kang, M.; Seo, J.; Chung, H. Ship block assembly sequence planning considering productivity and welding deformation. *Int. J. Nav. Arch. Ocean Eng.* **2018**, *10*, 450–457. [CrossRef]
22. Su, Y.; Mao, H.; Tang, X. Algorithms for solving assembly sequence planning problems. *Neural Comput. Appl.* **2020**, *33*, 525–534. [CrossRef]
23. Che, Z.; Chiang, T.-A.; Lin, T.-T. A multi-objective genetic algorithm for assembly planning and supplier selection with capacity constraints. *Appl. Soft Comput.* **2020**, *101*, 107030. [CrossRef]
24. Liu, Y.; Li, S.; Wang, J. Assembly auxiliary system for narrow cabins of spacecraft. *Chin. J. Mech. Eng.* **2015**, *28*, 1080–1088. [CrossRef]
25. Murali, G.B.; Deepak, B.; Raju, M.; Biswal, B. Optimal robotic assembly sequence planning using stability graph through stable assembly subset identification. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **2019**, *233*, 5410–5430. [CrossRef]
26. Ying, K.-C.; Pourhejazy, P.; Cheng, C.-Y.; Wang, C.-H. Cyber-physical assembly system-based optimization for robotic assembly sequence planning. *J. Manuf. Syst.* **2021**, *58*, 452–466. [CrossRef]
27. Masehian, E.; Ghandi, S. ASPPR: A New Assembly Sequence and Path Planner/Replanner for Monotone and Nonmonotone Assembly Planning. *Comput. Aided Des.* **2020**, *123*, 102828. [CrossRef]
28. Zhou, B.; Bao, J.; Chen, Z.; Liu, Y. KGAssembly: Knowledge graph-driven assembly process generation and evaluation for complex components. *Int. J. Comput. Integr. Manuf.* **2021**, *35*, 1151–1171. [CrossRef]
29. Wang, Z.; Kennel-Maushart, F.; Huang, Y.; Thomaszewski, B.; Coros, S. A Temporal Coherent Topology Optimization Approach for Assembly Planning of Bespoke Frame Structures. *ACM Trans. Graph.* **2023**, *42*, 1–13. [CrossRef]
30. Gulivindala, A.K.; Bahubalendruni, M.R.; Varupala, S.V.P.; K, S. A heuristic method with a novel stability concept to perform parallel assembly sequence planning by subassembly detection. *Assem. Autom.* **2020**, *40*, 779–787. [CrossRef]
31. Rodriguez, I.; Nottensteiner, K.; Leidner, D.; Kassecker, M.; Stulp, F.; Albu-Schaffer, A. Iteratively Refined Feasibility Checks in Robotic Assembly Sequence Planning. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1416–1423. [CrossRef]

32. Yang, X.; Xie, H.; Chen, L.; Gao, M.; Li, C.; Li, J. Assembly sequence planning and evaluating for deep oil and gas corer based on graph theory. *Geoenergy Sci. Eng.* **2023**, *231*, 212386. [CrossRef]
33. Gunji, B.M.; Deepak, B.B.V.L.; Biswal, B.B. Effect of Considering Secondary Parts as Primary Parts for Robotic Assembly Using Stability Graph. *Arab. J. Sci. Eng.* **2019**, *45*, 743–764. [CrossRef]
34. Tariki, K.; Kiyokawa, T.; Nagatani, T.; Takamatsu, J.; Ogasawara, T. Generating complex assembly sequences from 3D CAD models considering insertion relations. *Adv. Robot.* **2020**, *35*, 337–348. [CrossRef]
35. Gao, Y.; Meng, J.; Shu, J.; Liu, Y. BIM-based task and motion planning prototype for robotic assembly of COVID-19 hospitalisation light weight structures. *Autom. Constr.* **2022**, *140*, 104370. [CrossRef]
36. Xia, L.; Lu, J.; Lu, Y.; Gao, W.; Fan, Y.; Xu, Y.; Zhang, H. Semantic knowledge-driven A-GASeq: A dynamic graph learning approach for assembly sequence optimization. *Comput. Ind.* **2024**, *154*, 104040. [CrossRef]
37. Rehal, A.; Sen, D. An Efficient Disassembly Sequencing Scheme Using the Shell Structure. *Comput. Des.* **2023**, *154*, 103423. [CrossRef]
38. Ab Rashid, M.F.F.; Tiwari, A.; Hutabarat, W. Integrated optimization of mixed-model assembly sequence planning and line balancing using Multi-objective Discrete Particle Swarm Optimization. *Artif. Intell. Eng. Des. Anal. Manuf.* **2019**, *33*, 332–345. [CrossRef]
39. Zhang, J.; Wang, P.; Zuo, M.; Li, Y.; Xu, Z. Automatic assembly simulation of product in virtual environment based on interaction feature pair. *J. Intell. Manuf.* **2015**, *29*, 1235–1256. [CrossRef]
40. Ji, W.; Yin, S.; Wang, L. A Virtual Training Based Programming-Free Automatic Assembly Approach for Future Industry. *IEEE Access* **2018**, *6*, 43865–43873. [CrossRef]
41. Tao, S.; Wang, D.-Y.; Zhang, S.-W. A feature and optimized RRT algorithm-based assembly path planning method of complex products. *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* **2023**. [CrossRef]
42. Zhang, N.; Liu, Z.; Qiu, C.; Hu, W.; Tan, J. Optimizing assembly sequence planning using precedence graph-based assembly subsets prediction method. *Assem. Autom.* **2019**, *40*, 361–375. [CrossRef]
43. You, H.; Ye, Y.; Zhou, T.; Zhu, Q.; Du, J. Robot-Enabled Construction Assembly with Automated Sequence Planning Based on ChatGPT: RoboGPT. *Buildings* **2023**, *13*, 1772. [CrossRef]
44. Rodriguez, I.; Nottensteiner, K.; Leidner, D.; Durner, M.; Stulp, F.; Albu-Schaffer, A. Pattern Recognition for Knowledge Transfer in Robotic Assembly Sequence Planning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3666–3673. [CrossRef]
45. Sadeghi Tabar, R.; Wärmefjord, K.; Söderberg, R.; Lindkvist, L. Critical joint identification for efficient sequencing. *J. Intell. Manuf.* **2021**, *32*, 769–780. [CrossRef]
46. Li-li, L.; Kun, C.; Jian-min, G.; Jun-kong, L.; Zhi-yong, G.; Hong-wei, D. Research on optimizing-assembly and optimizing-adjustment technologies of aero-engine fan rotor blades. *Adv. Eng. Inform.* **2022**, *51*, 101506. [CrossRef]
47. Wang, Z.; Gan, Y.; Dai, X. Assembly-Oriented Task Sequence Planning for a Dual-Arm Robot. *IEEE Robot. Autom. Lett.* **2022**, *7*, 8455–8462. [CrossRef]
48. Masehian, E.; Ghandi, S. Assembly sequence and path planning for monotone and nonmonotone assemblies with rigid and flexible parts. *Robot. Comput. Manuf.* **2021**, *72*, 102180. [CrossRef]
49. Shahi, V.J.; Masoumi, A.; Franciosa, P.; Ceglarek, D. A quality-driven assembly sequence planning and line configuration selection for non-ideal compliant structures assemblies. *Int. J. Adv. Manuf. Technol.* **2019**, *106*, 15–30. [CrossRef]
50. Abidi, M.H.; Al-Ahmari, A.M.; Ahmad, A.; Darmoul, S.; Ameen, W. Semi-Immersive Virtual Turbine Engine Simulation System. *Int. J. Turbo Jet-Engines* **2017**, *35*, 149–160. [CrossRef]
51. Abdullah, A.; Ab Rashid, M.F.F.; Ponnambalam, S.; Ghazalli, Z. Energy efficient modeling and optimization for assembly sequence planning using moth flame optimization. *Assem. Autom.* **2019**, *39*, 356–368. [CrossRef]
52. Liu, C.; Zhang, F.; Zhang, H.; Shi, Z.; Zhu, H. Optimization of assembly sequence of building components based on simulated annealing genetic algorithm. *Alex. Eng. J.* **2023**, *62*, 257–268. [CrossRef]