

MULTI-OBJECTIVE SCHEDULING OPTIMIZATION OF MARINE ELEVATOR ASSEMBLY USING AN IMPROVED GENETIC ALGORITHM

Ruimei CI¹, Yuhang ZHAO², Yifei TONG^{3,*}, Xinye WU⁴

Marine elevator assembly in naval shipbuilding is a fixed-position process with strict precedence constraints and strong reliance on skilled labor. Conventional scheduling methods, based on manual experience, often lead to prolonged cycles, unbalanced workloads, and inefficient resource use. This study develops a comprehensive scheduling model that incorporates task precedence, resource allocation, worker assignment, and cost considerations. To solve the model, an improved dual-population genetic algorithm (DPGA) with integer encoding and precedence-preserving operators is proposed. Linear fitness scaling and adaptive weighting are applied to integrate multiple objectives, including cycle time, workload balance, skill-task matching, and cost. A case study on a WQ-type marine elevator validates the approach. Compared with manual scheduling and the standard GA, the proposed method shortens the cycle from 48 to 45 days, improves workload balance, raises skill-task matching above 76%, and reduces relative cost variance. The results confirm that the method enhances efficiency, fairness, and cost reliability.

Keywords: Marine Elevator; Multi-objective Decision Making; Production Scheduling Optimization; Genetic Algorithm

1. Introduction

In modern naval operations, the assembly of large-scale shipborne equipment is a decisive factor affecting combat readiness, logistical efficiency, and overall mission capability. Marine elevators, as a critical subsystem on warships, are primarily used for the vertical transportation of ammunition, supplies, and personnel across multiple decks. Their operational reliability directly impacts the efficiency of weapons handling and the responsiveness of combat systems, making them indispensable for sustaining naval missions.

* Corresponding author

¹ Yangzhou Polytechnic University, Jiangsu, 225009, China

² School of Mechanical Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

³ Prof., School of Mechanical Engineering, Nanjing University of Science and Technology, Nanjing 210094, China, e-mail: tyf1776@mail.njust.edu.cn

⁴ School of Mechanical Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

The assembly of military marine elevators presents unique challenges. Due to their large structural dimensions and strict safety requirements, the assembly process is conducted in a fixed-position mode rather than on a flow line. This mode demands intensive manual operations, complex task dependencies, and the coordination of limited resources under harsh naval construction environments. Unlike mass-production assembly, where standardized takt times can guide scheduling, naval elevator assembly requires flexible task allocation and precise resource scheduling to ensure both quality and timeliness. Efficient scheduling is therefore not only an engineering concern but also a strategic imperative to guarantee the rapid deployment and operational readiness of naval forces.

Research on assembly scheduling has evolved significantly over the past decades, with approaches ranging from classical optimization methods to advanced intelligent algorithms. Traditional methods primarily rely on mathematical programming techniques such as mixed-integer linear programming (MILP) [1], branch-and-bound [2], and constraint programming [3]. While these methods can produce exact solutions for small-scale problems, they often become computationally infeasible for large, complex systems involving multiple objectives and uncertainties. In naval equipment assembly, where tasks are highly interdependent and resource constraints are stringent, exact optimization approaches are rarely applicable in practice.

To overcome these limitations, heuristic and metaheuristic algorithms have been widely investigated. Techniques such as simulated annealing (SA) [4], tabu search (TS) [5], particle swarm optimization (PSO) [6], and genetic algorithms (GA) [7] have shown strong capabilities in addressing large-scale combinatorial scheduling problems. Among them, GA has been particularly popular for assembly scheduling, workforce allocation, and resource optimization, due to its global search ability and flexibility in handling nonlinear and multi-objective problems. Recent advances include multi-objective evolutionary algorithms (e.g., NSGA-II) and hybrid GA approaches that integrate problem-specific heuristics [8]. These methods have achieved promising results in flow-line scheduling by reducing cycle times, improving resource utilization, and balancing workloads.

Nevertheless, most of the existing studies are designed for flow-line production settings with repetitive product structures and well-defined takt times [9]. Fixed-position assembly, which is prevalent in naval shipbuilding, presents fundamentally different challenges. Task durations are highly variable, parallel workstations must be coordinated dynamically, and manual labor plays a dominant role in determining quality. Despite its importance, relatively few studies have specifically addressed scheduling in fixed-position assembly systems, particularly in the context of integrating precedence constraints, resource availability, workload balance, and skill-task matching into a unified optimization framework.

To bridge this research gap, this paper develops a novel approach to the scheduling of military marine elevator assembly. A comprehensive scheduling model is established that explicitly incorporates precedence relations, resource constraints, worker load balancing, and skill-task matching. Unlike conventional flow-line models, this formulation captures the unique characteristics of fixed-position naval assembly, where flexibility and coordination are critical. On this basis, an improved dual-population genetic algorithm (DPGA) is proposed to solve the multi-objective optimization problem. By combining exploration and exploitation sub-populations, employing topology-preserving operators, and adopting adaptive fitness scaling, the algorithm ensures feasible solutions and robust convergence. The proposed method is validated through a real-world case study of WQ-type naval elevator assembly, demonstrating its capability to reduce cycle time, improve workload balance, and lower costs when compared with manual scheduling and standard GA approaches.

2. Marine Elevator Assembly Scheduling Problem Description

The assembly of marine elevators is conducted in a fixed-position mode due to their large structural dimensions and strict naval safety requirements [10]. Unlike flow-line manufacturing, where products are transported across sequential workstations, the marine elevator remains stationary during assembly, while workers, tools, and equipment are mobilized around it [11]. This fixed-position approach reflects standard practice in naval shipyards, where elevators are integrated directly within the vessel hull. The process requires intensive manual operations, precise alignment of structural parts, and coordination among multiple teams working in parallel.

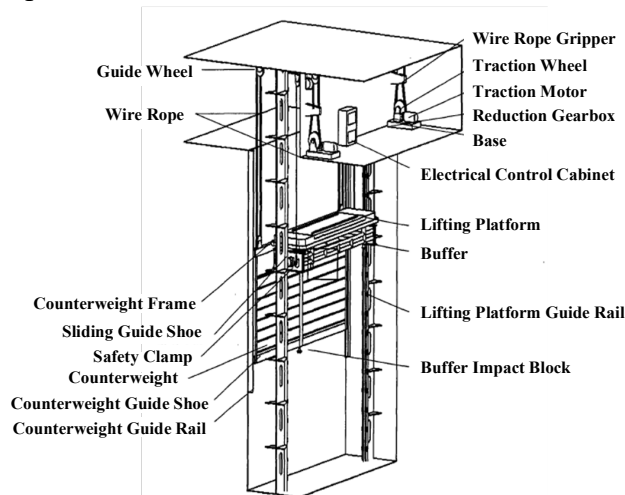


Fig. 1. Main Structure of Marine Elevator

The WQ-type marine elevator, used as the case study in this paper, consists of several functional subsystems, including the shaft structure, safety doors, electrical and control systems, hydraulic units, and the lifting platform. The complete assembly cycle spans approximately 48 days, covering tasks from initial inspection to final system testing. Individual tasks vary in duration from one to fifteen days, and while some can be executed in parallel, many are subject to strict precedence constraints.

Current assembly practice in naval shipyards reveals several inefficiencies. Assembly sequencing is often adjusted in real time by supervisors based on experience, rather than determined by a systematic optimization framework. As a result, parallelizable operations are not fully exploited, and the overall cycle time is extended. Resource allocation also lacks precision: although up to 15 workers are typically available, their workload distribution is uneven, and skill differences are rarely considered in task assignment. Additionally, critical tools such as cranes, hoists, and measuring devices are limited in number, creating resource bottlenecks.

These challenges underscore the necessity of a formal scheduling model that integrates precedence relations, resource constraints, and worker assignment. By improving scheduling efficiency, it becomes possible to shorten the total assembly cycle, balance workloads across workers, and reduce operational costs. More importantly, for military shipbuilding, optimized marine elevator assembly scheduling is a strategic requirement, as timely delivery of warships directly affects naval combat readiness and mission capability.

3. Marine Elevator Assembly Scheduling Model Formulation

3.1 Precedence Relationship Model

The assembly of a marine elevator can be represented as a set of interdependent tasks, each with a specific processing time. These tasks are subject to precedence relations, meaning that certain tasks cannot start until their predecessors are completed.

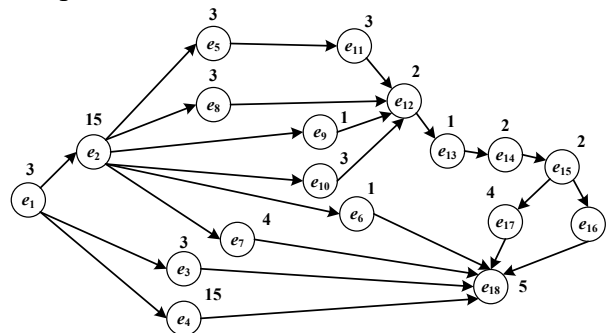


Fig. 2. Operation Priority Relationship Diagram

Formally, let the set of tasks be denoted as $E = \{e_1, e_2, \dots, e_n\}$, where each task e_i has a processing time t_i . The precedence constraints can be expressed using a directed acyclic graph (as shown in figure 2), where nodes represent tasks and edges represent precedence relations. The precedence matrix $A = [a_{ij}]_{n \times n}$ is defined as follows:

$$\begin{cases} a_{ij} = 1, e_i \text{ is the predecessor process of } e_j \\ a_{ij} = 0, \text{ otherwise} \end{cases} \quad (1)$$

This representation ensures that the scheduling process respects the logical order of assembly.

3.2 Resource Model

Marine elevator assembly requires both equipment resources and human resources. Let $R = \{r_1, r_2, \dots, r_m\}$ denote the set of resource types, and $N = \{n_1, n_2, \dots, n_m\}$ the corresponding maximum available quantities. The resource requirements of each task can be expressed by a demand matrix $C = [c_{ij}]$, where c_{ij} represents the demand of task e_j for resource r_i . The resource constraints can be formulated as:

$$\sum_{j=1}^n c_{ij} \cdot x_{ij}(t) \leq n_i, \forall r_i \in R \quad (2)$$

where $x_{ij}(t)$ is a binary variable indicating whether task e_j occupies resource r_i at time t . This ensures that no resource is over-allocated at any time.

3.3 Assembly Scheduling Model

The assembly scheduling model integrates precedence and resource constraints into a unified optimization framework. Let the start time of task i be denoted as S_i , and its finish time as F_i . The scheduling constraints can be defined as follows.

Precedence constraint:

$$S_j \geq F_i, \forall (i, j) \in \mathcal{P} \quad (3)$$

where \mathcal{P} represents the set of precedence relations among tasks. This ensures that successor tasks cannot start until their predecessors are completed.

Resource capacity constraint:

Let $x_{it} \in \{0, 1\}$ indicate whether task i is processed at discrete time slot t . With resource demand d_{ir} and capacity R_r for resource type r , the resource capacity constraint is expressed as

$$\sum_{i=1}^n d_{ir} x_{it} \leq R_r, \forall r \in \mathcal{R}, \forall t \in \mathcal{T}. \quad (4)$$

Here, $x_{it} = 1$ if $t \in [S_i, F_i)$ in the decoded schedule, and 0 otherwise. This formulation ensures that no resource is over-allocated at any time.

Chance-constrained form: Considering the variability of real shipyard operations, task durations t_i are modeled as stochastic variables following $t_i \sim$

$\mathcal{N}(\mu_i, \sigma_i^2)$, where μ_i represents the nominal duration and σ_i reflects environmental or operational fluctuation.

Accordingly, the resource capacity constraint in Eq. (4) is reformulated as a probabilistic constraint with a confidence level α :

$$\mathbb{P}(\sum_{i=1}^n d_{ir} x_{it} \leq R_r) \geq \alpha, \forall r \in \mathcal{R}, \forall t \in \mathcal{T}. \quad (4')$$

During optimization, Monte Carlo sampling is integrated into the decoding process to estimate the expected completion time and verify feasibility under uncertainty, ensuring the robustness of the obtained schedule.

Parallel execution constraint: At most L tasks can be executed in parallel due to limited workstations and spatial restrictions:

$$\sum_{i=1}^n x_{it} \leq L, \forall t \in \mathcal{T}. \quad (5)$$

This constraint prevents spatial congestion during fixed-position assembly. The objective of the model is to determine a feasible start–finish schedule $\{S_i, F_i\}$ for all tasks that satisfies the above constraints while minimizing total cycle time, balancing workloads, and controlling costs.

3.4 Worker Assignment Model

Given the strong reliance on manual labor in marine elevator assembly, worker assignment plays a decisive role. Let $W = \{w_1, w_2, \dots, w_k\}$ be the set of available workers. Each worker has an accumulated workload wt_i , and the load balance across workers can be measured by the variance of workloads:

$$SI = \sqrt{\frac{1}{k} \sum_{i=1}^k (wt_i - \mu)^2} \quad (6)$$

where μ is the average workload. A smaller Smoothness Index (SI) indicates better load balance. Furthermore, a capability index CI is introduced to evaluate the match between worker skills and assigned tasks, ensuring that tasks requiring specialized skills are assigned to appropriately trained workers.

4. Solution for Marine Elevator Assembly Scheduling Based on Genetic Algorithm

4.1 Genetic Algorithm and Proposed Improvements

GA is a population-based stochastic optimization method inspired by the principles of natural selection and genetics. To address the complexity of marine elevator assembly scheduling, the standard genetic algorithm is enhanced with several tailored strategies. To ensure robustness against uncertainty in task durations and resource fluctuations, the evaluation mechanism integrates stochastic sampling during fitness computation, allowing each schedule to be assessed based on its expected performance. First, a dual-population structure is adopted, where one sub-population emphasizes global exploration to maintain diversity, while the

other focuses on local exploitation to accelerate convergence. Periodic migration between the two sub-populations allows information exchange and prevents premature convergence.

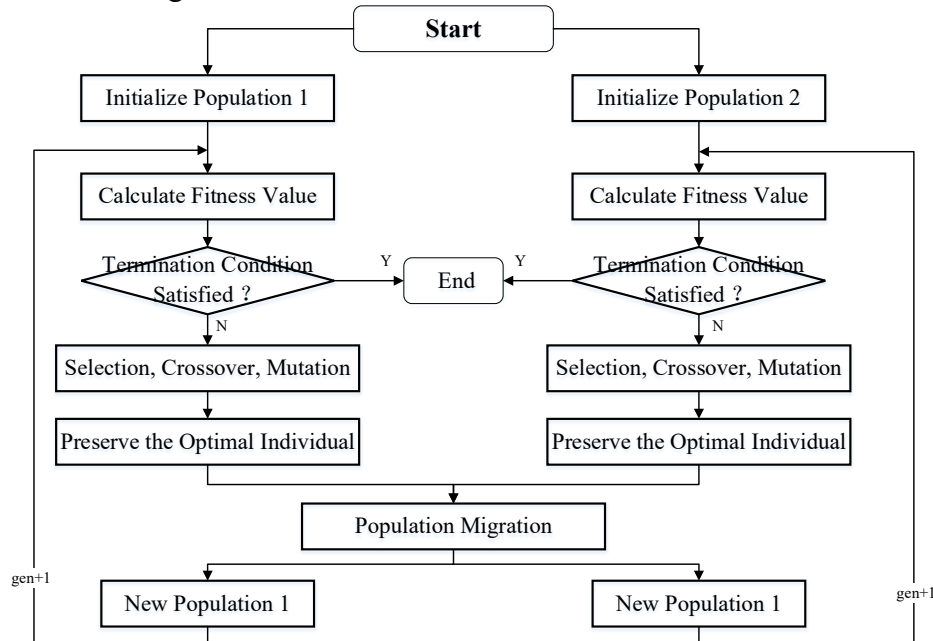


Fig.3. Flow of Dual-Population Genetic Algorithm

In addition, a precedence-based encoding scheme is employed to guarantee that crossover and mutation operations always produce feasible schedules without violating task precedence constraints. This topology-preserving mechanism ensures the validity of generated solutions throughout the evolutionary process. Furthermore, since the scheduling problem involves multiple objectives such as minimizing cycle time, balancing workloads, improving skill-task matching, and controlling costs, adaptive weights are introduced to dynamically adjust the relative importance of each objective during the search. Fitness scaling is also applied to normalize objective values and avoid domination by any single criterion. These improvements collectively enhance the algorithm's ability to generate high-quality and robust scheduling solutions.

4.2 Algorithm Framework

(1) Encoding: To identify optimal task sequences, integer encoding is adopted for its simplicity and effectiveness [12]. As illustrated in Figure 4, chromosomes are randomly generated based on the task precedence relationship model. The initial population is created by assigning processes from left to right, strictly adhering to the precedence constraints.

1	2	5	8	9	10	3	4	6	7	11	12	13	14	15	17	16	18
---	---	---	---	---	----	---	---	---	---	----	----	----	----	----	----	----	----

Fig. 4. Chromosome Encoding

(2) Decoding: Given the number of workstations l operating simultaneously on the marine elevator, the assembly process set $E = \{e_i | e_1, e_2, e_3, \dots, e_n\}$, and corresponding operation times $T = \{t_i | t_1, t_2, t_3, \dots, t_n\}$, the detailed decoding procedure is as follows: the decoding procedure begins by assigning processes sequentially according to the gene sequence in the initialized population. For each process e_i , the algorithm checks whether both the task itself and its predecessors have been scheduled, and then determines the latest completion time among predecessors $\max t_e(efu, k)$. The workstation with the earliest completion time $\min s e_k$ is identified, and the start time t_{ib} of process e_i is set as the later of these two values. Once the start time is fixed, the operation end time t_{ie} is calculated, and resource allocation constraints are verified; if they are not satisfied, the earliest available workstation is excluded and the search is repeated until a feasible allocation is found. For uncertain task durations, multiple realizations of $\{t_i\}$ are sampled to evaluate expected completion time and feasibility under constraint (4'). The process is then assigned to the selected workstation and further allocated to the worker with the least cumulative assembly time wt_i , whose operation times are recorded. This procedure continues until all tasks are scheduled, after which the total assembly cycle time T is obtained.

(3) Fitness Functions

To evaluate the quality of scheduling solutions, four objective metrics are designed to capture the efficiency, balance, capability, and cost of marine elevator assembly.

First, assembly efficiency is measured by the ratio of effective working time to the total cycle time, reflecting the utilization of available workstations:

$$fit_1 = \eta = \frac{\sum_{k=1}^l st_k}{l \cdot CT} \quad (7)$$

where CT is the assembly cycle time, st_k is the total operation time at workstation k , and l is the number of simultaneously active workstations.

Second, worker load balance is assessed using the SI , defined as the variance in cumulative workloads among workers:

$$fit_2 = SI = \frac{\sum_{i=1}^n (wt_i - \mu)^2}{n} \quad (8)$$

where wt_i is the cumulative assembly time of worker i , and μ is the average workload. A smaller SI indicates better load balance.

Third, skill-task matching is quantified through the Capability Index (CI), which evaluates the degree to which tasks requiring specialized skills are assigned to appropriately trained workers:

$$fit_3 = CI = \frac{1}{\sum_{j=1}^m sk_m} \quad (9)$$

where w_{gi} is the specialized task group of workers i , p_{gj} is the group of process j , and sk_m is the matching index defined as the reciprocal of total worker-task assignments.

Fourth, assembly cost performance is measured by the Relative Cost Variance (RCV), incorporating inventory cost and delay cost. Inventory cost is defined as:

$$C_{inv} = \sum_i t_i \cdot s_i \cdot c_i \quad (10)$$

Where t_i is storage duration, s_i storage area, and c_i storage cost per unit area and time. Delay cost is expressed as:

$$C_{delay} = c_2 \cdot \max(0, T - d) \quad (11)$$

where T is the actual completion time, d the delivery deadline, and c_2 the unit penalty cost. Combining these, the RCV is given by:

$$fit_4 = RCV = \frac{BCWP - ACWP}{BCWP} = \frac{BCWP - (C_{inv} + C_{delay})}{BCWP} \quad (12)$$

where ACWP is the actual cost of work performed and BCWP the budgeted cost.

Since these objectives differ in scale and units, a linear scaling transformation is applied to normalize the fitness values:

$$f' = af + b \quad (13)$$

where a and b are determined by the maximum and average fitness in the population, ensuring that superior individuals retain their advantage while avoiding excessively large values. Finally, the multiple objectives are combined into a single composite fitness function using the weighted sum method:

$$F = \sum_{i=1}^4 \lambda_i f_i \quad (14)$$

where f_i is the normalized value of each objective and λ_i is its adaptive weight. By dynamically adjusting these weights during the evolutionary process, the algorithm maintains a balance among cycle time, workload distribution, skill-task alignment, and cost performance.

(4) Selection:

To increase the probability of retaining high-performance chromosomes and enhance global convergence, this study employs roulette wheel selection [13], where selection probability is proportional to fitness values. For complex problems, to prevent the loss of optimal solutions due to the stochastic nature of selection operators, an elite strategy is integrated with the roulette wheel method. In the selection process, given a population size Num_Pop and fitness value f_i of individual i , the selection probability p_i for individual i in the current population is:

$$p_i = \frac{f_i}{\sum_{i=1}^{Num_Pop} f_i} \quad (15)$$

where p_i represents the proportion of the roulette wheel area occupied by individual i . The probability of being selected is proportional to this area. This method can also be represented linearly, as shown in Figure 5. In the interval $[0, 1]$, each segment D_i corresponds to the selection probability of individual i , with the right endpoint of each segment representing the cumulative probability up to that individual, satisfying: $\sum_{k=1}^{Num_Pop} p_c(i) = 1$. During each selection operation, a random number $RAND \in [0, 1]$ is generated as the pointer. If $RAND$ falls within interval D_i , individual i is selected for the offspring population. This process repeats until the offspring population reaches size Num_Pop .

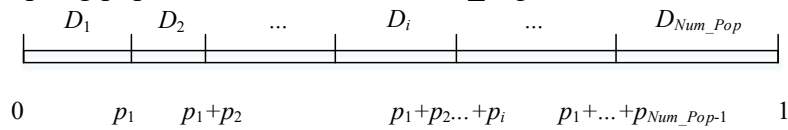


Fig. 5. Individual Probability Intervals

(5) Crossover:

For the encoding scheme of marine elevator assembly scheduling, this study employs Order Crossover, a two-point crossover method that exchanges gene sequences between two chromosomes. This approach prevents gene conflicts while ensuring that the resulting gene order remains a feasible solution satisfying precedence constraints.

(6) Mutation:

To maintain solution feasibility in marine elevator assembly scheduling, simple mutation operations like insertion or inversion cannot be used. Instead, the search process must be confined to the feasible solution space to enhance algorithm efficiency [14]. Therefore, for such combinatorial optimization problems, mutation is performed using a method similar to the initial feasible solution generation. The steps are as follows:

Step 1: Randomly select an integer from the interval $[0, n-1]$ as the mutation point. For example, suppose $Position_Mutation = 11$. The gene sequence before the mutation point remains unchanged, while the sequence after the mutation point is rearranged. (See Fig. 6.)

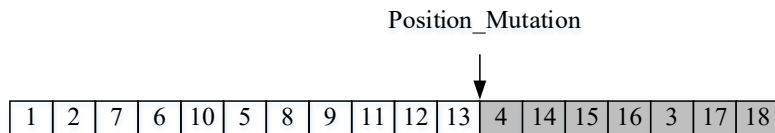


Fig. 6. Generating Chromosome Mutation Points

Step 2: Extract the partial precedence matrix from the full precedence matrix A by removing the gene segment before the mutation point, retaining only the precedence relationships for genes after the mutation point.

Step 3: From the precedence matrix definition, for a given j , if $a_{ij}=0$ for all i , process e_j has no predecessors and can be scheduled.

Step 4: Randomly select a gene from the schedulable set and append it to the previously scheduled gene segment.

Step 5: Update the precedence matrix and return to Step 3 until all genes are allocated.

The mutation process produces a new offspring chromosome, as shown in Fig. 7. To prevent local convergence, no competitive strategy is used during crossover and mutation.

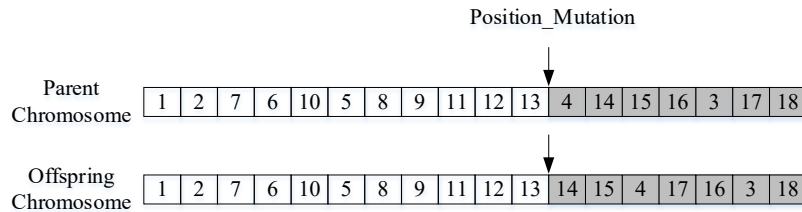


Fig. 7. Offspring Chromosomes after Mutation Operation

(7) Population Migration:

In the multi-population genetic algorithm, each population evolves independently. Information exchange is facilitated by introducing high-quality individuals from other populations, particularly during the middle and late stages of evolution. This approach enhances population diversity while accelerating the elimination of low-performing individuals, thereby preventing inbreeding. The migration process consists of two components: migration of elite individuals and random exchange of other individuals

The best individual from each population is directly copied to the other population. Then, Num_Exc individuals are selected from each population for exchange, while the remaining individuals remain unchanged. For random exchange, researchers often employ the golden section method [14]. Specifically, the top $0.618 \times Num_Pop$ chromosomes (ranked by fitness) are selected for roulette wheel selection, from which $0.618^2 \times Num_Pop$ chromosomes are randomly chosen for migration. These replace the lowest-fitness chromosomes in the other population, forming a new population.

4.3 Implementation Framework and Computational Complexity

To facilitate real-world application in naval shipyards, the proposed DPGA was implemented in MATLAB R2023a and Python 3.10 environments using modular interfaces. Each component of the algorithm—including population initialization, precedence-preserving crossover, adaptive mutation, and dual-population migration—is encapsulated as an independent function, allowing seamless integration with existing Manufacturing Execution Systems (MES) or Enterprise Resource Planning (ERP) platforms. Through standardized data exchange formats (CSV and XML), the scheduling results can be imported into MES databases for visualization, resource tracking, and production adjustment.

This modular structure simplifies practical deployment and supports interactive re-scheduling when real-time feedback is available.

From the computational perspective, the overall time complexity of the proposed DPGA is $O(NG)$, where N denotes the population size and G the number of generations. Since each individual evaluation involves deterministic decoding and limited Monte Carlo sampling under constraint (4'), the algorithm remains efficient for medium-scale assembly problems. Comparative experiments show that convergence is typically achieved within 100 generations, corresponding to an average runtime of less than 5 minutes on a workstation equipped with an Intel i7 CPU and 16 GB RAM. Therefore, the proposed scheduling approach achieves a practical balance between algorithmic sophistication and industrial applicability.

Experiments were conducted on a workstation with Intel i7 CPU (3.4 GHz) and 16 GB RAM. Each runtime represents the mean of five independent trials.

Table 1

Computational Performance and Convergence Comparison of DPGA

Algorithm	Population Size N	Generations G	Complexity Order	Avg. Convergence Gen.	Avg. Runtime (min)	Feasibility for Medium-Scale Projects
Standard GA	100	100	$O(NG)$	120	3.4	High
NSGA-II	100	100	$O(NG^2)$	95	6.8	Moderate
Hybird PSO-GA	100	100	$O(NG)$	90	4.8	Moderate
Proposed GA	100	100	$O(NG)$	80	4.7	High

As shown in Table 1, although the proposed DPGA framework includes several advanced features such as dual-population evolution and adaptive weighting, its computational and implementation requirements remain moderate. The algorithm was designed with modular operators that can be integrated into existing scheduling systems using standard programming environments such as MATLAB or Python. The total runtime for a complete optimization cycle on a workstation with typical industrial configuration (Intel i7, 16 GB RAM) is under 5 minutes, which is acceptable for practical shipyard planning. Therefore, the method maintains a balance between algorithmic sophistication and engineering applicability.

4.4 Experimental Setup

To validate the effectiveness of the proposed scheduling model and algorithm, a case study is conducted on the assembly of a WQ-type marine elevator used in naval vessels. The complete assembly process consists of 18 interdependent tasks, covering the installation of the shaft structure, guide rails, lifting platform, safety doors, hydraulic units, and control systems. The total baseline cycle is approximately 48 days under manual scheduling. The precedence relations among tasks are derived from the actual assembly workflow, and the resource demand

matrix reflects the requirements for workers and key equipment such as cranes and hoists. Each experimental run includes 50 stochastic samples for task durations, ensuring statistically consistent evaluation of schedule robustness.

In the experiment, a total of 15 workers is considered, each with different skill levels across mechanical, hydraulic, and electrical domains. The algorithm parameters are set as follows: population size of 100 for each sub-population, crossover probabilities of 0.8 (exploration) and 0.2 (exploitation), mutation probabilities of 0.2 (exploration) and 0.05 (exploitation), and a maximum of 100 generations.

For fitness scaling, the control coefficient ccc is adaptively adjusted according to both mutation probability and population evolution. A smaller ccc is used in early generations to suppress dominant individuals and preserve diversity, while a larger ccc is applied in later generations to strengthen the advantage of high-quality solutions. To distinguish early and late stages, the population similarity coefficient is introduced as follows:

$$\varphi = \frac{EX+1}{\sqrt{DX}} \quad (16)$$

where EX and DX are the expectation and variance of population fitness. Experimental observations show that for the exploration-oriented sub-population, the inflection point appears at $\varphi=17.9$ around generation 53, while for the exploitation-oriented sub-population, the inflection occurs at $\varphi=35.5$ around generation 42. These thresholds are adopted as reference points for dynamically adjusting the scaling coefficient c .

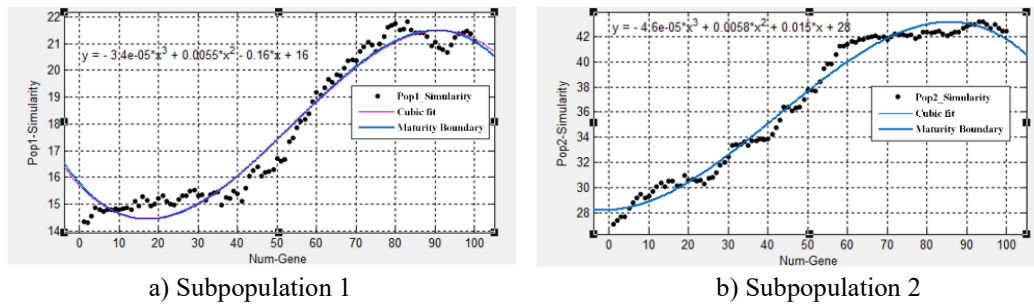


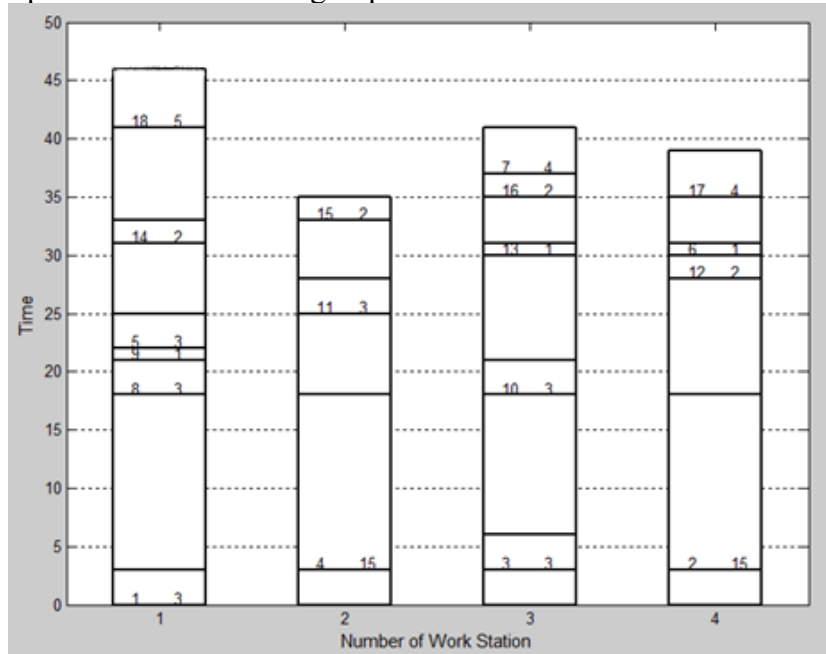
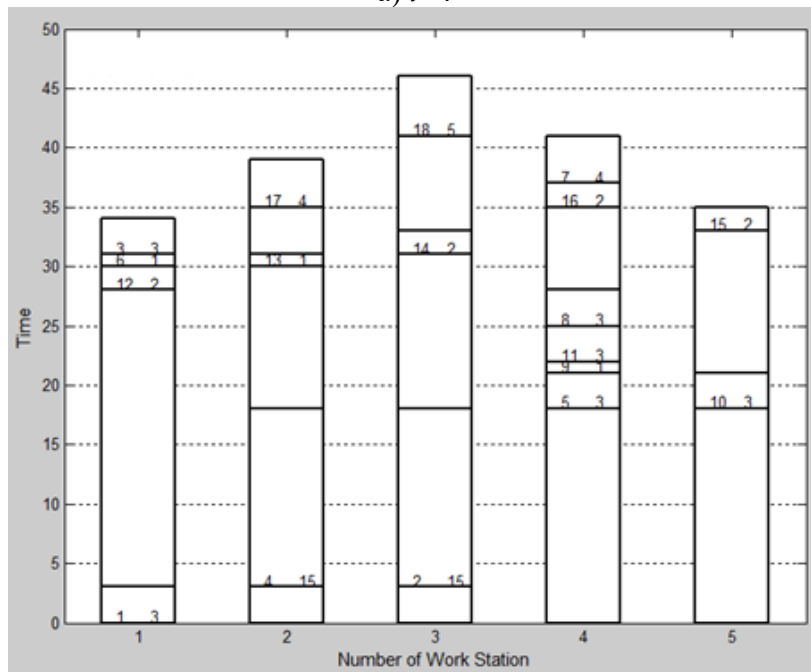
Fig. 8. Similarity Trend for Exploration Subpopulation 1 and 2

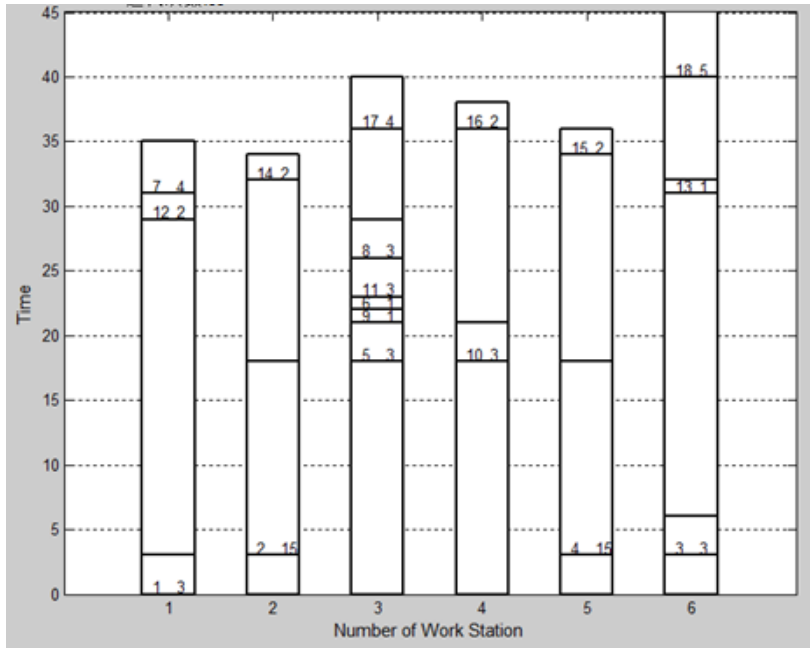
4.5 Results and Evaluation

The improved genetic algorithm was applied to optimize the scheduling of WQ-type marine elevator assembly under stochastic task durations, and results are reported as expected values across simulation runs. The iterative process was implemented in MATLAB R2023a, using the following input matrices: operation time, precedence relationships, resource requirements and availability, worker capabilities, process grouping, and unit-time material storage cost.

The optimization was performed for maximum simultaneous operations $l = 4, 5, \text{ and } 6$, yielding the final schedules shown in Figure 10. In these figures, the x -

axis represents simultaneously executable processes, the y-axis shows cumulative operation time, work sequence progresses from bottom to top. Bars are annotated with: left: process number and right: process duration.

a) $l=4$ b) $l=5$



c) $l=6$

Fig.10. Scheduling Optimization Results

Through extensive optimization trials, it was found that under current resource constraints and operation times, the maximum number of simultaneous operations is $l_{max}=3$. Optimal solutions can be obtained by setting $l \geq 3$. The scheduling chart for the minimum assembly cycle $T=45$ days and the corresponding worker allocation results are shown in Figures 11 and 12, where e_{i-j} represents operation number e_i with duration j .

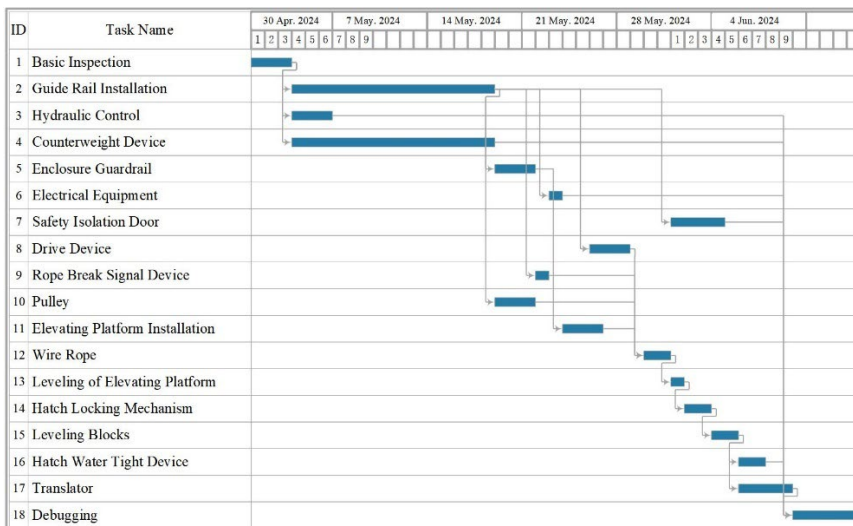


Fig. 11. Marine Elevator Assembly Scheduling

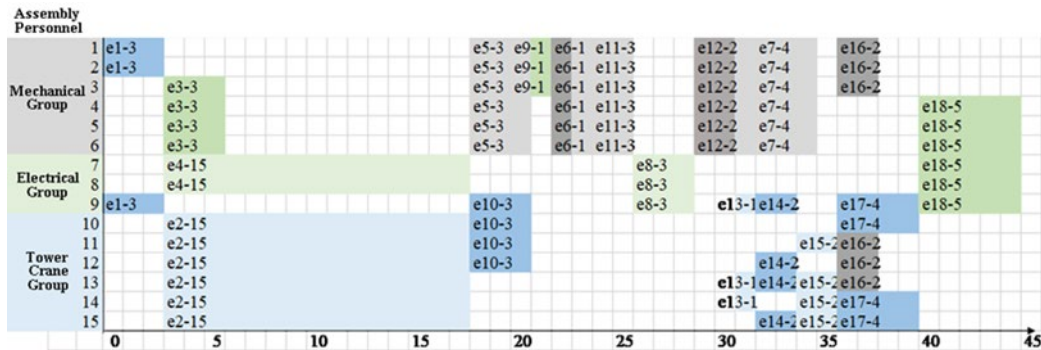


Fig. 12. Operation Assignment Results of Assembly Personnel

The scheduling results yield the minimum assembly cycle T , cumulative workload per worker (in days), capability index CI , and assembly process costs, as shown in Table 2. The worker load balance rate SI is calculated and compared with the pre-optimization assembly plan in Table 3:

The optimized schedule shortens the assembly cycle from 48 to 45 days, achieving a 6.25% improvement while meeting delivery requirements. Worker load balance is maintained below 1.39, ensuring traceable workload distribution and preventing fatigue, with more than 76% of worker-task assignments matching technical expertise, thereby enhancing proficiency and assembly quality. In terms of cost, actual inventory expenses remain under the budgeted 6750, and since delivery deadlines are met, delay costs remain zero. These results collectively demonstrate that the proposed method achieves significant improvements in efficiency, quality, and cost control for marine elevator assembly.

Table 2

Shortest Assembly Period and Cumulative Workload of Workers

l	T	Each Worker's Cumulative Workload / Day														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	46	19	19	19	21	21	21	21	22	22	22	22	22	23	23	23
5	46	21	21	21	23	19	19	23	23	19	23	22	22	22	21	21
6	45	21	21	21	21	21	21	20	20	20	22	22	22	22	22	24

Table 3

Final Assembly Period and Worker Workload Balance Rate

l	Time	Quality		Cost	
	Assembly Period T / Day	SI	CI	C_{inv}	C_{delay}
4	46	1.0110	81.71%	5945.17	0
5	46	1.3984	76.83%	6261.37	0
6	45	1.3499	79.26%	5689.31	0

To further validate the effectiveness of the proposed DPGA, comparative experiments were conducted against two representative metaheuristic algorithms: the Non-dominated Sorting Genetic Algorithm II (NSGA-II) and a hybrid Particle Swarm Optimization–Genetic Algorithm (PSO–GA). All algorithms were implemented in the same MATLAB R2023a environment, using identical

parameter settings for population size ($N = 100$) and maximum generations ($G = 100$). Each algorithm was executed five times under the same WQ-type marine elevator dataset, and the average results were recorded.

Table 4

Comparative Performance of DPGA and Baseline Algorithms

Algorithm	Assembly Cycle T	SI	CI	RCV	Avg. Runtime
Manual Scheduling	48	1.62	72.4	0.016	-
Standard GA	46	1.40	76.8	0.012	3.4
NSGA-II	46	1.41	77.8	0.011	6.8
Hybird PSO-GA	46	1.39	78.5	0.010	4.8
Proposed GA	45	1.35	79.3	0.009	4.7

Each value represents the mean of five independent runs under identical parameter settings. Lower T , SI , and RCV indicate better performance; higher CI indicates improved skill-task matching. The results in Table 4 show that the proposed DPGA achieves the shortest average assembly cycle and the best overall balance across all objectives. Although its runtime is slightly longer than that of the standard GA, it remains lower than NSGA-II and hybrid PSO–GA, confirming its computational efficiency. Therefore, DPGA maintains a favorable trade-off between convergence speed, optimization quality, and practical feasibility for large-scale fixed-position assembly scheduling.

5. Conclusions

This study addressed the scheduling problem of marine elevator assembly in naval shipbuilding, a fixed-position process characterized by long cycles, strict precedence constraints, and heavy reliance on skilled labor. A comprehensive scheduling model was developed that integrates precedence relations, resource allocation, worker assignment, and cost performance. To solve the model, an improved DPGA with integer encoding and precedence-preserving operators was proposed. The algorithm further incorporated fitness scaling and adaptive weighting to balance multiple objectives, including cycle time, workload balance, skill-task matching, and cost control.

A case study on the WQ-type naval marine elevator verified the effectiveness of the proposed method. Compared with manual scheduling, the DPGA significantly shortened the assembly cycle, improved workload balance, enhanced skill-task alignment, and reduced cost variance. These results demonstrate that the proposed approach can effectively improve the efficiency, fairness, and reliability of fixed-position assembly scheduling.

The methodology has broader implications beyond marine elevators. It can be applied to other fixed-position assembly systems in naval shipbuilding, such as weapon modules, radar systems, and power units, supporting improvements in overall shipbuilding efficiency and operational readiness. The proposed model already incorporates stochastic task durations and probabilistic resource constraints,

demonstrating its capability to generate robust schedules under uncertain shipyard conditions.

Acknowledgments

This work was supported in part by the National Defense Science and Technology Project Foundation under Grant 0106142, in part by the Ministry of Education in China Project of Humanities and Social Sciences under Grant 17YJC630139, and in part by the Fundamental Research Funds for the Central Universities under Grant 30917011303.

REFERENCES

- [1] *L. Wang*, Cutting plane algorithms for the inverse mixed integer linear programming problem, *Operations Research Letters*, Vol. **37**, Iss. 2, 2009.
- [2] *S. Boyd, J. Mattingley*, Branch and bound methods, Notes for EE364b, Stanford University, 2007.
- [3] *F. Rossi, P. V. Beek, T. Walsh*, Constraint programming, *Foundations of Artificial Intelligence*, Vol. **3**, 2008.
- [4] *S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi*, Optimization by simulated annealing, *Science*, Vol. **220**, Iss. 4598, 1983.
- [5] *X. Ma, Z. Wang, C. Wang*. An image encryption algorithm based on tabu search and hyperchaos, *International Journal of Bifurcation and Chaos*, Vol. **34**, Iss. 14, 2024.
- [6] *L. Abualigah, A. Sheikhan, Ikotun A. M., et al.* Particle swarm optimization algorithm: review and applications, *Metaheuristic Optimization Algorithms*, 2024.
- [7] *B. Alhijawi, A. Awajan*, Genetic algorithms: theory, genetic operators, solutions, and applications, *Evolutionary Intelligence*, Vol. **17**, Iss. 3, 2024.
- [8] *B. Doerr, T. Ivan, M. S. Krejca*, Speeding up the NSGA-II with a simple tie-breaking rule, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. **39**, Iss. 25, 2025.
- [9] *Y. Fu, Y. Wang, K. Gao, et al.* Review on ensemble meta-heuristics and reinforcement learning for manufacturing scheduling problems, *Computers and Electrical Engineering*, Vol. **120**, 2024.
- [10] *J. Xue, W. Ouyang, Y. Du, et al.* Simulation study on dynamic characteristic of marine elevator, *Proceedings of the International Conference on Mathematics, Modelling, Simulation and Algorithms (MMSA 2018)*, Atlantis Press, 2018.
- [11] *Z. Liu, Y. Mei, L. Jiang, et al.* Research of multi-process on assembly line balance cable products, *U.P.B. Science Bulletin, Series A*, Vol. **81**, Iss. 4, 2019.
- [12] *S. Ding, C. Su, J. Yu*, an optimizing BP neural network algorithm based on genetic algorithm, *Artificial intelligence review*, Vol. **36**, Iss.2, 2011.
- [13] *M. M. Salamattalab, M. H. Zonoozi, M. Molavi-Arabshahi*, Innovative approach for predicting biogas production from large-scale anaerobic digester using long-short term memory (LSTM) coupled with genetic algorithm (GA), *Waste Management*, Vol. **175**, 2024.
- [14] *G. Singh, A. K. Chaturvedi*, Hybrid modified particle swarm optimization with genetic algorithm (GA) based workflow scheduling in cloud-fog environment for multi-objective optimization, *Cluster Computing*, Vol. **27**, Iss. 2, 2024.