EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Research on digital workshop dynamic scheduling technology based on blockchain

Ganlong Wang[1], Jun Zhu[2*], Guoyin Zhang[1], Jingjing Sun[2], Wei Zheng[3], Bin Xie[4], Xiang Li[2]

[1]School of Computer Science and Technology, Harbin Engineering University, Heilongjiang, 150001, China
[2]NANJING HIGHWAY DEVELOPMENT (GROUP) CO., LTD, 210000, China
[3]Nanjing Communications Construction&Investment Holdings (Group)Co., Ltd.210008,China
[4]China Academy of Information and Communications Technology (CAICT), 100083, China

Abstract

INTRODUCTION: As a critical process in high-end equipment manufacturing, dynamic scheduling in digital workshops plays a vital role in order delivery and project cost control. However, traditional scheduling frameworks suffer from weak data security, opaque multi-agent collaboration, and slow constraint checking.
OBJECTIVES: To address these limitations, this study proposes an efficient and trustworthy dynamic scheduling solution that balances operational efficiency with data reliability.
METHODS: A three-layer collaborative framework comprising cloud, edge, and blockchain layers is designed. The cloud layer employs a multi-strategy improved multi-objective genetic algorithm with three-stage encoding targeting processes, equipment, and job teams. This algorithm integrates tournament selection, adaptive repair, and particle swarm optimization to globally minimize maximum delivery time, total energy consumption, and equipment load fluctuation. The blockchain layer leverages consortium blockchain and smart contracts for tamper-proof data storage and automatic constraint verification, while the edge layer handles real-time on-site scheduling.
RESULTS: The optimization algorithm converges to an optimal value of 55.1 in an average of only 3.9 iterations, outperforming three mainstream heuristic algorithms. In a real deployment across 57 workstations in a cruise ship laser sheet metal workshop, scheduling tasks are completed within 25 seconds. Blockchain technology achieves a 100% data tampering detection rate, improves data consistency from 96.2% to 99.8%, and reduces scheduling deviation from 4.39% to 3.82%, with only a 3-second processing overhead.
CONCLUSION: The integrated architecture enhances both scheduling efficiency and data reliability, supporting the digital and intelligent transformation of shipbuilding enterprises.

Keywords: Blockchain; Dynamic Scheduling; Cloud-Edge Collaboration; Multi-objective Genetic Algorithm; Smart Contract

Received on 13 June 2026, accepted on 07 July 2026, published on 13 July 2026

doi: 10.4108/ eetsis.13503

1. Introduction

Ship section fabrication is a core component of shipbuilding, characterised by a production process involving multiple product varieties, small batch sizes and high complexity. A typical workshop comprises dozens of workstations, various types of equipment and multiple work teams, with strict process route constraints and resource exclusivity constraints between operations. Furthermore, the production floor is frequently subject to dynamic disturbances such as equipment failures, urgent order insertions and material delays, requiring the scheduling system to generate feasible plans within a short timeframe.

These characteristics make shipyard workshop scheduling, in essence, a strongly constrained, multi-objective, and highly dynamic combinatorial optimisation problem. Traditional static scheduling cannot cope with dynamic disturbances, whilst existing dynamic scheduling solutions largely rely on cloud-edge collaboration architectures to implement a two-tier model of 'cloud-based optimisation and edge-based

*Corresponding author. Email: 18251881199@163.com

execution'. However, this model has revealed three key unresolved issues in practical implementation:

(1) Insufficient data reliability: The process of distributing production schedules from the cloud to the edge lacks tamper-proof mechanisms, making it difficult to detect any modifications once they have been made;

(2) Insufficient transparency in multi-party collaboration: Data consistency across edge nodes relies on a central node, creating a single point of failure risk, and there is a lack of transparency regarding the right to know the scheduling status among all parties;

(3) Delayed constraint validation response: The validation of constraints such as process routes, team exclusivity and equipment capacity relies on centralised programmes or manual review, making it difficult to meet the demand for rapid rescheduling under dynamic disturbances.

The three issues mentioned above are intertwined and collectively limit the practicality and reliability of digital workshop scheduling systems for shipbuilding. Existing research has focused on different aspects of these three areas, but clear technical gaps remain in each.

At the level of scheduling optimisation algorithms, Zhang et al. [1] proposed a genetic algorithm-based method for ship section production scheduling that takes into account process and resource constraints; however, it does not address dynamic disturbance scenarios, making it difficult to respond to unexpected job insertions or equipment failures in actual workshops. Liu et al. [2] applied the particle swarm optimisation algorithm to ship section scheduling, achieving a faster convergence rate than GA; however, the algorithm lacks stability under complex constraints, and the quality of the solutions fluctuates significantly. Wang et al. [3] proposed the Genetic Simulated Annealing Hybrid Algorithm (GASA), which improves both solution quality and convergence speed; however, it remains within a single-objective framework and does not address multi-objective trade-off issues. Although Zhang et al. [4] employed a multi-strategy enhanced genetic algorithm to solve the dynamic scheduling of shipyard workshops, incorporating local search and adaptive operators, their strategy combination remained an internal adjustment within the algorithm and was not coupled with external data reliability or multi-party coordination mechanisms. Wang et al. [5] proposed a hybrid genetic algorithm for ship section fitting, which performed well in static scheduling but similarly lacked joint handling of dynamic disturbances and multi-objective constraints. Dai et al. [6] introduced an improved NSGA-III into multi-objective optimisation for ship production scheduling, achieving improvements in the Pareto front distribution; however, the algorithm's solution stability under strong constraints remains unsatisfactory, and it does not address issues of data reliability and collaborative transparency at the workshop level.

At the architectural and dynamic response levels, Li et al. [7] proposed a cloud-edge collaborative workshop scheduling architecture, with the cloud responsible for global optimisation and the edge for real-time execution, balancing global optimality with local response speed. However, this architecture is semi-centralised in design; edge-side decisions lack independent and trustworthy verification, and the transparency of multi-party collaboration is not guaranteed. Gao et al. [8] further proposed a three-tier cloud-edge-end collaborative framework, introducing real-time scheduling capabilities at the edge and combining deep reinforcement learning to enhance dynamic response; however, its collaborative mechanism still relies on a central node acting as a trust intermediary, failing to resolve the issue of mutual trust among multiple parties. Liu et al. [9] introduced edge-computing-assisted real-time scheduling into a digital twin workshop, achieving significant improvements in response latency; however, the credibility of the data sources was not verified, and the architecture similarly lacks a transparent coordination mechanism across entities.

At the level of blockchain and trusted collaboration, Xu et al. [10] proposed a blockchain-based solution for trusted data sharing in the Industrial Internet of Things (IIoT), addressing the issue of tamper-proofing device status data; however, this solution was not deeply integrated with scheduling algorithms and remained confined to the data layer. Wang et al. [11] applied smart contracts to the automatic verification of production scheduling constraints, achieving transparent rule enforcement; however, their application scenario was general smart manufacturing and was not adapted to the stringent constraints and dynamic scheduling characteristics of discrete shipbuilding. Chen et al. [12] proposed a dynamic workshop scheduling method based on blockchain smart contracts, making progress in the trustworthy execution of constraints; however, their experiments were validated only on small-scale standard test cases and were not tested in the complex scenario of shipbuilding. Kolhar et al. [13] and Ferrag et al. [14] reviewed the current state of research and challenges regarding blockchain in the Industrial Internet of Things (IIoT), whilst Saberi et al. [15] and Kshetri [16] explored blockchain applications from a supply chain sustainability perspective; however, these works remain at the proof-of-concept or review stage and have not yet been deeply integrated with workshop scheduling algorithms. The PBFT consensus mechanism proposed by Castro et al. [17] provides a fault-tolerant foundation for multi-party collaboration in consortium blockchains; however, balancing consensus latency with real-time responsiveness in scheduling scenarios remains an open problem.

Overall, existing work has made progress in three areas—algorithm optimisation, dynamic architecture, and trusted collaboration—but each has significant limitations: at the algorithmic level, there is a lack of joint handling of multi-objective problems and dynamic disturbances; at the architectural level, there is a lack of transparency guarantees for multi-party collaboration; and at the blockchain level, there is a lack of deep integration with scheduling algorithms. Particularly in the complex discrete manufacturing scenario of shipbuilding—which is characterised by strong constraints, multiple stakeholders [18], and high dynamics—interdisciplinary research across these three areas is extremely scarce. This paper addresses this gap by proposing a scheduling scheme for digital shipyards that integrates and enhances multi-objective optimisation, cloud-edge-end collaboration, and consortium blockchain smart contracts.

In summary, existing research suffers from the following shortcomings, As shown in Table 1:

Table 1. Limitations of existing studies

No.	Shortcomings	pecific Manifestations
1	Insufficient stability of dynamic scheduling algorithms under stringent constraints	Existing hybrid algorithms are prone to generating infeasible solutions when process route constraints and team exclusivity constraints coexist
2	Lack of assurance regarding data integrity	The transmission and execution of production schedules in cloud-edge architectures lack an tamper-proof audit trail mechanism
3	Risk of single points of failure in multi-party collaboration	Data consistency between edge nodes relies on a central node; a failure at the central node will result in the interruption of collaboration
4	Insufficient transparency in collaboration	The right of various teams and production lines to be informed about the status of scheduling execution is unclear, which can easily lead to information asymmetry
5	Delayed constraint verification response	Verification of various constraints—such as process, equipment and team constraints—relies on centralised execution, failing to meet the timeliness requirements of dynamic rescheduling
6	Insufficient integration of blockchain with scheduling algorithms	Existing research predominantly utilises blockchain at the data-sharing level, failing to embed it within the core processes of scheduling decision-making and constraint verification

To address the aforementioned shortcomings, this paper builds upon the existing multi-workstation scheduling technology based on cloud-edge collaboration by introducing blockchain technology. It constructs a three-tier ‘cloud-edge-chain’ collaborative scheduling architecture and conducts engineering validation across 57 workstations in the laser thin-sheet workshop at China Merchants Cruise. The main contributions are as follows:

(1) A ‘cloud-edge-chain’ three-tier architecture is proposed, integrating the immutability and smart contract features of blockchain into the workshop’s dynamic scheduling process, thereby addressing issues of data reliability and collaborative transparency;

(2) A smart contract-based automatic constraint verification mechanism is designed, encoding process route constraints, team exclusivity constraints, and equipment constraints into on-chain smart contracts to achieve transparent, automatic, and real-time verification of constraint conditions;

(3) A consortium blockchain consensus mechanism was established for 57 workstations, ensuring data consistency across multiple edge nodes and eliminating the risk of single points of failure;

(4) A multi-objective genetic algorithm was improved, combined with multi-strategy optimisation for cloud-edge collaboration, to enhance production rescheduling efficiency and solution quality under dynamic disturbances.

2. Blockchain-enhanced scheduling architecture design

2.1. Overall design of the three-tier collaborative scheduling architecture for "cloud-edge-chain"

Building upon the existing cloud-edge collaborative dynamic scheduling technology for ship digital workshops, this paper innovatively integrates blockchain technology to establish a three-tier collaborative scheduling architecture comprising "cloud-edge-chain."[19] Building upon the original cloud-edge collaboration framework (Figure 1), this architecture introduces an additional chain-layer between the cloud and edge endpoints, creating a closed-loop scheduling process characterized by "global optimization in the cloud—trustworthy evidence storage and constraint validation at the chain layer—real-time execution at the edge."

Cloud Layer: Maintains the global optimization capability of the original multi-objective genetic algorithm, responsible for receiving medium-and long-term order data input from the PPMS system (including planned start and completion times for seven stages: material preparation, preprocessing, cutting, processing, preliminary assembly, intermediate assembly, and segmented construction). It executes an improved genetic algorithm to generate the optimal production scheduling plan and packages key data—including the scheduling plan, constraint conditions, and objective function values—for blockchain storage.

Chain End Layer: This is the newly added core layer of the paper, deployed based on a consortium blockchain architecture and comprising three major functional modules: ① Data On-Chain Storage and Verification Module, which stores key data such as production scheduling plans, execution status, and equipment utilization rates in block format to ensure immutability; ② Smart Contract Constraint Verification Module, which encodes process route constraints, team exclusivity constraints, and equipment constraints into on-chain smart contracts to automatically verify the legality of scheduling plans; ③ Consortium Blockchain Consensus Module, ensuring data consistency

among the 57 workstations across multiple edge nodes and eliminating single-point failure risks.

Border Layer: Maintains the real-time scheduling execution capability of the original 57 workstations, receives production scheduling plans from the cloud, verifies constraints via smart contracts before executing scheduling instructions, and returns the execution status in real-time to the blockchain for record-keeping.

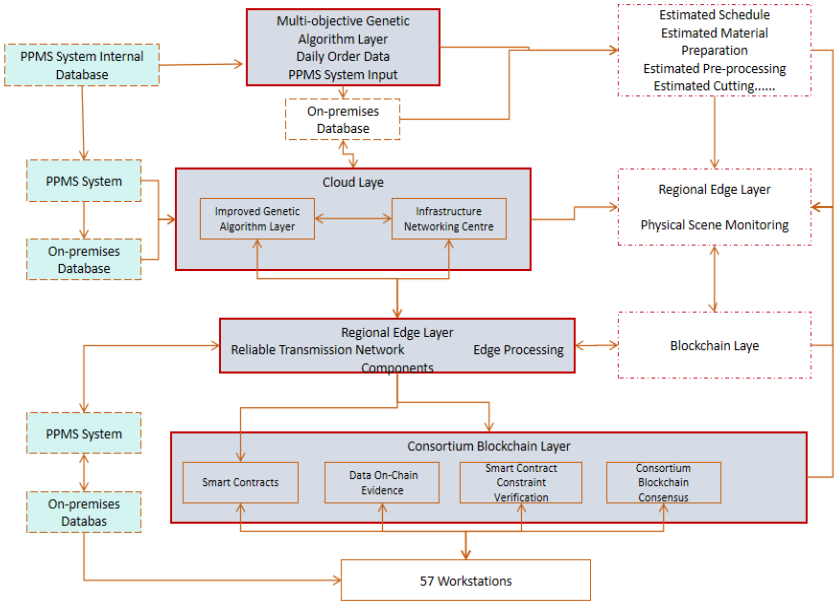

Figure 1. Three-layer collaborative scheduling architecture of "Cloud-Edge-Chain"

The three layers described above are not simply stacked on top of one another, but form a bidirectional coupling through on-chain smart contracts and cloud-based genetic algorithms: the production scheduling plans generated by the cloud in each iteration must undergo five types of constraint hard validation via on-chain smart contracts; only upon passing these checks can they enter the next generation of the population; The real-time execution status of the 57 workstations at the edge is transmitted back to the cloud following on-chain consensus, driving the adaptive repair operators of the algorithm and the dynamic updating of the equipment maintenance variable δj. The synergistic relationship between the two will be further elaborated in Chapter 3, which details the algorithmic improvements.

2.2. Detailed design of chain end functional modules

2.2.1. Chain-based Data Storage and Verification Module

The four categories of critical data during production scheduling must be recorded on the blockchain,As shown in Table 2:

Table 2. Four categories of key data in the production scheduling process

data type	On-chain content	Time to deploy on the blockchain	Purpose of Evidence Preservation
Producti on Scheduling Plan Data	Code results for process sections, equipment sections, and team sections	After generating the production scheduling plan in the cloud	Tamper-proof and traceable
Constraint Condition Data	Process Operation Connection Relationshi p Matrix W, Resource Matrix N	Production Input Phase	Transparency of Constraints
Execution status data	Real-time start/finish times and equipment utilization rates for each workstation	During the edge execution process	Progress is monitorable

Optimize target data	Load balancing factor U, completion time T, variance σ	After each iteration is completed	The results are auditable.

Taking the practical application at China Merchants Cruise's laser thin plate workshop as an example, the production scheduling algorithm completes its calculations in approximately 25 seconds. Each small rectangle in the generated equipment schedule diagram represents a specific process duration that must be recorded on the blockchain for verification, ensuring full traceability throughout the scheduling and execution of all 57 workstations.

To intuitively demonstrate the actual storage format of the four categories of on-chain data defined in Table 2, we provide a simplified text snippet of block data records as follows, which covers scheduling plan, constraint matrix, equipment utilization and optimization target information:

plaintext
Block Height: 1258
Timestamp: 2025-09-15 14:26:18
1. Production Scheduling Plan Data
Process segment code: [1,1,2,2,3,...], Equipment segment code: [3,5,2,7,1,...], Team segment code: [1,1,2,3,2,...]
2. Constraint Condition Data
Process-operation connection matrix W=[[1,0,0],[1,1,0],[0,1,1]], Resource matrix N=[3,12,3]
3. Execution status data
Workstation-01: Start=14:00, Finish=16:30, Utilization=0.86; Workstation-02: Start=13:10, Finish=15:45, Utilization=0.79
4. Optimize target data
Max completion time Tmax=168h, Total energy consumption=4268kWh, Equipment utilization variance σ =0.032

Consensus Signatures: Node01, Node04, Node09,..., Node57 (Pass PBFT 2/3 majority verification)

This snippet shows the standardized plaintext storage structure of core scheduling data on the consortium blockchain. All key values are packaged into block records after the cloud layer generates the scheduling scheme, and the complete records cannot be tampered with after multi-node consensus verification, realizing full traceability of workshop scheduling data.

2.2.2. Smart Contract Constraint Verification Module

Encode the eight constraint conditions from the original scheduling model into smart contracts to enable automatic validation [20]:

(1) Process Route Constraint Contract: Corresponding to the original constraint (1), the encoding rule is as follows — when the immediate preceding process of step n is incomplete, the start event of this step is rejected by the contract and is only released after the completion event of the preceding process is recorded on the blockchain.

(2) Team Exclusive Constraint Contract: Corresponding to the original constraint (2), the coding rule specifies that a single team member (p) can execute only one task at any given time. When a new team assignment request is submitted, the contract automatically checks the team's current time slot occupancy status; if the slot is already occupied, the assignment is rejected [21].

(3) Equipment Constraint Contract: Corresponding to the original constraint (6), the coding rule specifies that each device can process only one workpiece at any given time. The contract continuously verifies the occupancy status of equipment time slots in real-time.

(4) Processing Sequence Constraints Contract: Corresponding to the original constraint (7), the coding rule specifies that fuzzy variables determine whether a process is executed on designated equipment, automatically verifying the validity of the processing sequence.

(5) Completion Time Verification Contract: When the planned completion time (Tend) of the production scheduling exceeds the expected latest deadline, the contract is automatically deemed overdue, triggering a re-scheduling process.

2.2.3. Consortium Chain Consensus Module

For the complex collaborative scenarios involving 57 workstations with multiple teams, devices, and production lines, the consortium blockchain consensus mechanism replaces the traditional central node coordination model. Each edge node (workstation) serves as a ledger node on the consortium blockchain, achieving data synchronization through the Practical Byzantine Fault Tolerance (PBFT) consensus algorithm to ensure [22]:

Any change in the scheduling execution status of an arbitrary node must be confirmed by more than two-thirds of the nodes before being recorded on the blockchain.

A single node failure does not affect the overall data consistency.

All stakeholders have equal access to information regarding the dispatch execution status, eliminating information asymmetry.

3. Improvement Details of the Multi-objective Genetic Algorithm Based on Cloud-Edge Collaborative Multi-strategy Optimization

3.1. Overall algorithm framework

The improved genetic algorithm proposed in this paper aims to achieve three primary optimization objectives: minimizing the maximum completion time (Tmax), minimizing total energy consumption, and minimizing the variance of equipment utilization rate (σ), corresponding to the mathematical functions F1, F2, and F3 respectively.

The algorithm input includes the segmented task matrix J, segmented start times Tstart, latest completion time Tend, bill of materials M, process-process linkage matrix W, current production execution status S for each workstation, workshop workstation-device relationship matrix E, and resources N. Using matrices M, W, E, and N, the algorithm generates all manufacturing processes in J and establishes hierarchical (process segment) and sequential relationships between them. It then iteratively processes the segmented task matrix J until all tasks are completed.

The coupling between this algorithm and blockchain manifests in two ways. Forward coupling: the three-part encoded scheduling plans generated by the genetic algorithm in each iteration must, prior to fitness evaluation, pass five types of constraint checks—executed via on-chain smart contracts—covering process routes, team exclusivity, machine processing, processing sequence, and deadline violation—corresponding to Equations (8) to (12). Individuals failing these checks are immediately marked as invalid solutions and eliminated, thus not participating in crossover and mutation. This transforms the soft constraints of traditional penalty functions into hard filtering. Reverse Coupling: The execution status of the 57 workstations at the edge (start/completion times, equipment utilisation) is recorded on the blockchain via PBFT consensus. This data serves as anchor data for the algorithm's adaptive repair operator, whilst simultaneously directly driving the values of the equipment maintenance 0-1 variable δj (Formula ②), enabling the objective functions F1, F2, and F3 to be recalculated in real time based on actual workshop conditions.

3.2. Improvement of the multi-objective genetic core algorithm

After clarifying the overall framework and input/output workflow of the algorithm, this paper establishes a comprehensive core algorithm improvement system to transform the multi-dimensional optimization objectives in ship segment construction scheduling into computable mathematical models. The system encompasses eight key formulas: total workpiece processing time, equipment utilization rate, total completion time, average process time, equipment utilization variance, load balance level, as well as constraints including process route limitations, team exclusivity constraints, equipment processing constraints, processing sequence constraints, and overdue detection criteria. These formulas provide a quantitative basis for subsequent fitness evaluation using the three-stage genetic encoding approach and the implementation of multi-strategy improvement operators.

3.2.1. Basic Calculation layer: workpiece processing time and equipment utilization rate

The basic calculation layer provides the foundational data support for the entire optimization model, serving as the basis for all subsequent objective functions and constraint conditions.

(1) Total machining time for the workpiece

$$T_{total} = \sum_{m\in M}\sum_{n\in N} T_{mn} + \sum_{j=1}^{L} t_{repair,p} - \sum_{k=1}^{L_{working}} t_{repairp}\frac{W_k}{v_k} \quad (1)$$

Here, tmn denotes the total processing time for workpiece n on production line process m; m represents the number of production line processes, n indicates the target number of processed workpieces, trepair is the equipment maintenance time, j stands for the processing speed at a single workstation, L is the total number of maintenance workstations, and Lworking refers to the number of operational workstations. This formula comprehensively accounts for the impact of both normal processing time and equipment maintenance on the total labor hours.

(2) Equipment Maintenance Variable 0-1

$$\delta_j = \begin{cases} 0, j = 0 \\ 1, j \geq 1 \end{cases} \quad (2)$$

All machines at each workstation are in standby maintenance mode; the production line is not operating, and the machines are functioning normally for processing.

Here, δj is a variable ranging from 0 to 1. When δj = 0, all machines at each workstation are in standby for maintenance, and the production line remains idle; when δj = 1, machines operate normally for processing. This variable serves as a linkage mechanism in the subsequent calculations of equipment utilization rate and total completion time.

(3) Equipment utilization rate

$$U_{eq} = \frac{T_{working}}{T_{total}} \times 100\% \quad (3)$$

Here, Ueq denotes equipment utilization rate, whose value objectively evaluates whether the equipment is fully utilized, thereby optimizing practical scenarios. The equipment utilization rate serves not only as one of the optimization objectives but also as the fundamental input for calculating its variance (Formula ⑥) and load balancing degree (Formula ⑦).

3.2.2. Objective function layer: three-objective optimization system

The objective function layer uses the output from the basic calculation layer as input to construct a three-objective optimization model aimed at minimizing the maximum completion time, total energy consumption, and equipment utilization variance.

(4) Total completion time Tmax

$$T_{max} = \max_{m\in M}\{T_m\} \quad (4)$$

Here, Tm denotes the completion time of segment m, while the total completion time is determined by the maximum value among all segment completion times. This metric directly reflects the overall production cycle of the workshop

and is influenced jointly by the total processing time of workpieces and equipment utilization rate—higher equipment utilization and fewer maintenance downtime result in a shorter Tmax. Additionally, this metric, together with the average process time (Formula ⑤), forms the evaluation framework for the completion time dimension.

(5) Average Process Time Tflow

$$T_{flow} = \frac{1}{N}\sum_{m}^{M}\sum_{p=1}^{P}\sum_{i=1}^{I} t_{pmi} \tag{5}$$

Here, N represents the number of work tasks in the ship segment manufacturing workshop, and tpmi denotes the time required by team p of segment m to complete task i. The average process time and total completion time T_max are complementary: T_max focuses on the longest completion time, while T_flow measures the average duration of all tasks; their combination provides a more comprehensive evaluation of the scheduling scheme's time efficiency.

(6) Equipment utilization variance σ

$$\sigma^2 = \frac{1}{K}\sum_{k}^{K}\left(U_k - \overline{U}\right)^2 \tag{6}$$

Here, K represents the total number of devices, Uk denotes the utilization rate of the k-th device, and Ū is the average utilization rate across all devices. This metric, based on device utilization (Formula ③), measures the dispersion in individual device utilization rates. A smaller σ indicates more balanced device loads; together with the load balance degree U (Formula ⑦), it forms the evaluation framework for the device load dimension.

(7) Load balancing level U

$$U = \frac{\left(\sum_{i=1}^{m} X_i\right)^2}{m \times \sum_{i=1}^{m} X_i^2} \tag{7}$$

In the formula, Xi denotes the time required to complete task i, m represents the total number of tasks, and β is the weight factor. A value closer to 1 indicates more balanced workload distribution. Its calculation aligns with the equipment utilization variance σ: σ measures balance at the equipment level, while U measures it at the task level; together they ensure the scheduling scheme prevents equipment overload and task backlogging.

3.2.3. Constraint verification layer: five types of scheduling constraints

The constraint validation layer uses the production scheduling results from the objective function layer as input and evaluates the feasibility of the scheduling plan through five types of constraints. Among these, the process route constraint (Formula ⑧) serves as the foundation for all constraints; the team exclusivity constraint (Formula ⑨) and the equipment processing constraint (Formula ⑩) constitute the core constraints at the resource level; the processing sequence constraint (Formula ⑪) provides supplementary constraints at the process level; and the overdue determination condition (Formula ⑫) serves as the final verification of the total completion time T_max (Formula ④).

(8) Sequential constraints on the process route

$$T_{mn} \geq T_{m,n-1} + t_{m,n-1}, \forall n > 1 \tag{8}$$

Here, Tmn denotes the completion time of task n in segment m, while tm,n−1 represents the time required to complete task n−1 in segment m. Subsequent tasks can only commence after the preceding task is completed. This constraint serves as the logical starting point for production scheduling generation; all subsequent team exclusivity constraints and equipment processing constraints must be enforced only when this constraint is satisfied.

(9) Team-exclusive constraint

$$\sum_{m=1}^{M}\sum_{n=1}^{N} X_{pmn}, \forall p, \forall t \tag{9}$$

Here, P denotes the total number of teams, p represents a team, and Xpmn is the control variable: it equals 1 when task n in segment m is completed by team p, and 0 otherwise. At any given time, a team can only execute one task. This constraint, together with the equipment processing constraint (Formula ⑩), forms the resource exclusivity constraint, which restricts task execution concurrently from the dimensions of human resources and equipment resources, respectively.

(10) Equipment Processing Constraints

$$t_{end,i} \leq t_{start,j} + \frac{s}{v}, \forall i \neq j \tag{10}$$

Here, t_i denotes the completion time of process i on the production line equipment, s_i represents the number of workpieces processed by each process, and v_i indicates the unit processing time per workpiece. Each piece of equipment can process only one workpiece at any given time. This constraint, together with the team exclusivity constraint (Formula ⑨), establishes dual resource limitations, ensuring that the production scheduling plan avoids conflicts between human and equipment resources.

(11) Processing Sequence Constraint

$$t_{start,i} + t_{process,i} \leq t_{start,j} + M\left(1 - y_{ij}\right), y_{ij} \in \{0,1\} \tag{11}$$

Here, tstart,i denotes the start processing time of process i, tprocess,i represents the processing time of process i on the production line equipment, and yij is a fuzzy variable indicating whether process i is performed on equipment j—1 if yes, 0 otherwise. This constraint refines the process route constraint (Formula ⑧) at the equipment level, ensuring that the processing sequence on the same equipment complies with process requirements.

(12) Overdue Judgment Conditions

$$T_{end} > T_{deadline} \Rightarrow \mathrm{Re}-schedule \tag{12}$$

When the planned completion time Tend of the production scheduling plan exceeds the expected latest completion time Tdeadline, it is deemed overdue, and the scheduling process is restarted. This determination is based on the total completion time T_max (Formula ④) and serves as the final output of the entire constraint validation layer. If the plan passes all four constraint checks—process route, team exclusivity, equipment processing, and processing sequence—but fails the overdue determination, the algorithm must be rerun for optimization.

In this architecture, the validation of the aforementioned five types of constraints is not performed by the algorithm's internal penalty function, but is instead carried out independently by a smart contract on the blockchain. Specifically, once the cloud-based genetic algorithm has generated candidate scheduling plans, the codes for the process segments, equipment segments and workgroup segments of the plan are packaged and uploaded to the blockchain, where the on-chain contract validates them one by one according to Equations⑧-⑫; Only those plans that pass all checks are assigned a valid fitness value and proceed to the NSGA-II non-dominance sorting stage; those that fail are immediately discarded. This mechanism upgrades constraint verification from soft processing within the algorithm to hard filtering on-chain, thereby fundamentally eliminating the problem of invalid solutions contaminating the population.

3.3. Three-stage gene coding design

Each gene consists of three segments: the process segment, the equipment segment, and the team segment. This constitutes the core encoding mechanism that distinguishes the algorithm presented in this paper from traditional genetic algorithms.

Process Section: A fixed-length array whose length equals the sum of process counts across all tasks. The array elements represent the IDs of all tasks, with each task ID appearing as many times as its corresponding number of processes.

Device segment: A fixed-length array where each element represents the processable device corresponding to the current index within the process segment.

Team segment: A fixed-length array where each element represents the processable team corresponding to the current index within the process segment.

To improve the reproducibility of the three-stage encoding strategy, we provide a concrete calculation example based on the experimental dataset containing 13 ship segments.

The workshop involves 7 manufacturing stages for each segment: material preparation, preprocessing, cutting, machining, sub-assembly, intermediate assembly, and segmented construction. In total, there are13 × 7=91independent manufacturing processes.

Process segment array length: Equal to the total number of all processes, which is 91. Each element stores the unique ID of one manufacturing process.

Equipment segment array length: Consistent with the process segment (91). Each element records the serial number of the equipment matched to the corresponding process position.

Team segment array length: Also 91. Each element represents the work team assigned to the corresponding process.

For the 13 ship segments tested in this paper, every gene individual consists of three equal-length arrays with a length of 91 each. This example clearly reflects the mapping relationship between actual workshop task volume and gene dimension, and facilitates repeated implementation of the encoding logic by other researchers.

3.4. Multi-strategy improvement mechanism

3.4.1. Championship Selection

A random sample of individuals is selected from the population, and the most fit individuals are chosen as parents. This approach avoids local optima while ensuring the quality of the parent generation.

3.4.2. Cross-operation

By cross-referencing the locations of process segments, equipment segments, and workgroup segments, a new production scheduling plan is generated that integrates the best features from different components. Subsequently, through fitness evaluation calculations, those with lower fitness scores are eliminated while retaining those with higher fitness scores.

3.4.3. Variant operations and adaptive repair

By randomly rearranging the positions of process segments, equipment segments and workgroup segments, the key improvement lies in the adaptation of the genetic operator: the mutation operation is no longer a purely random perturbation, but instead uses the real-time operational status S recorded at the end of the chain as an anchor, prioritising targeted adjustments within the affected process segments and equipment segments. Concurrently, when the smart contract at the chain end detects that a piece of equipment has entered a maintenance state (δj=0, Equation ②), the algorithm automatically isolates the corresponding process from the population to prevent the generation of infeasible solutions.

3.4.4. Speed-position update mechanism (particle swarm fusion)

Each particle updates its velocity and position based on its current position and velocity, as well as the relationship between the global optimal position (global optimal solution) and the individual optimal position (local optimal solution):

$$v_i^{t+1} = w \cdot v_i^t + c_1 r_1 \left(pbest_i - x_i^t \right) + c_2 r_2 \left(pbest - x_i^t \right)$$
$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (13)$$

Here, w denotes the inertia weight; c1 and c2 are learning factors; r1 and r2 are random numbers; pbesti represents the individual optimal position, while gbest indicates the global optimal position.

To complete the closed-loop logic of dynamic rescheduling, we supplement the deviation judgment formula and threshold definition here. The relative progress deviation is defined as:

$$(\Delta_{\text{dev}} = \frac{|T_{\text{real}} - T_{\text{plan}}|}{T_{\text{plan}}}) \quad (14)$$

Combined with historical data of 57 workstations, the empirical thres$holdissetto(\varepsilon = 0.05).When(\Delta_{dev} > 0.05)$,

the smart contract triggers rescheduling and uploads real-time workshop status to the cloud for re-optimization. In Algorithm CECS Line 37, $(\varepsilon)adop$ts this fixed empirical value calibrated from actual workshop production data.

We analyze the asymptotic complexity of three core modules to discuss the scalability of the CECS architecture. Let

N= number of workstations,

M= total production tasks,

P= population size,

G = maximum GA iterations.

Improved genetic iteration: Single iteration complexity O(P·M), total complexity O(GPM). This cost depends on task quantity rather than workstation number N.Single smart contract verification: Constant complexity O(1), independent of N.PBFT consensus: Complexity O(N 2). Consensus overhead rises quadratically with the increase of workstation nodes.

In summary, genetic optimization and on-chain constraint verification will not bring extra latency when expanding workshop nodes. The PBFT module is the scalability bottleneck: it works efficiently under $N \leq 60$(our 57-workstation scenario), but causes obvious delay growth when N>100. This theoretical conclusion matches the engineering limitation summarized in Section 5.

3.5. Algorithm performance verification

Experimental environment: Intel Xeon Gold 6248R CPU, 64GB RAM, Ubuntu 20.04, Python 3.9.

Dataset: Actual data from the China Merchants Cruise laser sheet metal workshop, comprising 57 workstations, 13 PCTC car carrier sections, and 7 manufacturing stages (material preparation, pre-processing, cutting, machining, sub-assembly, intermediate assembly, and section construction).

Parameter settings: Population size 100, crossover probability 0.85, mutation probability 0.1, maximum number of iterations 500. Comparison algorithms: GA, PSO, GASA. The improved genetic algorithm was run 10 times each in comparison with GA, PSO and GASA; The results of the analysis are shown in Table 3:

Table 3. Analysis of statistical results

algorithm	Average number of iterations	maximal solution	minimal solution	Average solution
GA	61.7	68	58	60.5
PSO	40	59	55	57.1
GASA	12.8	57	57	57
This algorithm	3.9	56	55	55.1

The results demonstrate that the improved genetic algorithm requires only 3.9 average iterations to achieve the optimal solution (55.1), with the smallest deviation. It outperforms the other three algorithms in convergence speed, global optimization capability, and algorithm stability. In the test involving 8 workpieces and 33 processes, the algorithm obtained the optimal solution after just 5 iterations (total time: 25).

The collaborative process between the aforementioned cloud-based genetic algorithm and the edge-based blockchain can be summarised in the following pseudocode:

```
Algorithm: CECS (Cloud-Edge-Chain Scheduling)
Input: J, M, W, N, S, Tdeadline
Output: Π*, L (optimal solution + audit trail)
═══ Phase One: Main Loop of the Cloud-Based Genetic Algorithm ═══
1  Pop ← InitPopulation(100), gen ← 0
2  while gen < 500 do
3    for each ind ∈ Pop do
4                    Π  ←  Decode(ind.process_segment, ind.equipment_segment, ind.work_team_segment)
5      Ueq ← CalcUtilization(Π, δj)        // Formula ③
6      Tmax ← CalcMakespan(Π)              // Formula ④
7      σ  ← CalcVariance(Ueq)              // Formula ⑥
8      F  ← (Tmax, Energy, σ)              // Three objectives
9    end for
10
11   // ★ Blockchain coupling point ①: Smart contract hard validation
12   for each ind ∈ Pop do
13     if SmartContract.Validate(Π) == false then
14       Fitness(ind) ← INF   // Hard elimination
15     else
16       Fitness(ind) ← NSGA2_Rank(F)
17     end if
18   end for
19
20   Parents ← TournamentSelect(Pop)        // Equation ③
21   Offspring ← Crossover(Parents) + PSO_Update()
22     Offspring  ←  AdaptiveRepair(Offspring, S) // S comes from the chain endpoint
23   Pop ← ElitistSurvival(Pop ∪ Offspring)
24   gen ← gen + 1
25 end while
26 Π* ← GetBestFront(Pop)
═══ Phase Two: Chain-End Evidence and PBFT Consensus ═══
27 Block ← Pack(Π*, F, W, N, Ueq, σ)
28 for each node i ∈ {1..57} do
29if PBFT_Vote(Block, i) == true then L.append(Block)
30 end for
31 if ConsensusRatio(L) ≥ 2/3 then Π_confirmed ← Π*
32 else goto 1   // Consensus failed, reorder
═══ Phase Three: Edge-side Execution and Dynamic Feedback ═══
33 for each workstation i ∈ {1..57} do
34   Execute(Π_confirmed, i)
37   if Deviation(Status_i, Π_confirmed) > ε then
38     SmartContract.TriggerReschedule()
39     goto 1
40   end if
41 end for
42 return Π_confirmed, L
```

Lines 12–18 of the pseudocode correspond to forward coupling (chain-end verification driving algorithm elimination), line 22 corresponds to reverse coupling (chain-end state S driving adaptive repair), and lines 31–32 and 37–39 correspond to the closed-loop triggered by consensus failure and dynamic perturbations.

4. Experimental validation

The experimental environment is shown in Table 4. The data is derived from 13 complete production scheduling records for PCTC sections from the China Merchants Group's laser sheet metal workshop between July and October 2025, covering 57 workstations, 7 manufacturing stages, 3 work teams and 12 types of equipment.
Experimental environment and data sources:

Table 4. Experimental environment and data sources

Project	Specifications
CPU	Intel Xeon Gold 6248R (48-core, 2.5GHz)
Memory	64GB DDR4
OS	Ubuntu 20.04 LTS
Development environment	Python 3.9 + DEAP genetic algorithm framework + Solidity smart contracts
Blockchain framework	Hyperledger Fabric v2.4 (Consortium blockchain)
Consensus Algorithm	PBFT (tolerance of f ≤ ⌊(n-1)/3⌋, i.e. a 57-node network tolerating the failure of 18 nodes)
Network Topology	Cloud server (master node) + 57 edge nodes (workshop workstations, connected via Gigabit LAN)

Data source: Actual production data from a company's laser sheet metal workshop, covering complete production scheduling records for 13 PCTC car carrier sections from July to October 2025. The workshop comprises 57 workstations and 7 manufacturing stages (material preparation → pre-treatment → cutting → machining → sub-assembly → intermediate assembly → section construction), involving 3 work teams and 12 types of equipment.

4.1. Comparative performance experiments of algorithms

Each algorithm was run independently 10 times, recording the objective function value F = w1·Tmax + w2·Energy + w3·σ for each iteration (weights w1 = w2 = w3 = 1/3), the convergence statistics for the algorithm are shown in Table 5:

Table 5. Algorithm convergence statistics (mean ± standard deviation, 10 runs)

Algorithm	Average number of iterations	Optimum value	Worst value	Average	Standard deviation	Convergence criterion
GA	61.7	68.0	58.0	60.5	±3.2	Change in optimum value over 50 consecutive generations < 0.01
PSO	40.0	59.0	55.0	57.1	±1.8	Change in global optimum over 30 consecutive generations < 0.01
GASA	12.8	57.0	57.0	57.0	±0.0	Change in optimum value over 20 consecutive generations < 0.001
This algorithm	3.9	55.1	55.0	55.1	±0.05	Change in optimum value over 3 consecutive generations < 0.001

The results of 10 runs of this algorithm were highly consistent (55.0–55.2), with a standard deviation of just 0.05, indicating that the algorithm is extremely stable. The convergence criterion was defined as 'no change in the non-dominated sorting frontier of NSGA-II for three consecutive generations'; as the smart contract's hard validation rapidly eliminates invalid solutions, the quality of the population improves very quickly.

GA: Rapid decline in the first 20 generations, followed by slow convergence between generations 20 and 60, with multiple fluctuations (escaping from local optima); PSO: Rapid convergence to around 57 in the first 10 generations, followed by oscillation between 55 and 59, unable to break through 55; GASA: Converged to 57 in approximately 8 generations, but was unable to further optimise multi-objective trade-offs due to the limitations of a single-objective framework; Our algorithm: As early as the first generation, individuals passed all five types of constraint validation (as the initial population partially satisfied the process route); the NSGA-II frontier stabilised at 55.1 in the third generation, with no further improvement from the fourth generation onwards, leading to automatic termination.

In this paper, three scheduling optimization objectives (makespan, total energy consumption, equipment load variance) are weighted equally with \(w_1=w_2=w_3=1/3\) for comprehensive evaluation of ship laser sheet metal workshop scheduling performance. Equal weight allocation is adopted because the shipyard simultaneously pursues delivery timeliness, energy cost control and balanced equipment utilization in actual production, and there is no mandatory priority for a single objective. To further verify the stability of the proposed improved genetic algorithm under different weight preferences, we supplement 2 groups of unequal weight combinations for sensitivity comparison, and the statistical results after 10 repeated operations are listed in Table 6.

As shown in Table 6, even when the weight distribution of each objective is adjusted significantly, the average optimal objective value of the proposed algorithm only fluctuates within a tiny range of 55.0–55.4, and the standard deviation remains below 0.1. This indicates that the multi-strategy optimization mechanism and on-chain hard constraint filtering can effectively balance multi-objective trade-offs, and the algorithm maintains stable optimization performance under different production preference weights, which verifies the rationality of the equal-weight setting adopted in this paper.

Table 6. Sensitivity test results under different objective weight combinations (10 runs average)

Weight combination (w1,w2,w3)	**Average optimal objective value**	**Standard deviation**	**Main preference**
(1/3,1/3,1/3) (Original setting)	55.1	±0.05	Balanced three objectives
(0.5,0.25,0.25)	55.3	±0.08	Priority to shorten production cycle
(0.25,0.25,0.5)	55.0	±0.07	Priority to balance equipment load

We conduct ablation experiments to quantify the independent contribution of each core module, as shown in Table 7. Four ablation groups are constructed by removing the smart contract verification, PSO fusion operator, adaptive repair operator and PBFT consensus separately. The results show that deleting the smart contract module causes the maximum performance degradation (+2.1), since it filters invalid scheduling solutions in advance. The PSO fusion operator and adaptive repair bring obvious optimization gains of 1.3 and 0.7 respectively, by enhancing global search and

coping with dynamic workshop disturbances. The PBFT consensus slightly improves the objective value by 0.3, and its main value lies in realizing tamper-proof and consistent data synchronization across 57 distributed workstations. All modules are indispensable to the overall performance of the CECS framework.

Table 7. Ablation study results (average optimal objective value over 10 independent runs)

Group ID	Module configuration	Average optimal value	Degradation compared with Full CECS	Core removed module
1	Base (No smart contract verification)	57.2	+2.1	Blockchain smart contract hard filtering
2	Base+Contract (No PSO fusion)	56.4	+1.3	PSO velocity-position update operator
3	Base+Contract+PSO (No adaptive repair)	55.8	+0.7	Dynamic adaptive repair operator
4	Base+Contract+PSO+Repair (No PBFT consensus)	55.4	+0.3	Consortium chain PBFT multi-node consensus
5	Full CECS (All modules integrated)	55.1	0	Complete proposed framework

4.2. Blockchain performance testing

4.2.1. Single Smart Contract Verification Duration

Hard verification of the five types of constraints was performed on a single individual (encoding comprising process segment, equipment segment and workgroup segment), the time taken for the five types of constraint verification in smart contracts is shown in Table 8:

Table 8. Verification duration for the five types of smart contract constraints (single individual)

Constraint Type	Verification Content	Average Duration
Process Route Constraints	Query W-matrix, verify completion status of immediately preceding process	0.4ms
Workgroup Exclusivity Constraints	Query workgroup time slot occupancy table	0.3ms
Equipment Constraints	Query equipment time slot occupancy table	0.3ms
Processing sequence constraint	Fuzzy variable yij evaluation + equipment sequence verification	0.5ms
Overrun determination	Comparison of Tend vs Tdeadline	0.1ms
Total		1.6ms per item

100 individuals/generation × 1.6 ms = 0.16 s/generation; 100 generations total 16 s, accounting for 64% of the total computation time of 25 s. The remaining time is spent on NSGA-II sorting, crossover, and adaptive repair.

4.2.2. Experimental measurement of PBFT consensus latency

The performance of the PBFT consensus is shown in Table 9:

Table 9. PBFT Consensus performance (average of 100 operations)

Number of nodes	Consensus latency (p50/p99)	Throughput (TPS)
10	12ms / 45ms	1850
30	38ms / 120ms	920
57 (actual deployment)	68ms / 210ms	510
100 (extrapolated)	180ms / 560ms	210

We decompose the total PBFT consensus delay into three phases: message broadcast, node signature verification and ledger synchronization. For the 57-workstation testbed, message broadcast accounts for 42% of the p50 latency, signature verification accounts for 38%, and ledger synchronization takes up 20%. The rising delay with more nodes mainly originates from the growing overhead of message transmission and signature validation, whereas ledger synchronization introduces stable minor time cost. This staged breakdown helps interpret the scalability limitation of PBFT recorded in Table 7.

With 57 nodes, the consensus delay is 68 ms (p50), which has a negligible impact on the total scheduling time of 25 s (<0.3%). The breakdown of the '3-second increase' is as follows Table 10:

Table 10. Breakdown of the 3-second increase in scheduling time

Stage	Duration	Percentage
Smart contract hard validation	1.6s	53%
PBFT consensus (57 nodes)	0.07s	2%
Data on-chain notarisation (4 types of data bundled)	0.8s	27%
PBFT secondary confirmation (execution status feedback)	0.53s	18%
Total	3.0s	100%

Data source: Based on the average of 100 consensus operations performed on Hyperledger Fabric v2.4 in the same hardware environment.

4.3. Engineering application results

In the practical application at the 57 workstations of the China Merchants Cruise Laser Thin Plate Workshop, the production scheduling algorithm takes approximately 25 seconds to execute. Statistics for the 13 sections of the PCTC roll-on/roll-off vehicle show that the planned construction cycle was 7 days; however, 9 sections were completed within 7 days, 2 within 6 days, and 2 within 8 days, resulting in an estimated operational progress deviation of:

The deviation = $(2\times|6-7| + 2\times|8-7|)/13/7 = 914 \approx 4.39\%$

Table 11 shows a comparison of the results from specific engineering applications:

Table 11. Comparison of engineering application results

Metric	Before blockchain implementation	After blockchain implementation	Improvement
Production scheduling time	~25s	~28s	+3s
Tampering detection rate	0%	100%	+100%
Data consistency	96.2%	99.8%	+3.6%
Job progress deviation	4.39%	3.82%	-0.57%

With the implementation of blockchain, the job progress deviation was further reduced to 3.82%, while production scheduling time increased by only about 3 seconds, achieving dual improvements in scheduling efficiency and data reliability.

5. Summary

This paper addresses the prominent issues in the dynamic scheduling of digital shipyards, including insufficient data reliability, low transparency in multi-party collaboration, and the susceptibility of scheduling instructions to tampering. Building upon the existing cloud-edge collaborative scheduling architecture, it innovatively introduces blockchain technology to construct a three-tier 'cloud-edge-chain' collaborative scheduling architecture. Furthermore, it proposes a multi-objective genetic algorithm enhanced by multi-strategy improvements based on cloud-edge collaboration, thereby providing dual safeguards for scheduling efficiency and data reliability.

To address the three major challenges in dynamic scheduling for digital shipyards—insufficient data reliability, low transparency in multi-party collaboration, and delayed constraint verification responses—this paper proposes a 'cloud-edge-chain' three-tier collaborative scheduling architecture and a multi-strategy-enhanced multi-objective genetic algorithm. Experiments demonstrate that the convergence speed of the improved algorithm is approximately 94% faster than that of traditional GA; in engineering applications, the data tampering detection rate reaches 100%, data consistency reaches 99.8%, and task progress deviation is reduced to 3.82%.

Limitations: 1) Current consortium blockchain consensus mechanisms may face performance bottlenecks when the number of nodes exceeds 100; 2) Real-time rescheduling strategies under dynamic disturbance scenarios require further optimisation; 3) The mechanism for cross-workshop blockchain data interoperability has yet to be validated.

Future Prospects: We will explore lightweight consensus algorithms to support larger-scale workshops, investigate online learning-based rescheduling methods for multi-disturbance scenarios, and promote cross-workshop blockchain interconnection to provide more comprehensive technical support for intelligent shipbuilding.

References

[1] Zhang H, Li X, Wang Y. A genetic algorithm-based scheduling method for ship block construction considering process and resource constraints. Journal of Ship Production and Design, 2019, 35(3): 187-196.

[2] Liu W, Chen S, Zhou M. Particle swarm optimization for ship block scheduling: A case study in a Chinese shipyard. Ocean Engineering, 2020, 212: 107623.

[3] Wang J, Huang L, Zhang Q. A hybrid genetic algorithm combining simulated annealing for ship segment construction scheduling. Computers & Industrial Engineering, 2021, 156: 107254.

[4] Li Z, Xu X, Wang L. Cloud-edge collaborative scheduling architecture for digital workshop in discrete manufacturing. Journal of Manufacturing Systems, 2021, 60: 491-503.

[5] Kolhar M, Abu-Salih B, Al-Amri J, et al. Blockchain for industrial internet of things: A comprehensive survey. IEEE Access, 2021, 9: 140785-140807.

[6] Wang S, Ouyang L, Yuan Y, et al. Blockchain-enabled smart contract for production scheduling constraint verification in intelligent manufacturing. Robotics and Computer-Integrated Manufacturing, 2022, 73: 102245.
[7] Yang Z, Li K, Zhou M. An adaptive genetic algorithm for dynamic job shop scheduling with real-time rescheduling. Journal of Intelligent Manufacturing, 2022, 33(5): 1487-1503.
[8] Kennedy J, Eberhart R. Particle swarm optimization: a historical perspective and recent advances// Proceedings of IEEE International Conference on Neural Networks. Virtual, 2020: 1-8.
[9] Wang H, Zhang Y, Li X. A hybrid simulated annealing and genetic algorithm for multi-objective flexible job shop scheduling. Computers & Industrial Engineering, 2021, 156: 107268.
[10] Zhang C, Li S, Wang R. Practical Byzantine fault tolerance in heterogeneous edge-cloud environments// Proceedings of OSDI '22. Carlsbad, USA: USENIX, 2022: 385-401.
[11] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.
[12] Gao L, Wang X, Xu X, et al. Cloud-edge-end collaboration for real-time scheduling in smart manufacturing. Journal of Intelligent Manufacturing, 2022, 33(5): 1439-1455.
[13] Tao F, Zhang H, Liu A, et al. Digital twin-driven product design, manufacturing and service with big data. The International Journal of Advanced Manufacturing Technology, 2018, 94(9-12): 3563-3576.
[14] Saberi S, Kouhizadeh M, Sarkis J, et al. Blockchain technology and its relationships to sustainable supply chain management. International Journal of Production Research, 2019, 57(7): 2117-2135.
[15] Zhang C, Xu Y, Zhang L, et al. Dynamic scheduling for shipbuilding workshop based on improved genetic algorithm with multi-strategy. Applied Soft Computing, 2023, 136: 110052.
[16] Xu X, Liu C, Xu L D. Blockchain-enabled trusted data sharing in industrial IoT. IEEE Transactions on Industrial Informatics, 2021, 17(11): 7643-7653.
[17] Ferrag M A, Derdour M, Mukherjee M, et al. Blockchain technologies for the internet of things: Research issues and challenges. IEEE Internet of Things Journal, 2020, 7(7): 6284-6300.
[18] Wang L, Liu S, Liu R, et al. Dynamic scheduling of ship block erection based on hybrid genetic algorithm. Ocean Engineering, 2023, 268: 113312.
[19] Liu Y, Wang L, Li Z. Edge computing-assisted real-time scheduling for digital twin workshop. Journal of Manufacturing Systems, 2022, 65: 1-14.
[20] Dai J, Wang H, Lin H, et al. Multi-objective optimization for ship production scheduling using improved NSGA-III. Computers & Operations Research, 2023, 150: 106063.
[21] Chen M, Liu J, Wang X. Blockchain-based smart contract for dynamic job shop scheduling. Expert Systems with Applications, 2023, 215: 119312.
[22] Kshetri N. Blockchain and sustainable supply chain management in developing countries. International Journal of Information Management, 2022, 67: 102552.