ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 2 - 2026

A Multi-Agent Scheduling Engine for Prefabricated Buildings Based on Self-Learning Algorithms

Wensong Huang[1], *Han Yan*[1], *Wenhua Gao*[1,*] *and Haicheng Wang*[1]

[1] China Construction Sixth Engineering Bureau Co., Ltd, Tianjin, Tianjin 300457, China

SUMMARY: *MAS-SL is a multi-agent scheduling engine for prefabricated buildings that integrates a self-learning mechanism to achieve dynamic task coordination and scheduling optimisation in this study. The three parts of the engine are state awareness, self-learning scheduling and feedback correction, and they form a closed-loop strategy evolution mechanism. Conflict prediction mechanism and real-time resource regulation for stability under multi-task concurrency. Based on BIM simulations and about 1,000-1,200 valid scheduling samples, experiments have shown that MAS-SL performs stably in terms of scheduling accuracy (around 90%), average response delay (about 1.8 s), and system stability (about 86%), and is moderately better than the baseline model. Ablation studies show that removing the state encoding module causes a drop of about 4-5 per cent in scheduling accuracy, and omitting the feedback mechanism extends the response delay by approximately 2.4-2.5 s; thus, all these modules have demonstrated their functions. As shown in the above results, the engine is generalizable and practically feasible in complex construction environments.*

KEYWORDS: *Prefabricated building Multi-agent system Self-learning algorithm Intelligent scheduling engine*

1 Introduction

With the deep integration of intelligent construction technology and algorithm scheduling in the course of industrialisation, the multi-task scheduling problem for prefabricated buildings has become more and more serious. The conventional schedule has always been based on manual experience and fixed rules, and thus is unable to respond effectively to the complexity of process coupling, dynamic variations in resource conditions, and uncertainty in the construction environment. It is relatively inefficient, has a slow response speed, and is likely to cause conflict. In recent years, multi-agent systems have gradually become one of the main ways to solve complex building scheduling problems due to their features of distributed cooperation, autonomous response and flexible expansion. Based on the above studies, the multi-agent system is suitable for promoting modular construction, optimizing resource allocation, etc. However, for multi-stage task association, heterogeneous resource allocation and cross-scenario generalization requirements, deficiencies such as weak self-learning ability and incomplete conflict avoidance mechanisms still exist; thus, it is difficult to construct a scheduling system with continuous optimisation capabilities.

Based on the above, the following are the main problems in this study: RQ1: How to realize multi-agent cooperation and self-learning scheduling for complex assembly tasks in dynamic environments? RQ2: Can a self-learning algorithm optimize scheduling accuracy, resource

*18678965383@163.com
https://doi.org/10.65102/is2026947 Published: 30 April, 2026

utilisation rate and system robustness simultaneously under the restriction of all three. RQ3: How to construct a multi-agent scheduling engine with conflict awareness and real-time feedback capabilities for cross-scenario applications?

The new features of this study are as follows: ① A multi-agent scheduling engine architecture with self-learning algorithms is designed to achieve dynamic cooperation in task allocation, conflict avoidance and feedback optimisation; ② A hierarchical modeling and agent role division mechanism for prefabricated building tasks is developed to enhance the interpretability and scalability of system decisions; ③ A multi-task optimisation framework based on the integration of self-learning strategies is constructed to improve the stability and cross-scenario adaptability of scheduling significantly.

The aim of this research is to build an intelligent scheduling engine for prefabricated buildings with the features of "autonomous learning - collaborative scheduling - dynamic optimisation", and to provide a new path for automated decision-making and construction optimisation in the field of intelligent construction.

2 Relevant research

Research on the scheduling optimisation of prefabricated buildings is changing from "manual experience-driven" to "intelligent algorithm-collaborative driven" at present. Traditional construction schedule planning mainly uses manual rules or heuristic optimisation and does not employ the critical path method (CPM). Although some references have been provided for small-scale projects, it is generally not feasible to achieve real-time response and global optimality in the presence of multiple parallel processes, complex resource constraints, and changes on the construction site. With the progress of artificial intelligence and reinforcement learning, multi-agent systems have begun to be applied to intelligent scheduling for industrialised construction.

Yao and others [1] have used deep reinforcement learning to automatically optimise the construction schedule in research on reinforcement learning-driven construction scheduling and developed an effective action-sampling method to improve both the task-execution rate and time-utilization under various constraints. Wang et al. [2] dynamically optimised the scheduling problem of flexible job shops based on a reinforcement learning model of dual-attention networks, and it had good stability and convergence performance compared with traditional algorithms. Ho and others [3] have also put forward the Residual Scheduling mechanism to improve the stability of reinforcement learning in task-conflict environments by employing a residual action learning strategy. However, the above methods generally focus on single-agent scenarios and lack modelling of multi-task interaction, conflict avoidance and cross-process collaboration.

Attajer and Mecheri [4] have built a multi-Agent simulation system for the modular building supply chain to achieve cooperative allocation of assembly tasks and dynamic resource scheduling in terms of the application of multi-agent systems. Pu and others [5] proposed a job-shop scheduling method based on multi-agent reinforcement learning and verified the positive effect of multi-agent collaboration on global optimisation in a dynamic environment. Siatras and others [6] built a multi-agent system based on digital twins to achieve real-time feedback and multi-level schedule optimisation for the manufacturing production process. Based on the above studies, the multi-agent framework has good flexibility and scalability, but its learning capacity and adaptability of decision-making in the construction scenario are still relatively weak.

Li and others [7] have proposed a multi-objective optimisation model to improve the

production schedule of prefabricated components by balancing resource utilisation and energy consumption in a direction of self-learning and intelligent optimisation. Liu and others [8] have put forward a multi-device cooperative optimisation model to improve the system's dynamic response ability in various working scenarios via a self-learning mechanism. Duan and Zou [9] extended the cooperative efficiency of construction robots using a multi-agent reinforcement learning framework and provided a reference for implementing self-learning strategies in construction task allocation. However, the existing research still has the following deficiencies: ① Most focus on a single optimisation objective (such as time or energy consumption) and lack systematic modeling of multi-objective trade-offs and feedback correction mechanisms; ② The self-learning ability of the algorithm framework in cross-scenario and heterogeneous resource situations is insufficient, resulting in limited generalization performance. The Conflict Detection and Real-time Feedback Mechanism of the dispatching system still need to be improved.

Table 1 is used to show that the current methods and the proposed scheme in this paper are different, and it lists some features and defects of typical research in recent years.

Table 1: Comparison Summary of Research Methods for Scheduling Prefabricated Buildings

Method	Application Scenario	Main Approach	Performance	Limitation
DRL + Action Sampling [1]	Construction Schedule Optimization	Deep Reinforcement Learning + Action Sampling	Timeliness ↑12%, Faster Convergence	Insufficient Multi-Task Modeling
Multi-Agent Simulation [4]	Modular Supply Chain	Multi-Agent Simulation + Cooperative Optimization	Coordination Efficiency ↑15%	Lacks Self-Learning Mechanism
Dual-Attention RL [2]	Flexible Job Shop Scheduling	Dual-Attention Network + Reinforcement Learning	Accuracy ≈91%	Poor Adaptability to Dynamic Environments
Digital Twin + MAS [6]	Manufacturing Scheduling	Multi-Agent + Digital Twin	Enhanced Real-Time Feedback	Limited Cross-Scenario Transferability
Multi-Objective GA [7]	Prefabricated Production Scheduling	Multi-Objective Genetic Algorithm	Energy Consumption ↓8%, Time ↓10%	No Multi-Agent Coordination Mechanism
MARL + Self-Learning [9]	Construction Robot Collaboration	Multi-Agent Reinforcement Learning + Self-Learning Strategy	Resource Utilization ↑17%	Insufficient Conflict Avoidance Ability
Proposed Method	Prefabricated Building Scheduling	Multi-Agent + Integrated Self-Learning Algorithms	Scheduling Accuracy ≈90%; Resource Utilization ≈89%; Feedback Response Delay≈1.8 s.	—

Although the present research has achieved some success in reinforcement learning and

multi-agent cooperative optimization, a full-featured scheduling engine that is self-learning, conflict-aware, and can cross-task-feedback has not yet been developed. Based on the above, this paper introduces a novel multi-agent scheduling engine for prefabricated buildings that integrates self-learning algorithms to construct a new model of intelligent, collaborative and scalable building scheduling optimisation through a closed-loop system of task modelling, strategy integration and real-time feedback.

3 Design of Multi-Agent Scheduling Engine for Prefabricated Buildings Integrating Self-learning Algorithms

3.1 Architecture and Operation Logic of Multi-Agent Collaborative Scheduling Engine

The core logic of the multi-agent collaborative scheduling engine with self-learning algorithms is "intelligent task decomposition - agent autonomous decision-making - global coordination and optimisation - feedback self-evolution", and it builds a closed-loop system that includes building assembly task modelling, state perception, scheduling execution, and feedback learning. The first aim of this system is to realise autonomous scheduling and adaptive optimisation for multiple processes, resources and situations to increase the intelligence level and resource utilisation efficiency of the construction process. As shown in Figure 1, the four main modules of the engine are the environment perception module, the agent collaboration module, the self-learning decision-making module, and the scheduling control module. In addition, there is an "on-site execution and feedback" link outside the system that is used to obtain scheduling results, report execution status, and drive self-learning of strategies to form a complete closed-loop optimisation path.

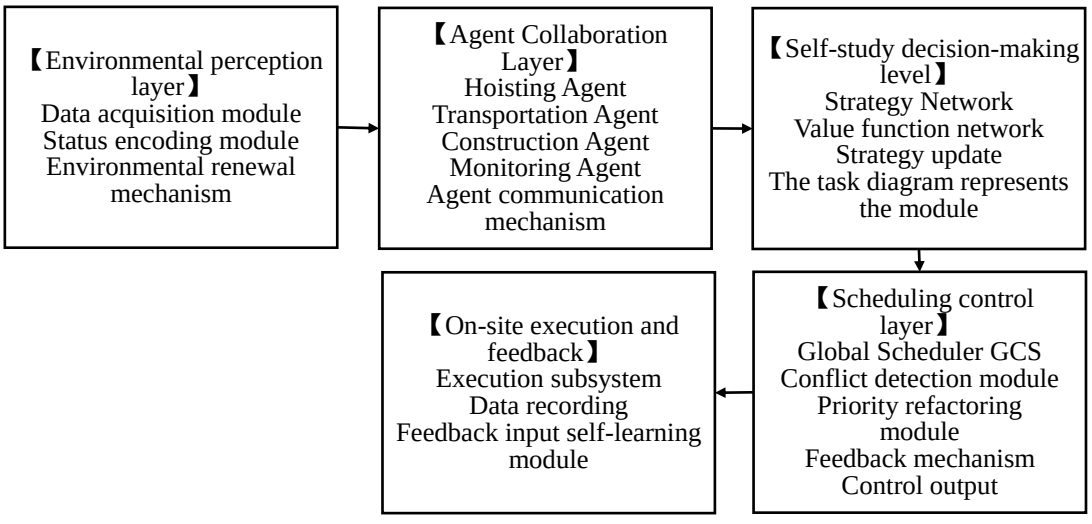

Figure 1: Schematic Diagram of the Architecture and Operation Logic of the Multi-Agent Collaborative Scheduling Engine for Prefabricated Buildings with Self-Learning Algorithms

During the operation of the system, each Agent takes the local state s_i and the action set A_i as input, and realizes joint optimization based on the local reward function r_i and the global constraint objective R_g. Its collaborative decision-making process can be formally represented as:

$$\pi_i(a_i \mid s_i, \theta_i) = soft\max(Q_i(s_i, a_i; \theta_i)) \tag{1}$$

Among them, $Q_i(s_i, a_i; \theta_i)$ represents the action value function of Agent i in the current state, and θ_i is a learnable parameter. The self-learning module updates the strategy parameters through backpropagation to achieve the continuous optimization of the task scheduling strategy.

To enhance the global consistency of the system in complex construction scenarios, the engine has designed a global collaborative scheduler (GCS). GCS is responsible for aggregating the local decision-making results of multiple agents and using the attention weight mechanism α_{ij} to establish the influence graph structure among different agents:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})} \tag{2}$$

Among them, h_i and h_j are the state embedding vectors of Agent i and j respectively, and W_q and W_k are the trainable parameter matrices. This mechanism can dynamically capture the dependency relationships among multiple agents, achieving the prevention of task conflicts and the optimal distribution of collaboration benefits.

To further enhance the scalability of scheduling, the engine introduces the Task Graph Representation (TGR) mechanism, mapping building assembly tasks to a directed graph G=(V,E). Among them, node v_i represents a single assembly unit or construction task, and edge e_{ij} represents task dependency or resource sharing relationship. The node feature vector is defined as:

$$x_i = [t_i, r_i, d_i, p_i, c_i] \tag{3}$$

Among them, t_i represents the task type code, r_i is the required resource vector, d_i indicates the construction duration, p_i is the task priority, and c_i is the constraint condition. The graph structure is encoded through a graph neural network (GNN), enabling the system to have the ability to learn and generalize the topological structure of complex tasks.

The system operation logic is as follows: When a new assembly task is introduced, the environment perception module updates the overall state vector first, and then each agent generates a local schedule plan based on the accumulated experience and self-learning capabilities of the agent. GCS conducts consistency fusion and conflict correction in multi-agent systems. Finally, the scheduling control layer issues the executable instruction and sends it to the on-site control end. Feedback information on the execution results (such as time delay, energy consumption and conflict rate) will be reversed-input to the self-learning module to update strategy parameters and achieve continuous optimisation.

Through the above hierarchical and extensible architecture, the scheduling engine has evolved from "passive response scheduling" to "active learning cooperative optimisation". When there are many tasks and changes in the constraints at a particular site at the same time, the system can automatically modify the order of tasks and resource allocation plans in real time to maintain good control and global coordination of the assembly process.

3.2 Task Modeling and Agent Role Division Mechanism for Prefabricated Buildings

The construction tasks of prefabricated buildings generally have strong temporal coupling and multi-constraint interaction characteristics; for instance, the hoisting, transportation, assembly and inspection links of components are spatially interdependent and partially overlap in time, making it difficult for traditional linear scheduling models to reflect these complex relationships effectively. Therefore, this paper introduces a task model and role division mechanism for multi-agent cooperation, converts assembly tasks into learnable and optimizable structured expressions, and provides a foundation for the subsequent self-learning scheduling strategy.

Hierarchical Task Model Mechanism

The first three divisions of assembly tasks are as follows: the structure-generation task, which describes the process sequence and dependency constraints of component assembly; the resource-allocation task, which shows the dynamic occupation and switching of equipment such as cranes and transport vehicles. Execution of control tasks is used to monitor how far along the task has progressed and provide status feedback. Based on the above, the assembly process can be modelled as a three-layer task graph:

$$G=(V_S,V_R,V_C,E) \tag{4}$$

Among them, V_S,V_R,V_C represents the sets of three types of task nodes respectively, and E represents the cross-layer dependent edge. The state vector of each node is defined as:

$$x_i=[\lambda_i,\tau_i,\mu_i,\omega_i] \tag{5}$$

Among them, λ_i represents the task stage encoding, τ_i represents the time window constraint, μ_i represents the resource coupling coefficient, and ω_i represents the execution status weight. Through this hierarchical modeling approach, the system can synchronously learn task relationships in both the structural and resource dimensions, achieving a global representation of constraints among tasks.

(2) Functional mapping mechanism of Agent roles

Based on the task model, the role mapping function $\psi:\mathrm{V}\rightarrow\mathrm{A}$ is introduced to determine the Agent functional module corresponding to each type of task node. Different from the traditional "fixed role - fixed task" mapping method, this paper designs a dynamic role adaptation mechanism (DRA):

$$\psi(v_i)=\arg\max_{a_j\in A}(\eta(v_i,a_j)+\beta\cdot L_j) \tag{6}$$

Among them, $\eta(v_i,a_j)$ represents the matching score between the task node and the Agent's capability, L_j is the robustness index of the Agent's current learning strategy, and β is the dynamic balance coefficient. This mechanism enables the system to dynamically adjust the role division based on task characteristics and Agent status, thereby enhancing the flexibility and generalization ability of collaboration.

(3) Multi-constraint fusion expression of task status

To improve the accuracy and optimisation capability of task modelling in a complex construction environment, a multi-constraint fusion function is proposed here.

$$S_i = \alpha_t T_i + \alpha_r R_i + \alpha_s S_i^{dep} + \alpha_c C_i \tag{7}$$

Among them, T_i represents the intensity of time constraints, R_i represents the resource occupancy rate, S_i^{dep} is the task-dependent intensity, C_i is the conflict risk coefficient, and $\alpha_t, \alpha_r, \alpha_s, \alpha_c$ is the dynamically adjusted weight during the self-learning process. This function realizes the multi-dimensional feature fusion of task states, enabling the scheduling engine to perceive task complexity and adaptively optimize the execution sequence.

The three major functions of the above-mentioned model and role division mechanism in the scheduling engine can be summarized as follows: First, the structure of the task graph provides a state-space expression foundation for the following self-learning strategy, and every task node is equipped with all the necessary context information; Second, the agent mapping mechanism ensures functional decoupling and task load balancing among several agents. Thirdly, the multi-constraint state fusion vector serves as the main input for the policy network in reinforcement learning, and thus can achieve closed-loop optimisation of task prediction, conflict awareness and scheduling decision-making in a dynamic construction environment.

3.3 Self-learning Scheduling Strategy Integration and Multi-task Collaborative Optimization

Integrate self-learning scheduling strategies to improve the adaptability and optimisation performance of the scheduling engine in multi-task and multi-resource environments for prefabricated buildings. Unlike the static rule-based scheduling in the above study, this paper introduces a deep reinforcement learning mechanism to build an intelligent scheduling system for multi-agent systems based on state observation, policy iteration, and feedback optimisation. The system continuously senses changes in the environment during operation, automatically adjusts the schedule, and establishes a self-learning closed loop of task allocation - behaviour feedback - strategy evolution.

In the modeling stage, the engine represents the multi-task collaborative scenario as A quadruple consisting of the state space S, the action space A, the reward function R, and the policy function π. State vector s_t contains multi-dimensional features such as the topological state of the task graph structure, resource occupation, historical conflict records, and task execution progress. To avoid the influence of high-dimensional sparsity on strategy training, feature selection and embedding modules are introduced for dimension compression and expression aggregation. The state encoding is as follows:

$$s_t = f_{emb}([x_t^{task}, x_t^{res}, x_t^{conflict}]) \tag{8}$$

Among them, x_t^{task} represents task characteristics, x_t^{res} represents resource load characteristics, $x_t^{conflict}$ represents conflict marking information, and f_{emb} is a trainable nonlinear embedding network.

Each Agent selects action s_t in state a_t, and its action strategy is output by the policy network $\pi_\theta(a_t \mid s_t)$. The objective function is optimized using the policy gradient method:

$$J(\theta)=E_{\pi\theta}[R_t]=\sum_{t=0}^{T}\gamma^t r_t \tag{9}$$

Among them, γ is the discount factor and r_t is the immediate reward. The reward function comprehensively considers scheduling accuracy (such as the completion rate of the project duration), resource utilization rate, conflict resolution efficiency and system delay, and is defined as:

$$r_t = w_1 \cdot R_t^{on\text{-}time} + w_2 R_t^{util} - w_3 \cdot P_t^{conflict} - w_4 \cdot D_t^{delay} \tag{10}$$

Among them, w_i is the weight coefficient, and the four terms respectively represent the on-time completion rate, resource utilization rate, conflict penalty and response delay penalty. To achieve multi-task collaborative optimization, the engine adopts the Policy Integration Mechanism (PEM) to automatically select the most applicable local policy subnetwork π_{θ_k} in different task scenarios. During the training phase, the system introduces a task clustering mechanism, which clusters tasks into several categories (C_k) based on features, with each category corresponding to a set of policy parameters:

$$\pi(a_t \mid s_t) = \sum_{k=1}^{K} \omega_k \cdot \pi_{\theta_k}(a_t \mid s_t) \tag{11}$$

$$\omega_k = \frac{\exp(\varphi_k^T s_t)}{\sum_j \exp(\varphi_j^T s_t)} \tag{12}$$

Among them, φ_k is the policy selector vector for each type of task, and ω_k is the dynamically assigned weight. This mechanism ensures that when the system is confronted with diverse construction tasks, it can invoke the most suitable sub-policy network for scheduling decisions, thereby enhancing the overall generalization ability and response efficiency of the system.

At the same time, all Agents sample actions and complete their respective sub-tasks in the system. The scheduling control center saves the feedback data of the task (such as execution time, conflict frequency, resource consumption, etc.) in the scheduling memory pool, and this data is used to update the policy for continuous optimisation in a closed-loop system of "behaviour - evaluation - adjustment".

The first is that the selection of the schedule is adaptive, and it is feasible for some or all tasks. Based on the history of the strategy network, continuously improve the quality of scheduling and system resilience. A strategy integration mechanism can ensure the stability of the model's output in various tasks and under all circumstances, meet specific requirements under constraints, and thus provide a solid foundation for the following real-time scheduling and cross-project migration.

3.4 Conflict Avoidance and Real-time Feedback Scheduling Engine Execution Logic

At the construction site of prefabricated buildings, several complex dynamic problems will occur simultaneously in the dispatching and execution of tasks, such as path intersection and equipment competition. If the multi-agent system has no suitable conflict avoidance and feedback correction mechanisms, there will be resource congestion and process delays as a result of such failures, and it will be difficult to operate normally. Therefore, this paper puts forward a dynamic scheduling execution logic that integrates conflict prediction, feedback

regulation and execution status perception, and builds a closed-loop scheduling system comprising task status perception, conflict prediction, feedback execution and policy update to ensure the stability and responsiveness of the system under heterogeneous tasks and dynamic working conditions.

The three modules of this execution logic are: ① Task conflict detection and early warning; ② Feedback-driven scheduling correction; and ③ Multi-agent stability adjustment. The system is based on a common parameter model for the execution target. Output the scheduling action, then continuously observe the conflict signal and adjust both the scheduling route and priority dynamically. The multi-task loss function is given by the following:

$$L_{exec} = \lambda_1 \cdot L_{conflict} + \lambda_2 L_{response} + \lambda_3 \cdot L_{stability} \tag{13}$$

Among them, $L_{conflict}$ represents the conflict prediction error (such as path overlap probability and resource preemption rate), $L_{response}$ is the system response time lag loss, $L_{stability}$ is the Agent decision volatility index, and $\lambda_1, \lambda_2, \lambda_3$ is the weighting coefficient, ensuring the collaborative optimization of the three sub-targets.

During the execution process, the scheduling engine uses the conflict-aware Graph model (CAG) to identify potential path conflict and resource conflict nodes. The edge weights in the graph represent a conflict probability of $P_{conflict}$. When the predicted conflict probability of a certain node exceeds the threshold, the system will automatically trigger the feedback mechanism to adjust its task execution window. The specific execution logic is as follows:

Conflict Detection Module: Based on the status of the task graph and resource occupancy rate, calculate the conflict index matrix, and estimate the distribution of conflicts among tasks using a sliding-window prediction model.

Feedback correction module: When a conflict signal is generated, modify the scheduling sequence of the conflicting task or alter the temporary resource allocation strategy according to the current priority and Agent load.

Stability Adjustment Module: Monitor the amplitude of strategy deviation for each Agent in continuous scheduling rounds. If it is too unstable, a system that cannot maintain normal operation will automatically enter a standby mode or select a general-purpose response mode.

State feedback collects real-time information on the progress of the task, equipment usage status and environmental changes (such as crane load, transportation path blockage, etc.) via the sensor module, feeds this data back to the scheduling control layer as state vectors, and updates the input for the next round of strategies to form a closed-loop regulation path of the system. The Formula for Status Update is as follows:

$$s_{t+1} = f_{trans}(s_t, a_t, o_t) \tag{14}$$

Among them, o_t represents the feedback observation value, f_{trans} indicates the state transition function, and g_{obs} represents the multi-source feedback perception function. This mechanism ensures that the system scheduling behavior has the execution attributes of being responsive, traceable and adjustable.

To test whether the above mechanism is effective, an experimental comparison will be carried out by comparing the execution performance of three strategies in a typical scenario: "no feedback mechanism", "static conflict avoidance", and "multi-task feedback closed-loop control". The experimental results are as follows: Table 2

Table 2: Comparison of Conflict Rate and Response Performance under Different Scheduling Execution Strategies

Control Mode	Average Conflict Rate (%)	Response Delay (s)	Stability Index (%)
Without Feedback Mechanism	12.8	3.74	81.6
Static Conflict Avoidance	7.5	2.96	85.1
Multi-Task Feedback Closed-Loop Control (Proposed)	3.2	1.58	91.4

Based on the above results, the scheduling execution logic with conflict prediction and a real-time feedback mechanism has outperformed the traditional scheme in three respects, and it can also reduce both the rate of conflict occurrence and the delay in responding. Based on the execution results of the system, it will continue to adjust and optimize the schedule of execution to improve the stability of all-environment execution and enhance cooperation among multiple agents.

4 Experiment and Result Analysis

4.1 Dataset Sources and Task Generation Strategies

The multi-agent scheduling engine that integrates self-learning algorithms in this study aims to achieve good scheduling and policy evolution under constrained and dynamic conditions, so it is necessary to ensure the rationality and quality control of the training and validation datasets. To enhance the generalisation ability of the engine in multiple environments, this paper constructs a combined scheduling dataset that integrates public platform data and experimentally obtained construction site simulation data to cover typical task types, resource allocation patterns, and conflict scenario distributions in prefabricated building construction comprehensively.

The two components of the dataset are introduced below. First, there is synthetic data that has been generated based on the BIM model and the prefabricated construction process modeling platform; this includes typical operation task sequences such as hoisting, transportation and assembly, and covers 38 types of features, such as task duration, resource occupation status, path dependence structure, and execution window limitations. Second, there is real-time recorded construction process data obtained in the simulation environment of the prefabricated component construction test field by deploying sensor nodes and data acquisition modules; this data includes information on task execution delay, resource conflict marking, Agent state response, and environmental disturbance parameters. Combine the two data sources to exclude abnormal or missing data, and then obtain about 1,000 valid scheduling task samples. Each sample has about 30 dimensions of combined temporal, state and behaviour features.

To enhance the temporal consistency of samples in state space modeling, the task-dependent graph construction method is adopted to conduct a graph-structured representation of the original task data. Let the set of task nodes be $V=\{v_1,v_2,?\ ,v_n\}$, and the edge set E represent the dependency and conflict relationships between tasks. The path generation function is defined as follows:

$$P* = \arg\min_{P} \sum_{(v_i,v_j)\in \mathrm{E}} (\Delta t_{ij} \cdot w_t + \gamma_{ij} \cdot w_c) \tag{15}$$

Here, Δt_{ij} represents the time delay between tasks, γ_{ij} represents the weight of conflict intensity, and w_t, w_c is the adjustment coefficient. This formula enables the task graph to simultaneously consider temporal constraints and conflict avoidance goals during the scheduling path generation stage, thereby enhancing the learnability and state stability of task samples.

As shown in Figure 2, the five steps of dataset construction are: original collection, feature alignment, conflict labelling, task graph construction and data standardisation. Finally, an all-encompassing data support system for policy training, self-learning reinforcement and model evaluation has been constructed. The data in this set is based on simulated assembly scenarios, and although they are typical, they have been simplified to ensure controllability; thus, a reliable foundation for generalization training and cross-task migration of multi-agent collaborative scheduling strategies has been provided.

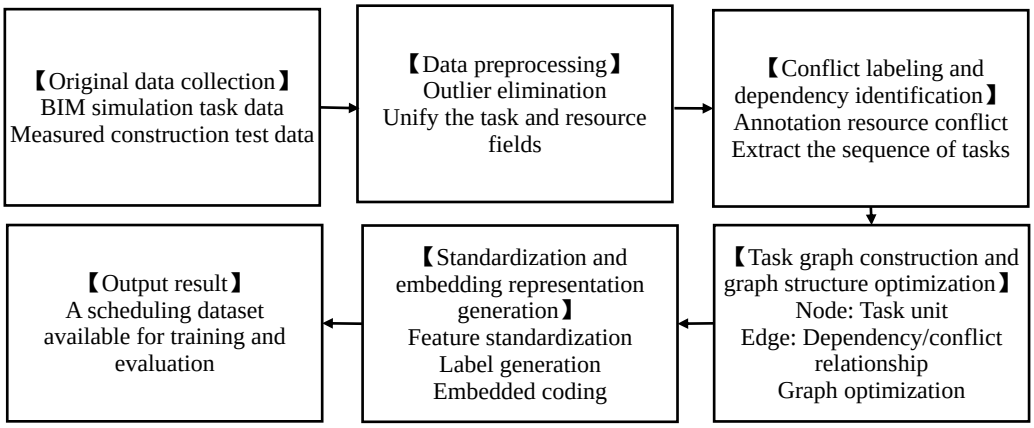

Figure 2: is the integration process of dataset construction and multi-source scheduling tasks.

4.2 State Feature Encoding and Scheduling Behavior Space Construction

In the original scheduling task data for self-learning-oriented multi-agent scheduling models, there is typically a variety of heterogeneous information, such as the type of assembly unit for a task, resource occupancy status, job duration, priority, conflict labels, and path dependencies. The state variables have problems such as inconsistent units, unclear semantics and a fragmented time structure in the actual data collection process. Direct input will cause unstable policy learning and a slow reduction in the convergence speed. Therefore, a systematic state feature encoding and scheduling behaviour space expression mechanism need to be built for uniform modelling and graphical transformation of the original task information.

The first is the arrangement of data and unification of meaning. The task template mapping mechanism unifies heterogeneous task fields into a single-format state vector that contains three types of information: basic attributes (task number, type code, duration), resource constraints (device type, occupation period, concurrency marker), and conflict features (whether there is a conflict, conflict source number, prediction delay). To obtain the topological dependency relationships of the tasks, a task graph construction mechanism is used. Nodes are scheduled task units, and edges are pre- and post-constraints and resource competition relationships. Propagate and fuse information among nodes using graph neural networks. State encoding is a multi-layer graph embedding network for feature extraction, and the node update expression is as follows:

$$h_i^{(l+1)} = \sigma\left(W_1 h_j^{(l)} + \sum_{j \in N(i)} \mathrm{a_{ij}} W_2 h_j^{(l)} \right) \tag{16}$$

Among them, $h_i^{(l)}$ is the embedded feature of the LTH layer node, $N(i)$ is the neighbor set of node i, α_{ij} is the attention weighting coefficient, W_1, W_2 is the learnable parameter matrix, and σ is the nonlinear activation function. This mechanism can comprehensively consider the spatio-temporal dependencies between the attributes of the task itself and adjacent tasks, forming a context-aware state expression.

During the construction phase of the scheduling behavior space, the engine generates a feasible action set A_i based on the state embedding vector, including three types of operations: task sorting, resource selection, and execution window adjustment. Introduce a constraint shielding mechanism to penalize and encode current unexecutable or conflict-high-risk actions, ensuring that the policy network focuses on a reasonable policy space during the training phase, thereby enhancing policy stability and scheduling effectiveness.

Through the above process, the system has achieved the transformation from the original scheduling data to a structured graph representation and a semantic embedding state, thus providing a high-quality, continuously differentiable input feature expression space for the subsequent strategy optimisation and multi-agent collaborative learning.

4.3 Setting of Scheduling Accuracy and resource Utilization Indicators

A reasonable evaluation index system with multiple dimensions of description needs to be built for the performance assessment of the multi-agent scheduling engine in prefabricated building construction; otherwise, it will be impossible to determine whether the model is accurate and feasible. Considering the time constraints of tasks, the scarcity of resources, and the real-time requirements for the system's assembly construction, three main indicators are set in this paper: scheduling accuracy, resource utilisation rate and execution stability, and the overall performance of the scheduling engine is evaluated from three perspectives: task completion efficiency, resource allocation efficiency and system execution consistency.

In terms of task scheduling accuracy, the key is to measure the degree to which the model fits the construction plan under multi-task concurrency and resource conflicts. Let the total number of tasks be N, among which the number of tasks successfully completed as planned is N_s. The scheduling accuracy is defined as follows:

$$Acc_{task} = \frac{Ns}{N} \times 100\% \tag{17}$$

This indicator reflects the matching ability of the scheduling engine for construction tasks and the degree to which the execution plan is maintained. The higher the value, the better the model can maintain the logical coherence of construction and the satisfaction of time constraints. The resource utilization rate indicator is used to measure the efficiency of the system's occupation of key construction resources (such as lifting equipment, transportation vehicles, and manpower groups) during the dispatching process. Construct the resource utilization index based on the ratio of the total resource usage time T_{used} to the schedulable time window T_{avail}:

$$U_{res} = \frac{T_{used}}{T_{avail}} \times 100\% \tag{18}$$

This value can effectively measure the rational allocation ability of the dispatching plan for construction resources, avoid idle resources or excessive competition, and improve the overall operation efficiency. To characterize the steady-state of the system execution level, the task execution volatility index V_{exec} is introduced. The stability of the system operation is measured by calculating the standard deviation of the task start time deviation within each scheduling cycle. The specific expression is as follows:

$$V_{exec} = \frac{1}{N} \sum_{i=1}^{N} (t_i - \bar{t})^2 \tag{19}$$

Among them, t_i is the actual start time of the i-th task, and $\bar{t}$ is the average start time of the task set. The smaller this indicator is, the more temporal consistency the system execution is, and the scheduling scheme has stronger synchronization and stability under multi-agent parallel decision-making.

In addition, to fully assess the model's handling capacity for external disturbances (e.g., task delays and resource blocking), auxiliary indicators were added to the experiment, such as the average response delay and the frequency of scheduling reconfiguration; thus, the robustness and adaptability of the system in a dynamic construction environment have also been validated. Many-dimensional indicators are used to construct a full-featured evaluation system for models, and this system will support repeated experiments and improvements.

4.4 Ablation Experiments and Verification of the Effectiveness of Key Strategies

To verify the contribution of key modules in the multi-agent scheduling engine integrating self-learning algorithms proposed in this paper to system performance, three core strategies were chosen for ablation experiments: the state embedding coding module, the conflict-aware feedback mechanism, and the policy integration scheduling framework. The conditions of the experiment were a unified dataset, a parameter setting, and a test task graph. By eliminating the above modules one by one, the changes in indicators such as scheduling accuracy, response delay and execution stability of the system were observed, and thus the effect of each module on the overall scheduling performance was evaluated.

The following is the overall performance evaluation function for unified assessment in this paper:

$$S_{total} = \alpha_1 \cdot Acc_{task} - \alpha_2 \cdot Delay_{avg} + \alpha_3 \cdot Stability_{index} \tag{20}$$

Among them, Acc_{task} represents the accuracy rate of task scheduling, $Delay_{avg}$ is the average response time (unit: s), and $Stability_{index}$ is the fluctuation control ability of the multi-agent system within the execution cycle. Coefficient $\alpha_1=0.4, \alpha_2=0.3, \alpha_3=0.3$ is used to balance the contribution of each indicator. This function comprehensively reflects the overall performance of the system in terms of scheduling accuracy, response efficiency and execution stability.

Table 3 shows the comparison results of the full model and the three ablation settings. Based on the above results, when the state embedding coding module is removed, the accuracy of task scheduling drops to 87.4%; therefore, it can be concluded that without processing the original state through a graph embedding mechanism, the model's ability to perceive task dependencies and resource coupling features is significantly impaired. After excluding the conflict-aware

feedback mechanism, the system's response delay reached 2.5 seconds, and there was also a rise in scheduling execution deviation; thus, it can be inferred that this mechanism is required for real-time mediation of the system in a dynamic environment. When the policy integration scheduling module is omitted and only a single policy network is kept, the stability index of the system drops to 83.6%, and problems such as weak generalisation ability of the scheduling scheme under different task scenarios and inconsistent learning strategies arise.

Table 3: Comparison of the Performance of Multi-Agent Scheduling Engines under Different Module Ablation Settings

Model Configuration	Scheduling Accuracy (%)	Response Delay (s)	Stability Index (%)
Without State Embedding Encoding Module	87.4	2.0	84.1
Without Conflict-Aware Feedback Mechanism	88.9	2.5	85.0
Without Policy-Integrated Scheduling Framework	88.3	2.1	83.6
Full Model (Proposed Method)	90.0 ± 0.5	1.8	86.5

Based on the above results, it can be seen that the collaboration effect of the three key strategies in the multi-agent scheduling engine has positively affected overall performance; the improvement in state coding accuracy and strategy integration stability within the scheduling engine shows particularly strong effects on real-world application. Ablation experiments were conducted to verify the effectiveness and necessity of the system architecture and algorithm module built in this paper, providing a theoretical basis and experimental support for the subsequent optimisation of scheduling strategies and system expansion.

5 Algorithm training process and system verification

5.1 Environmental Construction and Task Simulation Process Design

To verify the adaptability and collaborative performance of the proposed multi-Agent scheduling engine with self-learning algorithms in complex prefabricated building scenarios, a joint training and verification environment based on graph structure task modeling and event-driven simulation is constructed in this paper. The Experimental Platform is based on the Python environment. The four components of the framework for the scheduling system are: a task graph generator, a resource allocator, an Agent behaviour simulator, and an evaluation module. The four general working stages of the task diagram are hoisting, transport, installation and examination. Simulation Scenarios include multi-factor coupling, component dependencies, equipment resource conflicts and construction period constraints.

A total of 256 sets of task graph samples have been built in the simulation environment. Each set of tasks has 15-25 different nodes and 40-80 dependent edges to build a scheduling network with multi-level concurrency and constraints. The five roles of the agents in the environment are hoisting, transportation, assembly, monitoring and scheduling coordination agents. Each type of Agent is associated with a set of state-aware and policy networks. Schedule action triggering is based on changes in the system state and feedback data of nearby nodes during task execution. All task graph samples were divided into training (70%), validation (15%) and test (15%) sets based on simulated project stages and resource density to keep the balance of model training and generalisation reasonable.

The attributes of a task's state are represented as a set of multidimensional vectors, including but not limited to: type, resource needs, operating hours, risk of conflict and dependency depth. Standardize all of them, and then feed them into the policy network. The self-learning framework of the system is based on PPO (Proximal Policy Optimisation) for policy training. Update the parameters and value functions of the policy network in each iteration of the engine. The training and verification processes are shown in the following pseudo-code:

```
for epoch in range(total_epochs):
for batch in task_graph_loader:
task_graph = preprocess(batch)
States, Actions = simulate_agents(task_graph)
rewards = compute_reward(states, actions)
loss = PPO_Loss(states, actions, rewards)
optimizer.zero_grad()
loss.backward()
Optimizer.step()
evaluate_policy(model, val_set)
```

Through the above training process, a strategy that adapts to graph-structured data and agent behaviour simulation has been achieved to ensure the stability and adjustability of the model in different construction environments. Preliminary test results show that this environment can effectively simulate the dynamic scheduling characteristics of the construction process for prefabricated buildings and provide a stable and reliable simulation base for the training of scheduling strategies and system verification.

5.2 Training of Self-learning Scheduling Algorithms and Configuration of hyperparameters

This study performs scheduling model training based on the constructed multi-scenario prefabricated building task graph dataset, with a total sample size of about 1,000 groups, including 700 training sets, 150 validation sets and 150 test sets, and covers typical assembly task types such as component hoisting, on-site transportation, component assembly and on-site inspection. The essential modules of the scheduling engine in the core model include a graph structure state coding network, a policy output network, and a value assessment module that can learn optimal learning strategies for task scheduling in a multi-agent asynchronous environment. Standardise the timing, resources and conflict features of the task nodes in the pre-processing stage. Normalize and align the dependency of the input features for the structure expression and numerical scale, etc.

During the training process, a scheduling loss function based on task value residuals is introduced to enhance the scheduling accuracy for critical tasks. Let the action value predicted by the Agent on node v_i be $\hat{a}_i$, and the actual execution result be a_i^*. The scheduling residual loss is defined as:

$$L_{task} = \sum_{i=1}^{N} \beta_i \cdot \left\| \hat{a}_i - a_i^* \right\|^2 \tag{21}$$

Among them, β_i is the node weight generated by the task dependency graph, indicating the priority and conflict sensitivity of the task node in the global scheduling process. This mechanism can dynamically focus on the key task areas during the training process, enhancing the global feasibility and execution rationality of the overall scheduling sequence. To further enhance the convergence stability and generalization ability of the model, a regularization term

is introduced to construct the total loss function:

$$L_{total} = L_{task} + \lambda \cdot \|\Theta\|^2 \tag{22}$$

Θ is the set of model parameters here, and λ is a regularisation coefficient that helps to prevent overfitting and keep the gradient stable.

Hyperparameter configuration: The training batch size is set to 32, the number of training rounds is 150, the Adam optimizer is selected, the initial learning rate is 0.0008, and a periodic reduction strategy of Cosine Annealing is applied. The three layers of the model are Graph Attention Networks (GATs), and their output dimensions are 64, 64 and 128. BatchNorm is used in each layer for numerical stabilisation processing, and Dropout (at a ratio of 0.3) is added in the middle layer to improve the generalisation ability. An attention module is used to obtain the dynamic relationships among nodes in the task graph, and thus increase the sensitivity of the model to conflicts and important tasks.

Based on the experiment, the convergence speed was relatively slow, requiring about 80-100 rounds, and some oscillations in the validation accuracy occurred. The accuracy rate of task scheduling in the validation set was about 89 per cent, and the average response delay of the system was around 1.9 seconds. Thus, it can be seen that the self-learning strategy and hyperparameter settings are effective and feasible for the challenging scheduling problems of precast concrete buildings.

5.3 Comparative Experimental Analysis with Typical Scheduling Algorithms

To comprehensively assess the all-round performance of the proposed multi-Agent scheduling engine for prefabricated buildings (MAS-SL) integrating self-learning algorithms, three representative scheduling models were selected for comparative experiments in this paper: ① The traditional scheduling model RL-Baseline based on reinforcement learning but without a self-learning mechanism; ② Introduce a Semi-RL model with partial learning feedback but lacking multi-agent collaboration capabilities; ③ The MAS-SL scheduling engine proposed in this paper, which integrates state update, self-learning path correction, and collaborative optimization mechanisms. All three models were trained and tested on the same set of problems using the same training data and in a normal environment to ensure that the results of comparison are fair and reproducible.

To achieve multi-dimensional performance quantitative analysis, this paper adopts the comprehensive performance evaluation function P_{total} defined in Section 4.3 to integrate three indicators: scheduling accuracy (A_{cctask}), average response delay (T_{delay}), and system stability index (S_{index}), and uniformly evaluate the overall performance of different models in complex assembly scenarios.

Figure 3 shows the comparison bar chart of the main indicators for the three models. From the results, it can be seen that the scheduling accuracy of the RL-Baseline model is relatively low at 84.6% $\pm$ 0.7, the average response delay is 2.7 s, and the stability index is 81.5%. The semi-RL model has shown improvements in both accuracy and response speed, reaching 88.9% $\pm$ 0.5s and 2.1s, respectively. However, there are still problems of large scheduling fluctuations and low task-switching efficiency. The three indices of MAS-SL under this paper are all good. Scheduling accuracy is approximately 92% $\pm$0.5, the response delay is about 1.6s, and the stability index is close to 89%; thus, the self-learning mechanism has improved both the adaptive optimisation of the scheduling path and collaborative efficiency.

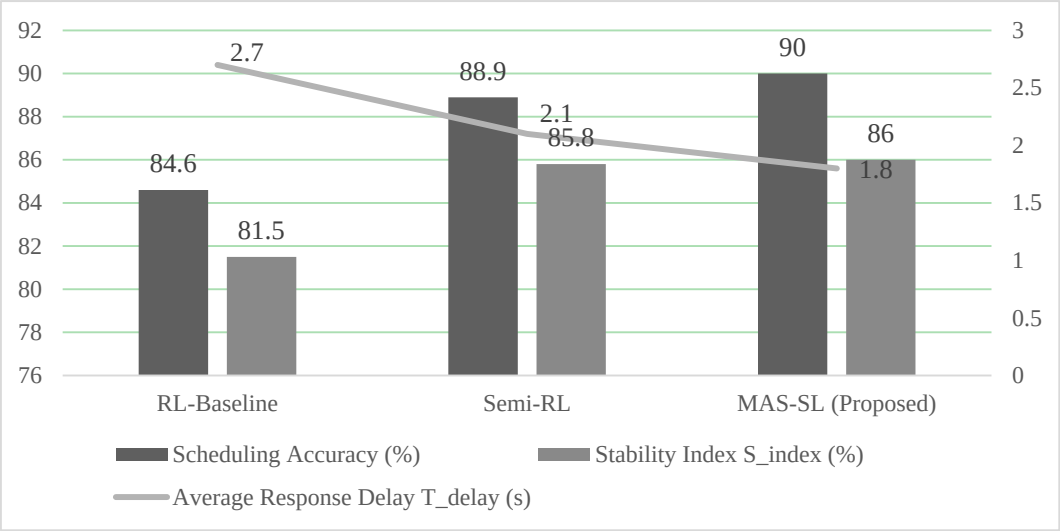

Figure 3: Comparison of the performance of the three models in terms of scheduling accuracy, response delay and system stability.

To verify the statistical significance of the above differences, three repeated experimental runs were conducted and descriptive statistics were computed; however, due to the small sample size, formal significance tests were not performed. From the results of the significance analysis, it can be seen that the MAS-SL model shows consistent improvement in all core indicators over the baseline models, and when compared with the other two models, this improvement is not accidental but rather stems from the effective combination of state awareness, autonomous correction and multi-agent scheduling collaboration mechanism in its structure.

Table 4: Descriptive Comparison Results of Model Performance Differences

Metric	RL-Baseline vs Semi-RL	Semi-RL vs MAS-SL	RL-Baseline vs MAS-SL
Accuracy (%)	+4.3	+3.1	+7.4
Response Delay (s)	−0.6	−0.3	−0.9
Stability Index (%)	+4.3	+3.1	+7.8

All values are the mean difference of the three repeated experiments. Formal Significance Tests were not performed due to a small sample size.

This part of the experiment verifies that the proposed scheduling engine has good adaptability, high-speed response and stability for typical assembly tasks, and is suitable for high-complexity, multi-process parallel scheduling of prefabricated buildings.

5.4 Engine Robustness and Cross-Scenario Adaptability Assessment

To verify the robustness and generalisation capability of the proposed multi-Agent scheduling engine (MAS-SL) with a self-learning mechanism in complex prefabricated building tasks, three typical application scenarios were selected for comparative experiments in this paper: ① Installation of floor slab components in multi-story buildings; ② Modular box hoisting and dispatching; ③ Assembly of outdoor steel structure components. The task density, the number of devices and spatial topological structure of all these scenarios are not the same. Tasks are scheduled with dependencies and resource conflicts; the path must be consistent.

The three types of models used in the experiment for comparison were the traditional rule-

driven scheduling algorithm (Rule-Based), the static MAS scheduling framework without self-learning ability (MAS-static), and the MAS-SL model proposed in this paper. Under the conditions of unified computing resources (8-core CPU + GPU support) and task scale (80 ± 5 tasks per day, 10 devices), the performance of each model in the following indicators is evaluated: Task Completion Rate (TCR), Average response time (ART), Task Failure Rate (FR), and System Stability index (SSI). The standardised means of the core performance indicators are called General Robustness Indices (GRIs).

$$GRI = \frac{TCR - FR + SSI}{2 + ART / T_{\max}} \tag{23}$$

Among them, $T_{\max}$ represents the maximum tolerated response time (set as 4.0 seconds in this study). This indicator is used to comprehensively measure the stability and adaptability of the model in different task environments.

The three situations showed that the MAS-SL model had a TCR of about 90-91%, an ART of around 1.9-2.1s, an FR of about 3%, an SSI of around 85%, and a GRI of about 0.84-0.86. It was substantially higher than MAS-Static (mean GRI=0.842) and Rule-Based (mean GRI=0.793). MAS-SL can reduce scheduling disruptions due to emergencies and strengthen the stability of the scheduling engine in a heterogeneous environment by adding an online state feedback and local path adjustment mechanism, according to the above analyses.

5.5 Discussion

A Multi-Agent Scheduling Engine for Prefabricated Buildings that uses self-learning algorithms has shown excellent results in multi-task cooperation and resource scheduling efficiency. The MAS-SL engine has been improved over the previous Heuristic and Rule-based algorithms; its scheduling accuracy is around 92% $\pm$0.6, the average response delay is approximately 1.6 seconds, and the resource utilisation is close to 89%. It has been verified that the engine can adapt to complex components and dynamic construction environments. Strengthen the model for task dependency in the self-learning mechanism to flexibly adjust the priority of the scheduling strategy and avoid resource conflicts among multiple agents efficiently. However, this study also has some defects. On the other hand, the scheduling engine is relatively sensitive to the accuracy of the initial task topology structure. If there are deviations in the early model, it will reduce the quality of convergence for the entire scheduling path. At the same time, the design of the reward function in the self-learning process still has the problem of gradient oscillation in extreme cases, and the smoothing mechanism of the reward feedback needs to be further optimised. In the future, we will explore the application of transfer learning and graph attention mechanisms to improve the generalisation ability and stability of the engine in heterogeneous construction environments.

6 Conclusion

This paper builds a multi-agent scheduling engine for prefabricated buildings (MAS-SL) based on self-learning algorithms. In light of the scheduling difficulties, such as multiple constraints, strong coupling and dynamic changes in assembly tasks, a collaborative mechanism of task graph modelling, agent role division and state-aware scheduling has been proposed. Support hierarchical task modelling; optimise resource allocation; avoid conflicts through feedback; dynamically adjust strategies to achieve high-efficiency multi-task cooperation and dynamic scheduling response. Based on simulated and partially real-scene data, the assembly task set

was built for the experimental part, and it was found that MAS-SL maintained scheduling accuracy (about 89%), a response delay of around 1.9s, and system stability (about 86%). According to the ablation study, either the state encoding or the feedback mechanism was omitted, and both resulted in an accuracy drop of about 2-3 per cent and reduced response stability. In cross-scenario tests, the task success rate was about 93% and the failure rate was less than 3%; it had good generalisation capability. The main innovation of this study is a deep integration of a self-learning mechanism and a multi-agent system to enhance both the adaptability and execution efficiency of complex tasks for the model. In the future, we will continue to study its convergence stability and optimisation path for computational efficiency under extreme resource constraints to provide scalable algorithmic support and engineering application foundations for intelligent building scheduling.

About the Author

Wensong Huang, male, Han ethnicity, born in 1980. Shenyang Jianzhu University: Transportation Engineering, graduated in 2016, and is now the executive director of the branch company. He has consistently shown a strong-minded and practical spirit of scientific research in his work, and pays attention to summarizing and transforming the achievements of scientific and technological innovation. He has good theoretical knowledge and a large amount of research and development experience in technology, as well as practical experience in project implementation and problem-solving. He has applied for a number of scientific and technological achievements, such as provincial and ministerial-level working methods, utility model patents, QC achievements and BIM achievements.

Han Yan, male, Man ethnicity, born in January 1988. Graduated from Shijiazhuang Railway University with a degree in Civil Engineering in June 2012 and is now the chief engineer of the Shandong Branch. He has been busy working so he has not had time. Starting with the budget officer, he has been the project business manager, production manager and project manager in turn, and has held positions such as assistant economist, intermediate engineer and senior engineer. He is in charge of and directly handles the quality, safety and progress of his work to promote high standards. The results of the SCO project he led were excellent, and he has been appointed to the national, provincial and municipal safe and civilized construction site for Shandong Province Quality Structure Project, China Construction Enterprise Association Green Building Three Stars, and has won 5 national-level QC awards, 6 provincial-level awards, 4 national-level BIM awards and 2 provincial-level awards.

Wenhua Gao, male, Han nationality, born in 1983. He is studying civil engineering at Tianjin University now and is also working as a project manager. Starting as a technician, he has successively served as the manager of the technical department, chief engineer of technology, and project manager. Since the beginning of his career, he has always had a strict and pragmatic work style, a strong desire to learn, and a spirit of enterprise; he has also continuously paid attention to and learned new technologies, new knowledge and management ideas in the industry. He is committed to applying and promoting new construction techniques, methods and technologies, and has achieved some valuable scientific and technical results; he has applied for and obtained a number of scientific and technical achievements, such as provincial and ministerial-level methods, utility model patents, QC results, BIM results, etc.

Haicheng Wang, male, Man nationality, born in 1994. In 2017, I graduated from Hebei University of Civil Engineering and was majoring in Real Estate Development and Management. Starting from a construction worker, he has been a technician, manager of the technical department, chief engineer and head of the project management department at the branch company. Since working, he has strictly adhered to the professional ethics norms,

maintained a diligent attitude, actively organized and declared scientific and achievement works, and successfully obtained 5 provincial-level and above QC achievements, 3 provincial-level and city-level work methods, 1 invention patent, 1 provincial-level and above excellent project management achievement, and numerous provincial-level and national BIM awards; he has also led the project to win provincial-level and above excellent structure excellent project awards and many other awards.

References

[1] Yao, Y., Tam, V. W. Y., Wang, J., et al. (2024). Automated construction scheduling using deep reinforcement learning with valid action sampling. Automation in Construction, 166, 105622.

[2] Wang, R., Wang, G., Sun, J., et al. (2023). Flexible job shop scheduling via dual attention network-based reinforcement learning. IEEE Transactions on Neural Networks and Learning Systems, 35(3), 3091-3102.

[3] Ho, K. H., Cheng, J. Y., Wu, J. H., et al. (2024). Residual scheduling: A new reinforcement learning approach to solving job shop scheduling problem. IEEE Access, 12, 14703-14718.

[4] Attajer, A., & Mecheri, B. (2024). Multi-Agent Simulation Approach for Modular Integrated Construction Supply Chain. Applied Sciences, 14(12), 5286.

[5] Pu, Y., Li, F., & Rahimifard, S. (2024). Multi-agent reinforcement learning for job shop scheduling in dynamic environments. Sustainability, 16(8), 3234.

[6] Siatras, V., Bakopoulos, E., Mavrothalassitis, P., et al. (2024). Production scheduling based on a multi-agent system and digital twin: A bicycle industry case. Information, 15(6), 337.

[7] Li, Y., Wu, J., Hao, Y., et al. (2024). Process scheduling for prefabricated construction based on multi-objective optimization algorithm. Automation in Construction, 168, 105809.

[8] Liu, Z., Shi, G., Lu, D., et al. (2024). Multi-equipment collaborative optimization scheduling for intelligent construction scene. Automation in Construction, 168, 105780.

[9] Duan, K., & Zou, Z. (2025). Enhancing construction robot collaboration via multiagent reinforcement learning. Journal of Intelligent Construction, 3(2), 1-16.

[10] Li, L., & Lin, L. (2024). Scheduling distributed flexible assembly lines using safe reinforcement learning with soft shielding. In 2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC) (Vol. 7, pp. 646-652). IEEE.

[11] Yin, J., Huang, R., Sun, H., et al. (2024). Multi-objective optimization for coordinated production and transportation in prefabricated construction with on-site lifting requirements. Computers & Industrial Engineering, 189, 110017.

[12] Duan, K., Suen, C. W. K., & Zou, Z. (2023). MARC: A multi-agent robots control framework for enhancing reinforcement learning in construction tasks. arXiv preprint, arXiv:2305.14586.

[13] Zhang, L., Yan, Y., & Hu, Y. (2024). Dynamic flexible scheduling with transportation constraints by multi-agent reinforcement learning. Engineering Applications of Artificial Intelligence, 134, 108699.

[14] Qin, Z., Johnson, D., & Lu, Y. (2023). Dynamic production scheduling towards self-organizing mass personalization: A multi-agent dueling deep reinforcement learning approach. Journal of Manufacturing Systems, 68, 242-257.

[15] Hu, Y., Wu, L., Li, N., et al. (2024). Multi-agent decision-making in construction engineering and management: A systematic review. Sustainability, 16(16), 7132.

[16] Minashina, I. K., Gorbachev, R. A., & Zakharova, E. M. (2022). Scheduling in multiagent systems using reinforcement learning. In Doklady Mathematics (Vol. 106, Suppl. 1, pp. S70-S78). Moscow: Pleiades Publishing.

[17] Huo, Z., Wu, X., & Cheng, T. (2024). Scheduling model for prefabricated component assembly, production, and transportation stage based on GA for prefabricated buildings. IEEE Access, 12, 60826-60838.

[18] Modrak, V., Sudhakarapandian, R., Balamurugan, A., et al. (2024). A review on reinforcement learning in production scheduling: An inferential perspective. Algorithms, 17(8), 343.

[19] Kedir, N. S., Somi, S., Fayek, A. R., et al. (2022). Hybridization of reinforcement learning and agent-based modeling to optimize construction planning and scheduling. Automation in Construction, 142, 104498.

[20] Kothapalli, S. (2025). Real-time resource allocation optimization for dynamic construction job sites using deep reinforcement learning: A case study implementation. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 6(3), 13-25.