Engineering, Construction and Architectural Management

Deep reinforcement learning-based schedule optimization for parallel precast production

Yuan Yao, Vivian WY Tam, Jun Wang, Khoa N Le and Anthony Butera
School of Engineering, Design and Built Environment, Western Sydney University, Sydney, Australia

Received 19 March 2025
Revised 11 April 2025
13 May 2025
Accepted 4 June 2025

Abstract

Purpose – With the increasing use of precast concrete elements in off-site construction, optimizing precast component production scheduling (PCPS) has become critical for improving construction efficiency. This study aims to develop a deep reinforcement learning (DRL)-based scheduling optimization method for parallel precast production to minimize earliness and tardiness penalties as well as the makespan.

Design/methodology/approach – A parallel production process model is developed considering resource constraints, including crew quantities and fixed mold plates. A pre-trained DRL model is employed for rescheduling under varying precast orders with different quantities and due dates. The practicality of this approach is validated using real case data from field studies, comparing its performance with traditional dispatching rules (DPs) and the genetic algorithm (GA).

Findings – The DRL-based method generates production schedules that are viable for practical applications. Compared to traditional DPs and GA, the proposed approach demonstrates superior stability, enhanced rescheduling capability and reduced computational time.

Practical implications – The proposed DRL-based scheduling method offers a practical and efficient solution for optimizing precast production scheduling. It enhances decision-making in dynamic construction environments by reducing penalties and makespan while improving scheduling adaptability.

Originality/value – This study expands the limited research on parallel PCPS by introducing a DRL-based approach, which integrates scheduling optimization with dynamic rescheduling adaptability under real-world conditions.

Keywords Deep reinforcement learning, Parallel production, Production optimization, Rescheduling

Paper type Research article

1. Introduction

Precast components (PCs) refer to structural or architectural elements that are manufactured in a controlled factory environment before being transported to a construction site for installation. Utilizing PCs in construction offers several advantages over traditional cast-in-place construction methods (Jang and Lee, 2018; Anvari *et al*., 2016), including environmental sustainability (Ma *et al*., 2021), high quality, and on-site time savings (Mao *et al*., 2013; Tam *et al*., 2015). For a precast concrete project, the production of prefabricated components is a critical factor constraining the project's completion timeline. As the PC market grows (Zhou and Ren, 2020), issues may arise due to the absence of production planning and scheduling, leading to early or delayed deliveries, excessive inventory, prolonged construction times, and increased expenses (Wang *et al*., 2018). Therefore, PC production scheduling (PCPS) is crucial for the PC industry.

Recently, research has been conducted to identify several practical concerns in real-world PC production. First, current practices primarily rely on classic dispatching rule (DP) methods (Kim *et al*., 2017; Lin *et al*., 2019). Earliest Due Date (EDD) and Shortest Process Time (SPT)

Engineering, Construction and Architectural Management
Vol. 32 No. 13, 2025
pp. 285-318
Emerald Publishing Limited
e-ISSN: 1365-232X
p-ISSN: 0969-9988
DOI 10.1108/ECAM-03-2025-0429

are the most commonly utilized rules for managing a large number of production orders. While these DP methods can quickly generate a feasible plan, they lack flexibility handling diverse orders and cannot guarantee an optimal scheduling policy. DPs often oversimplify production objectives, which typically results in convergence to local optimum when considering multi-objective optimization. Second, human expertise continues to play a crucial role in determining detailed production plans (Du *et al*., 2020), especially regarding resource allocation across multiple production lines. Schedulers often can only provide estimated values for various resource needs and lack a clear understanding of the resource flow direction. This can lead to suboptimal utilization of resources and create challenges in achieving multiple optimization objectives. Third, there is an increasing demand for rapid rescheduling in response to parallel production system with real-world constraints (Wang *et al*., 2021), as order quantities and due dates are often variable. Schedulers need a constrained rescheduling method that can adapt to the diverse demands of different orders.

Many studies have been made to propose optimal precast production scheduling methods (Wang *et al*., 2023). Among current studies, heuristic algorithm especially Genetic Algorithms (GAs) are the most commonly used baseline algorithm (Peiris *et al*., 2023; Xie *et al*., 2024). Leu and Hwang (2001) were the first to applyGA with improved random crossover and mutation strategies to solve the resource-constrained production scheduling problem in PCPS. GA demonstrated flexibility in generating near-optimal solutions for the proposed flow shop model. After their work, GA was improved to further solve multi-objective schedule optimizations such as minimizing total makespan, penalties for earliness, and penalties for tardiness. Those improved GA outperformed the traditional DP methods in multi-objective tasks (Chan and Hu, 2002b; Dan *et al*., 2021). However, most research has focused on the idealized setting of a single production line (Du *et al*., 2020, 2021), while actual factory environments typically involve multiple parallel production lines to execute order manufacturing concurrently (Dan *et al*., 2021). Studies on optimization in parallel production are limited and their core algorithms predominantly rely on GA (Yang *et al*., 2016; Wang *et al*., 2021). As an Evolutionary Algorithm (EA), a key limitation of GA is its requirement for numerous iterations to generate an optimal schedule. The required number of iterations can significantly impact performance, with additional iterations leading to prolonged computational time. Moreover, the scheduling policies produced by the GAs lack the adaptability to changing initial conditions. For different PC orders, the GA needs to be recomputed to obtain rescheduling results. This means that GA cannot be used as a pre-calculated scheduling policy. In the complex and dynamic scenarios typical of real-world applications, the use of GAs presents significant challenges.

Compared with evolutionary algorithms, Deep Reinforcement Learning (DRL) shows superiority in its generalization ability and has been applied in a wide range of research domains (Sivamayil *et al*., 2023). A well-trained RL agent could even handle very complex tasks such as StarCraft II (Vinyals *et al*., 2019), Go Game (Schrittwieser *et al*., 2020) and flexible recommendation systems (Sivamayilvelan *et al*., 2024). In the field of PCPS, Du and Li (2024) integrated DRL with an EA to optimize parallel PCPS. They used DRL as an operator selector for evolutionary algorithms. The schedules produced by the EA required further processing through two additional refinement strategies. However, their method was essentially still an EA and algorithm iteration remains mandatory. Thus, their method cannot be used as a pre-trained rescheduling policy. Kim *et al*. (2022) applied DRL to directly make an optimal production plan with a single objective of minimizing the tardiness penalty. They designed an agent with the action space consisting of four classical DPs. However, their work did not discuss DRL performance on parallel production scenarios. Their method still relies on DPs to decide production sequences. The DRL policy can select only one job at each decision point, which cannot be directly implemented on parallel production scenarios.

Despite the growing interest in DRL for production scheduling, current studies lack a thorough investigation into parallel PCPS and the rescheduling capabilities of pre-trained DRL policies. The complexity of parallel production lies in the combinatorial explosion of potential job sequences across multiple production lines and diverse PC types. A major challenge in applying DRL as a pre-trained scheduling policy is enhancing its ability to generalize across varying production conditions. Generalization is critical for enabling DRL to generate feasible and optimized schedules in scenarios that differ from the original training configuration.

To address these practical demands and bridge the identified research gaps, this study proposes a DRL-based multi-objective scheduling framework that minimizes total makespan and earliness/tardiness penalties while incorporating real-world constraints such as mold plate availability, limited labor resources, and working hour restrictions. Unlike traditional methods that require retraining or recalculation for each new production scenario, the proposed approach develops a generalized policy capable of rapid and adaptive rescheduling. The framework is designed to support both initial schedule generation and responsive adjustments, offering a practical and scalable solution for modern precast production systems. The key contributions are summarized as follows:

(1) A novel DRL architecture with an efficient and scalable action space design for parallel PCPS.

The proposed framework introduces a new action space formulation that supports the simultaneous selection of multiple jobs for parallel processing. As the number of PC types and production lines increases, the number of possible job combinations expands rapidly, which can hinder training efficiency due to the large parameter space. To mitigate this, the action space is strategically constrained, and an action masking mechanism is applied to dynamically mask invalid actions during training and inference. This approach significantly reduces computational complexity and enhances training convergence. As a result, production managers can generate optimized schedules quickly and make timely decisions on incoming PC orders.

(2) A generalizable DRL model capable of effective rescheduling under varying production scenarios.

The state representation integrates both resource utilization and order progress, capturing the dynamic status of the production environment. A deep neural network, based on convolutional layers, is used to extract rich features from this composite state. Crucially, the state design is decoupled from specific job quantities and due dates, allowing the DRL model to generalize across different scheduling tasks. Once trained with fundamental facility parameters (e.g. mold plate availability and labor resources), the resulting pre-trained DRL policy can be directly applied to reschedule tasks with different order configurations—without the need for retraining. This enables practical deployment in real-world PC factories where order variations are frequent and rapid rescheduling is essential.

The rest of this paper is arranged as follows: Section 2 introduces the related works in the field of PCPS. Section 3 describes the problem definition and the optimization objectives. Section 4 illustrates the details of proposed DRL method. Section 5 conducts experiment on case studies. Section 6 makes discussions on the results and Section 7 analyses the limitation. Finally, Section 8 makes the conclusion of this study. The key abbreviations of this paper are listed in Appendix 1.

2. Related works

2.1 Production scheduling approaches for precast components

PCPS is an NP-hard problem due to its complexity, combinatorial nature, the presence of discrete variables, and the lack of a polynomial-time algorithm that can solve the problem

exactly (Du and Leung, 1990; Lin and Liao, 2013). Researchers have conducted studies on precast scheduling optimization over the past decades. Several optimization theories have been adopted to illustrate the PCPS problem. Dawood (1995) proposed a simulation-based method and defined the PCPS problem as a job shop scheduling problem. Chan and Hu (2001) chose the Flow Shop Scheduling Problem to identify the relationships between jobs and machines in the production processes. Given these diverse problem formulations, the optimization algorithms used are also varied. Chan and Hu (2002a) adopted a Constrained Programming (CP) approach and their research showed CP outperformed traditional heuristic rules in makespan and cost. Khalili and Chua (2014) introduced a mixed integer linear programming and the model reduced the production cost by over 9%. Apart from these algorithms, hybrid methods that combine multiple algorithms were proposed. Metaheuristic algorithms are also widely used in the field of project management (Alshboul *et al*., 2025a, b). Chang and Han (2021) improved a discrete differential evolution algorithm with a local search strategy. Xiong *et al*. (2021) proposed a hybrid tabu search and iterated greedy algorithm and compared it with a hybrid GA and variable neighborhood search and a two-phase heuristic method. The computational analysis indicated that the proposed hybrid method performed better than others on average. Du *et al*., (2023) used a biogeography-based optimization algorithm to schedule PC lean production. However, these studies did not address parallel production or rescheduling capability.

Apart from these diverse optimization algorithms, GA is more widely adopted as it provides better optimization in off-site construction and PCPS (Li and Love, 1998). Leu and Hwang (2001) first used GA to solve the resource-constrained production scheduling problem in PCPS. They adopted improved random chromosome operators for gene insertion and mutation, which outperformed previous algorithms in optimizing makespan. GA showed flexibility in generating near-optimal solutions for the proposed flow shop model. In their later research (Leu and Hwang, 2002), an improved union crossover operator was employed to accommodate the random key representation. Chan and Hu (Dan *et al*., 2021) first proposed a normalized GA to optimize multiple objectives. Inspired by their research, the most frequently multi-objectives include minimizing makespan as well as earliness and tardiness penalties (Chan and Hu, 2002b). Other major concerns in multi-objective PCPS research include the cost of mold type change (Benjaoran and Dawood, 2006), minimizing machine idle time (Liu *et al*., 2023) , and minimizing total labor over time (Liu *et al*., 2021) are also major concerns in multi-objectives PCPS research. Dan and Liu (2024) proposed a GA based integrated scheduling method for production and transportation, which aimed at minimizing earliness/tardiness penalties and transportation cost under delivery time window. Mao *et al*., (2024) used a hybrid GA for just-in-time delivery of PCs. Xie *et al*., (2025) improved GA with topological sorting priority method to optimize production makespan. Their work presented superiorty compared to other optimization methods.

In terms of rescheduling, Ko (2010) utilized a schedule adjustment principle to address the issue of demand variability based on customer behaviors, although their approach lacked a robust optimization mechanism. Kim *et al*. (2020) introduced a discrete-time simulation method to manage due date uncertainties. Du *et al*. (Ko and Wang, 2011) proposed an elite GA to address demand fluctuation. Later, they proposed a multi-agent model with GA to establish a dynamic decision support framework. However, their study was limited to a single production line. In terms of parallel production with multiple production lines, Wang and Hu (2018) utilized GA to dynamically reschedule in response to frequent demand fluctuations across different types of PCs. Wang *et al*., (2021) developed a hybrid GA method aimed at improving machine efficiency and mold utilization during machine breakdowns. Nevertheless, as an iteration-based metaheuristic, GA requires reinitialization of configurations when making reschedule, which means it cannot be used as a pre-trained scheduling policy. Fixed parameters, such as the mutation rate, do not dynamically adapt to different rescheduling scenarios, potentially leading GA to converge on local optima in rescheduling optimization tasks.

2.2 DRL in precast production

DRL is an area of artificial intelligence that is initially inspired by the learning abilities of animals in nature. It simulates the response of the animal, namely the agent, to the received environmental feedback reward after taking an action (Figure 1 (Yao *et al*., 2024)). In DRL methods, the agent aims to maximize the total reward by interacting with the dynamic environment through making optimal decisions (Sutton and Barto, 2018). Unlike traditional supervised and unsupervised learning, DRL algorithms do not require external data for the training and validation (Kaelbling *et al*., 1996; Kurinov *et al*., 2020). The agent automatically collects data and feedback from the simulation environment and eventually reaches an optimum action policy. Moreover, RL methods show superiority in the area of sequential decision-making (Moerland *et al*., 2020). The implementation of DNN makes RL methods capable of learning from high-dimensional inputs, which means DRL can solve real-world complexity (Mnih *et al*., 2015).

The application of DRL in production scheduling can be categorized into two types: (1) directly generating optimized production schedules through interaction with the environment from starting to ending; and (2) using DRL as heuristic operator to refine an initial production schedule. The first approach is designed to derive a policy that facilitates end-to-end production scheduling. Similar research has been conducted in the field of job shop scheduling to optimize production problems with real-world complexity (Li *et al*., 2022; Pan *et al*., 2021; Luo *et al*., 2021; Du *et al*., 2022b). The second approach typically requires DRL to be integrated with heuristic algorithms (Chen *et al*., 2020; Lin *et al*., 2022; Du *et al*., 2022a).

With DRL application in related field, the research on precast production schedule optimization remains limited. For the first approach, Kim *et al*. (2022) proposed a Deep Q-Network (DQN) method to generate precast production schedules with minimum tardiness penalties. They adopted four DPs as DRL actions to interact with their precast environment. Their results indicated that using DRL for end-to-end schedule optimization outperformed other classic DPs. However, their method still relied on DPs as actions to schedule PC orders. Parallel production was not addressed in their case studies, as their action design only supported selecting one job at each decision point. In contrast, real-world parallel production often requires the execution of multiple jobs simultaneously. Their DRL architecture also lacked generalization ability as their state design only focused on the progress of PC orders. Resource utilization was not adequately considered in their DRL framework. For second approach, Du and Li (2024) further discussed the DRL integrated heuristic method application in distributed precast production. They designed 17 evolution operators as actions for algorithm iteration and refinement. However, both studies have limitations in considering various constraints. Kim *et al*. (2022) set limitations to the amount of molds, but they did not set same limitations to the labors. Du and Li (2024) considers only time-of-use electricity price as the constraints. Their work did not take other

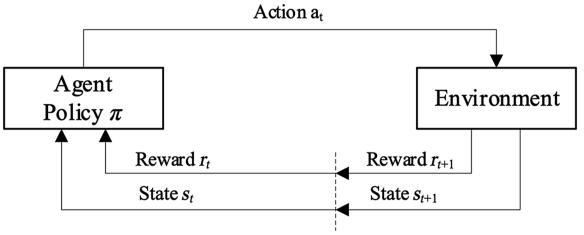


Figure 1. DRL framework. Source: Authors' own work (2024)

typical resources constraints or labor constraints into account. The omission of these key constraints renders their approaches less consistent with actual production conditions, highlighting a major limitation of their studies.

2.3 Summary and research gaps

Table 1 presents a summary for the most relevant studies with details of their methodologies based on the literature review. GA methods can address the PCPS problem, accommodating various constraints and multiple objectives. Nonetheless, when rescheduling tasks become necessary, GA requires recalculation with new initial configurations for the rescheduling tasks, as GA only provides optimized schedules but not a policy for scheduling. Current DRL based methods also have limitations. Kim *et al*. (2020) considered only a single objective and did not consider labor constraint. Their work was not capable of optimizing parallel PCPS. Dy and Li (2024) did not explore the use of pre-trained DRL as a rescheduling policy in parallel production, as their method still required an EA for schedule optimization. To bridge the gaps, this study proposes a multi-objective DRL method for parallel production. The proposed DRL is capable of generating a pre-trained policy suitable for task rescheduling.

3. Problem definition

This study categorizes PC production as a fixed-location production system, which requires fewer material and labor movements (Yang and Lu, 2023). Contemporary PC factories adhere to a procedure where concrete preparation precedes the production process, and it is subsequently transported via specialized cranes to maintain a consistent supply. Additionally, reinforcement is pre-manufactured in advance. Labor resources are also a critical factor, as specific work crews in the factory are responsible only for designated production processes, and crews do not interchange between tasks. Therefore, when creating production schedules, it is necessary to assess the impact of varying crew quantities on production progress. As revealed by the findings of the field study, the PC production process identified in this research comprises six distinct processes: mold assembling (P1), reinforcement setting (P2), concrete casting (P3), curing (P4), mold stripping (P5), and finishing (P6). These processes are graphically represented in the following figure (Figure 2), providing a comprehensive model of the production process from a flow view. In this study, each of the six processes is assigned to a dedicated crew. Upon completing their respective processes, these crews transit to a new

Table 1. Summary of most relevant studies

References	Method	Constraints	Objectives	Parallel production	Pre-trained rescheduling policy
Liu *et al*. (2023)	GA	M_e, H_w	M, E, T	Yes	No
Wang *et al*. (2021)	GA	N_m, H_w	C_r	Yes	No
Wang and Hu (2018)	GA	N_m, H_w	t_i, C_m, C_d, C_r	Yes	No
Du and Li (2024)	DRL-based EA	P_{WE}	E, T, C_e	Yes	No
Kim *et al*. (2020)	DQN	N_m, H_w	T	No	Yes
Proposed Method	*DQN*	N_m, N_l, H_w	*M, E, T*	*Yes*	*Yes*

Note(s): M_e: machine environment constraints; H_w: working hour limits; N_m: number of molds; P_{WE}: total weighted energy price; N_l: number of labors,
M: makespan; E: earliness penalty; T: tardiness penalty; C_r: rescheduling cost; t_i: total idle time; C_m: mold changeover fees; C_d: delivery penalty; C_e: energy cost

Source(s): Author's own work

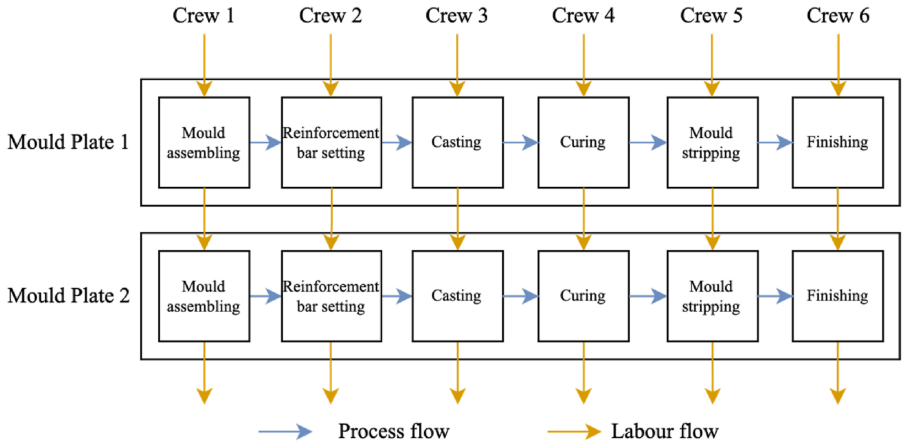


Figure 2. Flow-view production processes. Source: Authors' own work

mold plate and commence new tasks. When preceding processes are accomplished, the subsequent processes are undertaken by new crews. In essence, processes unfold on the same mold plate, with labor transitioning between different mold plates.

As indicated in the research of Ko and Wang (2010), certain PC production processes can be interrupted. The need to define interruptible processes arises from the fact that crews can only work a limited number of hours within a single workday. Once their work hours are exhausted, no new jobs are assigned to the crews. Interruptible processes can be paused after regular work hours and resumed at the beginning of the next workday. In contrast, non-interruptible processes must be completed during overtime hours and cannot be deferred to the following day. The interruptible processes include P1, P2, P5, and P6. Hence, P3 is an uninterruptible process because the casting process must be continuous. Furthermore, P4 is a unique uninterruptible process as it does not require crew assignment and must be performed consistently.

To sum up, the problem studied in this paper can be defined as follows: consider a PC factory equipped with N_m mold plates, each of which is assigned to various groups of PC tasks. Each task comprises of j types of different PC jobs that enters the mold plate at different times. When a job initially enters a mold plate, the crew responsible for P1 is assigned to that mold plate. If no crew is available at that moment, no new job will be allocated. In this study, even jobs that have already been scheduled on the manufacturing process and have completed some subprocesses might still be in the queue, waiting for available crews who are currently engaged in executing other jobs to be released (Figure 3).

3.1 Objectives and constrains

In a typical PCPS, time and cost are the two major concerns. Managing these factors effectively becomes crucial for optimizing the production workflow and ensuring timely deliveries. The objectives of this research then consist of two parts: minimizing the total makespan and the penalties for earliness and tardiness. The relationships among makespan, tardiness, and due dates are illustrated in Figure 4. A reduced makespan not only indicates improved productivity but also ensures that resources are utilized efficiently, ultimately contributing to cost savings and enhanced profitability. Early completion of jobs can be as detrimental as tardiness, especially if it leads to excess inventory costs or disrupts subsequent processes. On the other hand, tardiness in delivering products can result in dissatisfied customers, financial penalties, and damage to the company's reputation. In fact, earliness and

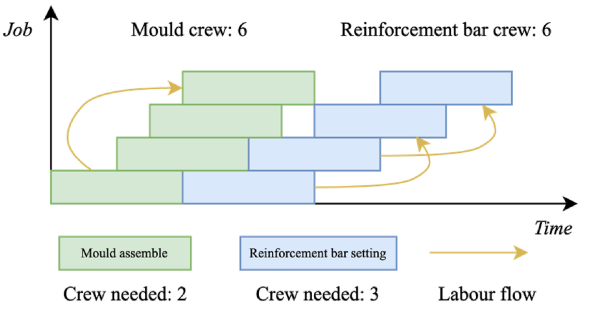


Figure 3. Example job schedule with labor flow. Source: Authors' own work

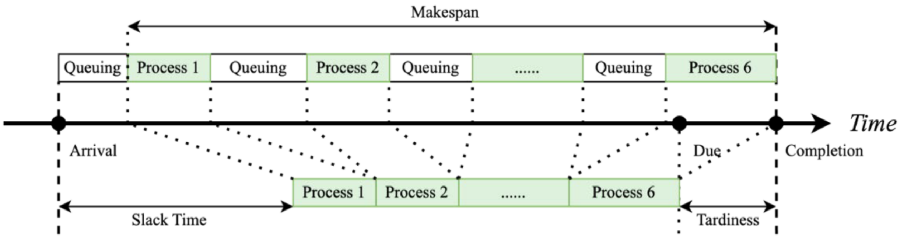


Figure 4. Makespan, tardiness, and due date relationships. Source: Authors' own work

tardiness penalties are conflicting performance indicators. Completing orders too early can lead to increased storage costs, while delaying production, although reducing storage expenses, may result in penalties for breach of contract. Therefore, achieving a balance between these two penalties is essential. The definition of the two objectives is presented in Equations (1) and (2) respectively. The variable explanations are provided in Table 2.

Objectives:

$$\text{Minimize: Makespan} = Max(C_{i,6}) \tag{1}$$

$$\text{Minimize: } \sum_{i=1}^{n} \left(\alpha \cdot \max\left(0, D_i - C_{i,6}\right) + \beta \cdot \max\left(0, C_{i,6} - D_i\right)\right) \tag{2}$$

In the production of precast concrete elements, production efficiency is constrained by various factors (Wang *et al.*, 2023). Researchers have identified many types of constraints in existing

Table 2. Variables in objective functions

Variable	Description	Equation
$C_{i,k}$	Completion time of the process ***k*** of job ***i***	Eq (1) - Eq (5)
D_i	Due date of job ***i***	Eq (2)
α	Penalty factor for earliness (negative value, representing storage cost)	Eq (2) and Eq (16)
β	Penalty factor for tardiness (negative value, representing late delivery cost)	Eq (2) and Eq (16)

Note(s): The penalty factors are derived from field research data collected during visits to PC manufacturing facilities. For different real-world facilities can adopt their own specific values based on individual operational requirements

Source(s): Authors' own work

literatures. In this study, three types of constraints are considered to model the production problem, namely working hour constraints, labor constraints, and resource constraints. Working hour constraints refer to that the production jobs must be processed within specific time during a day (Podolski, 2022; Dan *et al*., 2021; Kim *et al*., 2022). Labor constraints refer to the availability and skill level of the workforce involved in the production of precast concrete elements (Chang and Han, 2021; Podolski, 2022). A limited number of skilled workers mean that there is a finite labor force available to handle the jobs at hand. Resources constraints refer to the availability of raw materials such as concrete, molds, and rebars (Yang *et al*., 2016). In this study, the quantities of concrete and rebar are assumed to be unlimited. The quantity of mold plates is the primary resource constraint. Production scheduling must align with plates' availability, ensuring that the production pipeline is optimized to utilize these resources efficiently. The descriptions of three constraints are given in Equation (3) and (7) respectively and the variables are presented in Table 3.

Subject to:

$$C_{i,k} \geq \begin{cases} T_{i,k} & if\ T_{i,k} \leq 24d + H_W\ and\ k = 1,2,5,6 \\ T_{i,k} + H_N, & if\ T_{i,k} > 24d + H_W\ and\ k = 1,2,5,6 \end{cases} \tag{3}$$

$$C_{i,k} \geq \begin{cases} T_{i,k} & if\ T_{i,k} \leq 24d + H_W\ and\ k = 3 \\ 24(d+1) + P_{i,k}, & if\ T_{i,k} > 24d + H_W\ and\ k = 3 \end{cases} \tag{4}$$

$$C_{i,k} \geq \begin{cases} 24(d+1) & if\ 24d + H_W \leq T_{i,k} \leq 24(d+1)\ and\ k = 4 \\ T_{i,k}, & if\ T_{i,k} < 24d + H_W\ or\ T_{i,k} > 24(d+1)\ and\ k = 4 \end{cases} \tag{5}$$

$$\sum_{n=1}^{m} L_{n,k} \leq \mathcal{L}_k \tag{6}$$

$$N_m \leq N_M \tag{7}$$

3.2 Assumptions

To simulate the PC production processes, several necessary assumptions have been made. These assumptions guarantee that the simulation of the production process reflects the real production situation as accurately as possible, while appropriate simplifications allow the problem to be modeled and solved more effectively.

(1) Each production step is operated by specialized crews.

(2) Crews can only transfer to next mold plate after their current step completed.

Table 3. Variables in subject-to function

Variable	Description	Equation
$T_{i,k}$	The estimated completion time without interruption of the process k of job i	Eq (3) - Eq (5)
d	Current day count	Eq (3) - Eq (5)
H_W	Daily working hours	Eq (3) - Eq (5)
H_N	Daily non-working hours	Eq (3) - Eq (5)
$P_{i,k}$	Processing time of the process k of job i	Eq (3) - Eq (5)
$\mathcal{L}_k$	Total number of crews responsible for process k	Eq (6)
L_n	Current working crew executing the process k on the mold plate n	Eq (6)
N_m	Numbers of current occupied mold plates	Eq (7)
N_M	Total number of available mold plates	Eq (7) and Eq (10)

Source(s): Authors' own work

(3) Production steps are non-interruptible.

(4) There is no transfer time for crews between different mold plates.

(5) There is no idle time between different production steps.

(6) Concrete casting on different mold plates can be performed simultaneously.

(7) The quantities of crews and mold plates are limited but renewable.

(8) Mold plates are unordered, meaning that selecting a specific mold plate does not impact the execution of the order.

4. Method

4.1 DRL architecture

The proposed DRL architecture is depicted in Figure 5. The original PC order information includes quantities and due dates for each type of PC elements The initial PC order data, which includes the quantities and due dates for each type of PC element, is combined with information about the PC facilities, such as the number of molds and crews, to set up the production environment. Meanwhile, an initial action space that includes all possible job combinations is then established regarding the input information. The size of the initial action space is determined only by the number of mold plates and types of PC elements. The production environment outputs the current state, including mold utilization, crew utilization, order progress, and ongoing job progress. This state is processed alongside the initial action space through a valid action masking algorithm, which filters out any actions that are not executable under the current state conditions. In this study, DQN is chosen as the backbone algorithm. As the state S_t is designed as a two-dimensional matrix, so it is appropriate to use a

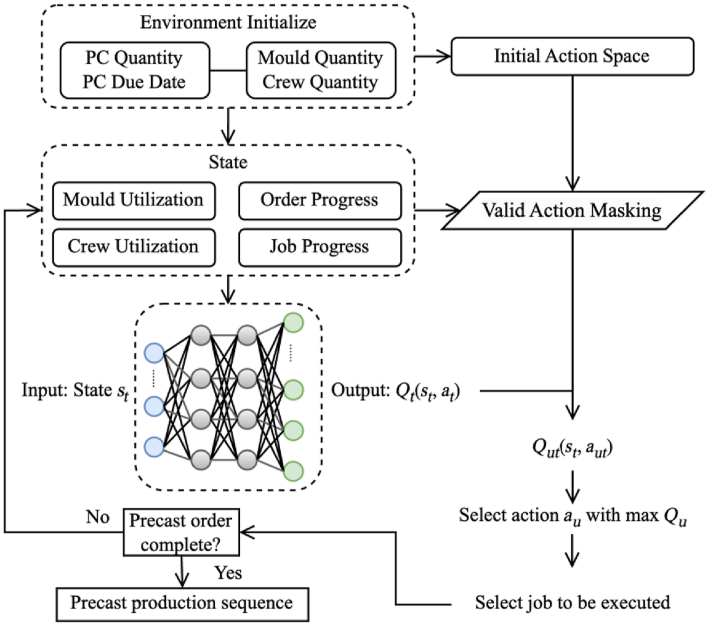


Figure 5. DQN architecture. Source: Authors' own work

two-layer convolution neural network Convolutional Neural Network (CNN) which outputs a value $Q_t(S_t, a_t)$. For each possible action a_t. Then the value Q_t will be refined as Q_{ut} for valid actions a_{ut}. The action with the highest Q-value is selected to be executed at that decision point. The chosen action represents the job that the DRL has selected to perform at that moment. Subsequently, the selected job is added to the execution sequence, and the environment state is updated accordingly for the next decision point. The proposed DRL architecture differs from previous research (2022) in three main aspects: (1) to enhance the generalization ability, this study uses CNN to extract hidden features from the state tensor rather than using scalar values; (2) this study considers the complexity of parallel production lines and PC types in action design, allowing for arrange multiple jobs to be arranged and processed simultaneously; and (3) this study adopted a reward shaping mechanism for optimizing multiple objectives. The key variables used in the DRL model are defined and explained in Table 4. The neural network diagram is presented in Figure 6. The agent-environment flow chart is presented in Appendix 2.

4.2 DRL design

To train a valid DRL agent, the state, action, reward, and environment need to be designed. In reinforcement learning, the environment refers to the external system or context with which an agent interacts and learns through feedback in the form of rewards, shaping the agent's decision-making process. In this study, such interaction is simulated using Python. Another role of the environment is to identify validate executable actions at each decision point and provide this information to the agent, a process known as action masking, which will be explained in the next section.

4.2.1 State. In DRL, the final policy is derived from the Markov Decision Process, so the actions of the agent are determined solely by the current state. For the PCPS problem, the state needs to represent the details of the current production environment, including manpower status, mold plate utilization, overall job progress, and so on. Additionally, the state should

Table 4. Variables in DRL model

Variable	Description	Equation
S_t	Current state of the environment at time t, including mold and crew utilization, and job progress	Eq (8)
a_t	Selected action at decision point t, representing job allocation to mold plates	Eq (9)
a_{ut}	Valid action at decision point t	–
$Q_t(S_t, a_t)$	Estimated Q-value of taking action a_t in state S_t	Eq (12)
Q_{ut}	Refined estimated value after action masking	–
s_m	The information about the job currently being executed on the mold plate m	Eq (8)
O_m	One-hot encoding. For each type of PC jobs, a unique vector is designed to identify which type of job is currently under execution on mold plate m	Eq (8)
P	Order completion progress $P \in \mathbb{R}$, representing the current percentage of jobs completed or being executed for a specific type of PC order, consists of amount percentage and relative due dates	Eq (8)
I	Job completion progress, representing the completion progress of various processes for jobs currently being executed	Eq (8)
$\widehat{A}$	Action space	Eq.(11)
n_j	Selecting n orders from the j^{th} type of PC jobs	Eq.(9)
ε	The coefficiency of ε – *greedy* policy	Eq.(12)
$\varphi(S_t, a_t)$	The action masking function at decision point t	Eq (14)Eq (13)
$\widehat{A_\varphi}(S_t)$	Masked action space at decision point t	–
RW	Reward received after taking action	Eq (14) - Eq (17)

Source(s): Authors' own work

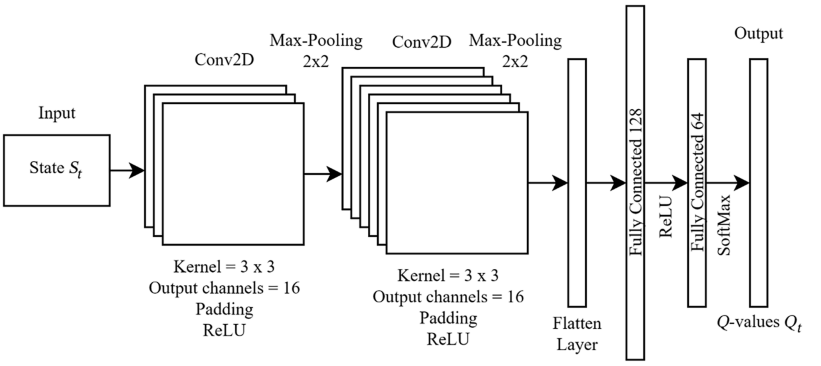


Figure 6. DQN neural network diagram. Source: Authors' own work

exhibit distinct and clear transitions for different actions. This is because when state changes are not apparent, the agent struggles to extract sufficient features from subtle variations, making it difficult for DRL training to converge.

To meet the actual concerns in PC facilities, this study designed a mixed state S_t to combine various types pf facility and production line information. s_m represents the information about the job currently being executed on the m^{th} mold plates.

$$S_t = \begin{bmatrix} s_1 \\ \vdots \\ s_m \end{bmatrix} = \begin{bmatrix} [O_1, P_1, I_1] \\ \vdots \\ [O_m, P_m, I_m] \end{bmatrix} \tag{8}$$

4.2.2 Action. Action space $\widehat{A}$ is a set of all the possible activity combinations. Instead of taking different DPs as actions Kim *et al*. (2022), in this study, the design of the action space primarily considers two aspects: the number of mold plates and the quantities of PC component types. The state transition when selecting an action is explained in Appendix 3. For a batch of PC orders consisting of j different types, a common approach in designing the action space is as follows: each mold plate can have j possible job allocations; therefore, in the case of m mold plates, the size of the action space becomes j^m. However, when the number of PC types in the orders is large and the number of mold plates is significant, the size of the action space becomes exceedingly large. This combinatorial explosion makes it difficult for the agent to converge during training.

This paper introduces a practical method for representing the action space, significantly reducing the search complexity for the agent during the training process. Instead of designing the action from a simplistic random allocation perspective, this study comprehensively utilizes two types of information: the total number of mold plates and the variety of job types. The action vector is defined using the following formula Equation (9) and (10):

$$a_t = [n_1, n_2, \ldots, n_j] \tag{9}$$

$$0 \leq \sum_{i=i}^{j} (a_i \, in \, a_t) \leq N_m \tag{10}$$

The job selection subjects to the total quantity of mold plates m. Thus, the size of the action space becomes Equation.(11) which is smaller than j^m when j and m are large:

$$len(\widehat{A}) = \sum_{n=1}^{j+m-3} \binom{j+m-n}{2} \tag{11}$$

At each time step, the DRL agent chooses action a_t from a fixed action space $\widehat{A}$. This selection method is known as behavior policy. In the adopted DQN method, an $\varepsilon - greedy$ policy is applied (Equation.12).

$$a_t = \begin{cases} \underset{\widehat{a}}{\text{argmax}} Q(S_t, a_t), & in\ the\ probablity\ of\ 1-\varepsilon \\ random, & in\ the\ probablity\ of\ \varepsilon \end{cases} \tag{12}$$

The $\varepsilon - greedy$ policy tends to guide the agent to select the action which gives the maximum reward in the time step. At the beginning of the agent training, ε is set to be a larger value to let the agent explore the action space in a random pattern. This policy prevents the agent to be trapped in local optimization and helps the neural networks have better generalization ability.

However, even within a narrowed action space, certain actions may still lead to constraint violations. Typically, such violations result in an early stop of the training episode, thereby requiring the agent to undergo more episodes to converge, consequently diminishing training efficiency. To address this issue, action masking (Huang and Ontañón, 2020) is employed. At each timestep, the mask provides a set of legally safe actions for the DQN agent based on the environmental state $\boldsymbol{S_t}$.

To mask the invalid actions, a masking function is introduced (Equation.13):

$$\varphi(S_t, a_t) = \begin{cases} 1, & if\ (S_t, a_t)\ is\ verified\ safe \\ 0, & otherwise \end{cases} \tag{13}$$

The masked action set $\widehat{A_\varphi}(S_t) = \{a_t | \varphi(S_t, a_t) = 1\}$ is defined for the environment state S_t. After the action masking, the agent will only choose valid actions from $\widehat{A_\varphi}$ under the $\varepsilon - greedy$ policy in Equation.(12).

4.2.3 Reward design. A reward is the feedback obtained by the agent after interacting with the environment. The definition of the reward function determines the optimization direction of the DQN model. In this study, reward shaping is adopted to enhance the training efficiency (Hu *et al.*, 2020). The reward consists of three parts: reward that marks the completion of PC orders RW_1, reward that indicates makespan RW_2, and reward that indicates earliness and tardiness RW_3 (Equation.(14) - Equation.(17)).

$$RW_1 = \begin{cases} 20000 & , if\ order\ completed \\ 0 & , if\ order\ not\ completed \end{cases} \tag{14}$$

$$RW_2 = k \tag{15}$$

$$RW_3 = \sum\nolimits_{i=1}^{c} \left(\alpha \cdot \max\left(0, D_i - C_{i,6}\right) + \beta \cdot \max\left(0, C_{i,6} - D_i\right)\right) \tag{16}$$

$$RW = RW_1 + RW_2 + RW_3 \tag{17}$$

Equation.(14) represents the end reward for completing one episode. Equation.(15) represents a penalty mechanism where k denotes a negative real number. This negative reward aims to shorten the total makespan by implying negative rewards. This reward encourages the agent to finish the scheduling completely as early as possible. The value of k is determined by a trial-and-error process. In this study, integers between $[-50, 0]$ were tested, and the results indicated that training efficiency was optimal when when $k = -20$. It should be noted that the value of k is not determined and can be vary for different cases. The trial-and-error approach is widely

accepted in reinforcement learning studies (Silver *et al*., 2021; Booth *et al*., 2023). Equation.(16) indicates the reward for earliness and tardiness of the completed job in the timestep. This reward helps the agent to better balance the two opposite penalties. Equation.(17) represents the total reward.

5. Case studies

5.1 Case description

5.1.1 Case study 1. The cases in this research are derived from field studies conducted at PC facilities in Jiangxi Province, China. Essentially, these PC facilities operate under a make-to-order production model, indicating that production only commences once the corresponding orders are received. The range of prefabricated components produced by these PC facilities is diverse, including floor slabs, wall panels, stairs, and external facade decorative panels, among other PC components. The surveyed PC facilities have both fixed production areas and flow shop production areas. Fixed production areas are typically used for manufacturing medium-sized PC components, whereas flow shop areas are mainly dedicated to producing larger structural elements. This research focuses on fixed-location production as primary case study.

The scheduling tasks in this study are based on the real PC orders that provided by the surveyed factory. The basic configurations of the PC facility are displayed in Table 5 and Table 6. In Case study 1, three distinct types of PC components (namely wall panel 0, wall panel 1, and roof panel 2) are produced in various quantities. The facility contains six fixed mold plates, and the daily working time is set to 8 hours, leaving 16 non-working hours. The penalty factors for earliness and tardiness (α and β in Equation (16)) are set to -0.5 and -10 respectively according to the real contracts.

Given that the surveyed facility possesses a large open-air space for storing orders completed ahead of schedule, the penalty for earliness is comparatively less severe than the penalty incurred for breach of contract resulting from delayed production. In the factory, the number of specialized skill crews is limited, which means that in some cases, even with sufficient workstations, some PC orders cannot be executed due to insufficient labor availability. In such cases, it becomes essential to carefully schedule the execution sequence of different order types to ensure resource efficiency. In this study, each PC production process is handled by a distinct crew, and the number of crews is diverse and restricted. The order specifics that provided by the PC facility are outlined in Table 7.

Table 5. PC facility configurations

Basic configuration			
H_w	Working hour	8	Eq.18 - Eq.19
N_M	Number of Mold plates	6	Eq.20
α	Earliness penalty	-0.5	Eq (16)
β	Tardiness penalty	-10	Eq (16)

Source(s): Authors' own work

Table 6. Facility crew configurations

Processes	P1	P2	P3	P4	P5	P6
Crew Quantity	6	6	4	0	6	6

Source(s): Authors' own work

Table 7. Task details of case study 1

PC code	Quantity Task 1	Task 2	Task 3	Due date (h) Task 1	Task 2	Task 3	Process time (h) / crew requirement (shift) P1	P2	P3	P4	P5	P6
0	15	26	11	240	480	240	4/2	4/3	1/2	8/0	1/2	1/2
1	20	23	19	360	456	312	3/4	3/3	1/2	8/0	1/3	1/2
2	17	18	19	168	432	336	3/2	1/1	1/2	8/0	1/1	1/1

Source(s): Authors' own work

The ability of the pre-trained agent to effectively manage different upcoming PC orders plays a pivotal role in ensuring the seamless operation of manufacturing processes. To assess the agent's robustness and adaptability under diverse order conditions, Task 1 is used to train the DQN model, while two rescheduling tasks, namely Rescheduling Task 2 and Rescheduling Task 3 (as detailed in Table 7), are adopted and implemented for evaluation. Two rescheduling tasks present more intricate challenges to the agent. In both cases, the due dates and job quantities differ significantly from those in the original task. This variation reflects the complexity often encountered in practical manufacturing environments, where projects may vary significantly in scope and urgency. Such dynamic changes demand an agent capable of rapidly adapting its scheduling strategies to meet the unique demands of each order, ensuring timely delivery and optimal resource utilization.

It is worth emphasizing that in surveyed factories, most incoming orders predominantly revolve around specific types of PC components. This trend underscores the critical importance of the agent's proficiency in efficiently scheduling future orders of recurring PC component type. The rescheduling tasks provided by the surveyed facility, differing in due dates and job quantities, reflect the real challenges faced in the industry. The fluctuations in due dates, a common occurrence in real-world production, necessitate an agent that can make real-time adjustments to its scheduling algorithms. As off-site construction heavily relies on a multitude of unpredictable factors, the pre-trained agent's ability to dynamically respond to shifting timelines is important. Simultaneously, the quantities of PC orders are intricately related to the scale and complexity of the corresponding projects. Hence, the agent's ability to handle different order quantities is equally important.

5.1.2 Case study 2. In addition to the large PC components, the surveyed factory also kept production records for some smaller prefabricated components. These smaller components typically require only a few workers and can be completed within a shorter period. However, they are subject to more stringent delivery requirements, often within one week. Therefore, careful scheduling of the production sequence is necessary to avoid incurring substantial penalties for breach of contract. Table 8 presents a portion of the order, which includes 10 different PC elements intended for the construction of prefabricated townhouses. PC0–PC3 refer to decorate panels, PC4–PC7 refer to thin wall panels, and PC8–PC9 refer to thin roof panels. Additionally, two rescheduling tasks are also provided, each with different quantities and due dates compared to the original order. Similar to case study 1, Task 4 is used to train the DQN model, while Task 5 and Task 6 are the rescheduling tasks with different quantities and due dates.

5.2 Model implementation

5.2.1 DQN training settings. The proposed DQN consists of a two-layer CNN to extract features from the state S_t. The proposed CNN architecture comprises two convolutional layers followed by a fully connected layer. The first convolutional layer, with a 3×3 kernel and ReLU activation, processes input data to produce 16 output channels, followed by max-pooling. The second convolutional layer takes the 16-channel output, applies a similar

Table 8. Task details of case study 2

PC code	Quantity Task 4	Task 5	Task 6	Due date (h) Task 4	Task 5	Task 6	Process time (h) / crew requirement (shift) P1	P2	P3	P4	P5	P6
0	3	5	4	48	48	60	1.5/2	2.0/3	0.5/2	8.0/0	1.0/2	0.5/1
1	3	7	5	48	48	72	1.0/2	2.0/3	0.4/2	8.0/0	1.0/2	0.5/1
2	3	3	3	48	48	60	1.0/2	1.5/2	0.5/2	8.0/0	0.5/1	0.5/1
3	3	2	2	36	36	36	0.5/1	1.0/2	0.3/2	8.0/0	0.3/1	0.5/1
4	3	4	5	36	36	72	1.0/2	0.8/1	1.0/3	8.0/0	1.5/3	0.5/1
5	3	6	3	36	36	48	0.5/1	2.0/3	0.4/2	8.0/0	0.5/1	0.5/1
6	3	1	2	72	72	72	1.5/2	2.0/3	0.5/3	8.0/0	1.0/2	0.4/1
7	3	2	3	72	72	48	0.5/1	2.0/3	0.3/2	8.0/0	0.6/1	0.3/1
8	3	2	2	36	36	60	1.5/2	1.8/2	1.2/3	8.0/0	1.5/3	1.5/2
9	3	1	2	72	72	48	0.4/1	0.5/1	0.6/2	8.0/0	0.5/1	0.2/1

Source(s): Authors' own work

operation with a 3×3 kernel, and produces 32 output channels. ReLU activation and max-pooling are again applied. The final layer consists of two linear layers: the first with ReLU activation and the second with Softmax activation. These layers transform the output into a tensor representing the Q-values of all actions. In the DQN framework, the input state tensor dimensions are 10×6 for case study 1 and 17×6 for case study 2, representing different configurations of production status information. The output layer produces a Q-value vector of length 83 for case study 1 and 559 for case study 2, respectively, corresponding to the number of actions in each case. To train the DQN agent, the batch size is set to 256, update frequency is set to 3, with a 10,000-replay buffer size. The training was carried out with ϵ-greedy policy at an exploration factor of 0.95 and linearly decreasing to 0.01 at the end of the episode. Adam optimizer is adopted, and L2 Normalization is introduced during the network training. The hyperparameters are determined by Grid Search. The initial search space for hyperparameters was based on recommended values from the open-source project Stable Baselines3 (Raffin *et al*., 2021). The final hyperparameters were determined as presented in the manuscript. Through the model training upon different case studies, the DQN provides optimal convergences. The training data was derived from real-world production scenarios collected during field visits to PC factories. Tasks 1 and 4 were used to train the DQN model, while other tasks were reserved for testing the generalization capability. Table 7 and Table 8 present the specific task configurations used in training and evaluation. The DQN model is trained on a personal computer with an i7-11700 k chip and an RTX 3080. The interpreter is based on Python 3.9.0, PyTorch 1.12.1, and Numpy 1.23.4. Task 1 and Task 4 are used to train the DQN agent, while the other tasks are rescheduling task using the pre-trained agent. The parameter settings are listed in Appendix 4.

5.2.2 Comparison algorithms. The results were compared with traditional scheduling methods, including two classic DPs and a simulation-based GA. Selected DPs are EDD and SPT. EDD prioritizes jobs based on the EDD, while SPT prioritizes those with the shortest processing time. Due to the different cases and assumptions made in current research, existing algorithm in the field of PCPS cannot be directly applied in this study. To tackle the gap, a simulation-based GA is proposed in this study. The simulation-based GA consists of a modified GA (Liu *et al*., 2020) and a simulation environment. Adopted baseline GA has been validated by standard benchmark problems of size J30, J60, and J120 from Project Scheduling Problem Library, which is a well-known scheduling problem library created by Kolisch (1997). It has also been used for comparison in recent published papers (Hua *et al*., 2022; Peng and Zheng, 2023; Martin *et al*., 2024). In this study, each gene represents a PC element, with the order of the genes indicating the sequence in which these elements are arranged on the

mold plates for execution. Given that the research focus is on parallel production problems, the simulation environment proposed here serves as the chromosome decoder for the GA. The simulation decoder ensures that at every moment, resources and mold plates are fully utilized. The adopted algorithm uses regret-based biased random sampling and serial schedule generation scheme to generate initial population. Evolution strategy consists of elite selection mechanism, two-point crossover, swap mutation, and linear decreasing probability-based mutation. The parameter settings of adopted GA are shown in Table 9. For the EDD and SPT DPs, the generation of schedules also relies on the simulation decoder, which directly decodes job sequences that conform to the EDD and SPT rules.

5.3 Results

5.3.1 DQN training. The training of DQN is illustrated in Figure 7–9. These figures display the smoothed reward curve, indicating that the agent stabilizes around the 90th episode. After this point, continued training does not significantly improve the agent's scheduling performance and training reward. Therefore, it can be concluded that the DQN has achieved the training objectives at this stage.

Figure 8 illustrates the tardiness and earliness curves during the DQN training process. As the training progresses, the penalties associated with earliness and tardiness exhibit distinct

Table 9. GA parameters

Element	Unit
Population size	200
Crossover Rate	0.8
Mutation Rate	0.2
TOP	0.15

Source(s): Authors' own work

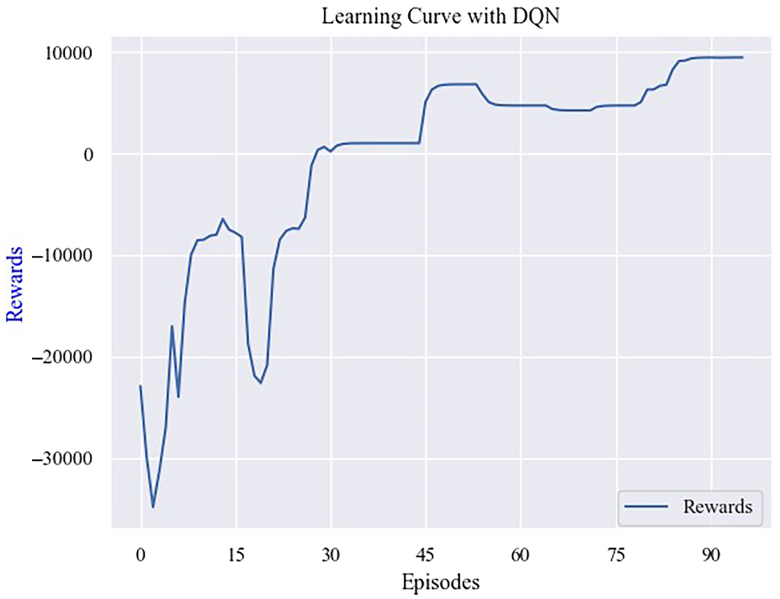


Figure 7. Reward curve. Authors' own work

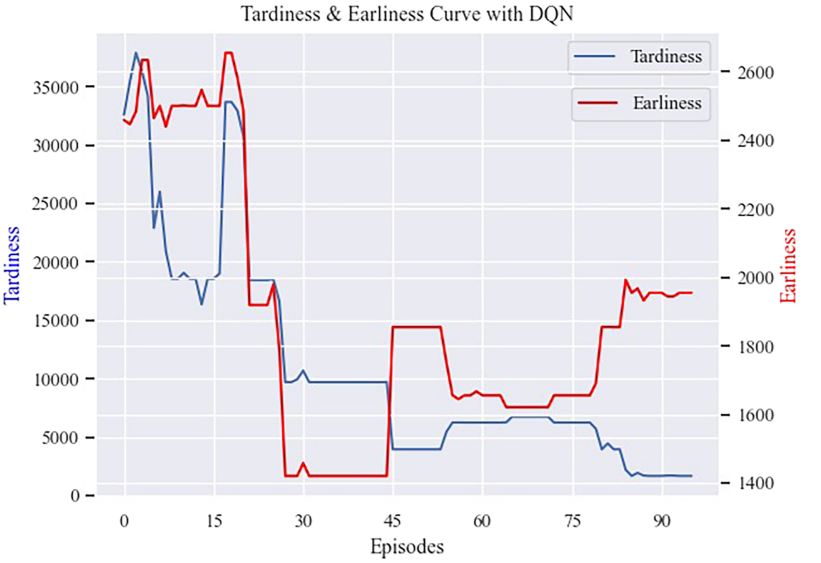


Figure 8. Earliness and tardiness curves. Source: Authors' own work

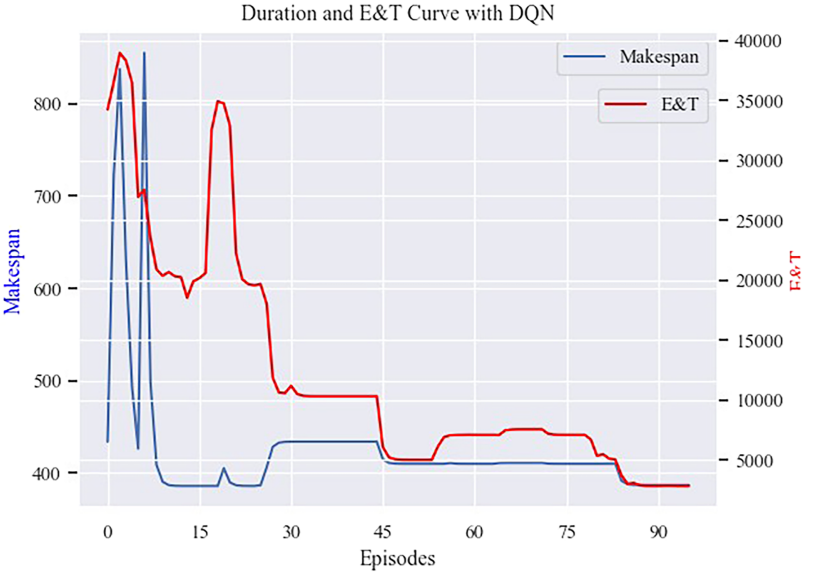


Figure 9. Makespan and earliness and tardiness curves. Source: Authors' own work

trends. Specifically, while the earliness penalty increases, the tardiness penalty gradually decreases. Additionally, Figure 8 presents the evolution of the makespan and the sum of the two penalties throughout the training. As training nears completion, both the makespan and the total penalty converge toward optimal values, demonstrating the agent's ability to balance timely completion with efficient resource utilization.

5.3.2 Results comparison on case study 1. Table 10 presents the scheduling and rescheduling outcomes of both traditional methods and the proposed DQN approach. EDD method is chosen as the baseline as its wide usage in the surveyed real-world factories. The gaps of other methods with the baseline are calculated by Equation (21). Where p_i represents the results of three other methods and p_{edd} represents the results of EDD.

$$g = \left(1 - \frac{p_i}{p_{edd}}\right) \times 100\% \tag{21}$$

In Task 1, the results demonstrate that DQN achieves the optimal schedule with reduced total penalties and a shorter makespan. EDD schedules the task with smallest earliness penalty, but the makespan is longer than DQN method. Compared to DQN, GA provides a better earliness penalties; however, it incurs higher tardiness penalties. Among the four evaluated methods, SPT exhibits the poorest performance, with the highest total penalty. The CPU times of the methods show that although GA is capable of generating suboptimal solutions superior to traditional DPs, its processing time is significantly longer than that of the DQN model.

In both rescheduling tasks, DQN and GA outperform traditional methods (EDD and SPT) in terms of tardiness penalty reduction and makespan minimization. In Task 2, GA achieves the lowest earliness penalty, yet its tardiness and total penalties are larger than DQN results. In Task 3, DQN outperforms GA with less total penalty. And the process times of GA remains longer than the pre-trained DQN model. With apparent shorter time, the same pre-trained DQN can generate near-optimal solutions under different task settings.

5.3.3 Results comparison on case study 2. Table 11 presents the findings from case study 2. In Task 4, the DQN method produces the most efficient schedule, characterized by the lowest total penalty, shortest makespan, and reduced CPU time. Compared to the EDD method, the DQN achieves a 15.8% reduction in total penalty and a 14% reduction in makespan, while also necessitating less CPU time. Regarding the two rescheduling tasks, the pre-trained DQN

Table 10. Scheduling results of case study 1

Task		EDD	SPT	GA	DQN
Task 1 (Training)	Earliness ($)	*1944*	2,469	2003.5	2,122
	Tardiness ($)	1,330	17,110	890	*590*
	Total Penalty ($)	3,274	19,579	2,893.5	*2,712*
	Penalty Gap	/	−498.01%	11.62%	*17.17%*
	Makespan (h)	410	410	386	*386*
	Makespan Gap	/	0%	6%	*6%*
	CPU time (s)	0.37	0.35	168.73	*0.24*
Task 2 (Rescheduling)	Earliness ($)	6,837	6,837	*6,668*	7,336
	Tardiness ($)	1,610	1,610	1,120	*320*
	Total Penalty ($)	8,447	8,447	7,788	*7,656*
	Penalty Gap	/	0.00%	7.80%	*9.36%*
	Makespan (h)	530	530	506	*506*
	Makespan Gap	/	0%	5%	*5%*
	CPU time (s)	0.47	0.41	197.65	*0.32*
Task 3 (Rescheduling)	Earliness ($)	*2011*	3,572	2,357.5	2,392.5
	Tardiness ($)	2,400	10,790	1,410	*1,150*
	Total Penalty ($)	4,411	14,362	3,767.5	*3,542.5*
	Penalty Gap	/	−225.60%	14.59%	*19.69%*
	Makespan (h)	386	386	362	*362*
	Makespan Gap	/	0%	6%	*6%*
	CPU time (s)	0.53	0.55	183.71	*0.23*

Source(s): Authors' own work

Table 11. Scheduling results of case study 2

Task		EDD	SPT	GA	DQN
Task 4 (Training)	Earliness ($)	*30.3*	122.4	68.05	40.65
	Tardiness ($)	9,241	9,632	8,315	*7,766*
	Total Penalty ($)	9,271.3	9,754.4	8,383.05	*7,806.65*
	Penalty Gap	/	−5.21%	9.58%	*15.80%*
	Makespan (h)	144.7	145.5	124.0	*124.0*
	Makespan Gap	/	−1%	14%	*14%*
	CPU time (s)	8.49	8.59	671.54	*5.21*
Task 5 (Rescheduling)	Earliness ($)	*35.45*	87.45	50.65	37.75
	Tardiness ($)	13,124	13,204	12,336	*12,085*
	Total Penalty ($)	13,159.45	13,291.45	12,386.65	*12,122.75*
	Penalty Gap	/	−1.00%	5.87%	*7.88%*
	Makespan (h)	145.5	147.0	145.5	*145.5*
	Makespan Gap	/	−1%	0%	*0%*
	CPU time (s)	8.36	8.91	716.5	*5.77*
Task 6 (Rescheduling)	Earliness ($)	*67.5*	68.6	78.95	84.2
	Tardiness ($)	6,775	6,802	*6,529*	6,632
	Total Penalty ($)	6,842.5	6,870.6	*6,607.95*	6,716.2
	Penalty Gap	/	−0.41%	*3.43%*	1.85%
	Makespan (h)	145.4	148.0	146.5	*145.5*
	Makespan Gap	/	−2%	−1%	*0%*
	CPU time (s)	8.37	8.65	655.44	*5.71*

Source(s): Authors' own work

continues to outperform both traditional DPs and the GA in Task 5. In Task 6, although DQN does not generate the best schedule among the four methods, it still produces a schedule comparable to GA in the least amount of time.

6. Discussion and limitation

To enhance transparency and stakeholder acceptance, this study provides visual and quantitative illustrations of the model validation process through comparative analyses across multiple real-world-inspired tasks. The validation was carried out by applying the pre-trained DQN model to both the original training scenarios and a series of rescheduling tasks with varying order quantities and due dates. As shown in Table 10 and Table 11, the DQN's performance was benchmarked against traditional DPs (EDD and SPT) and a simulation-based GA. The results demonstrate the model's ability to generalize beyond the training task, maintaining competitive performance in minimizing penalties and makespan across different task configurations. Detailed performance curves (Figures 7–9) further illustrate the convergence behavior of the model, showing how the agent progressively balances earliness, tardiness, and makespan during training. These multi-level validations strengthen the credibility of the proposed approach and confirm its practical value for real-world precast production environments.

The results obtained in both training and rescheduling tasks illustrate the strengths and implications of the proposed DRL-based method. The use of a DQN framework with a simplified action space enables the agent to make decisions that are both optimized and computationally efficient. Unlike traditional algorithms that require full recomputation for every new task, the pre-trained DRL policy leverages learned scheduling strategies, allowing for rapid rescheduling across orders with different due dates and quantities. This adaptability is particularly valuable in real-world factory environments where conditions frequently change. The integration of action masking further improves efficiency by masking infeasible actions, enabling the model to focus on valid scheduling paths and contributing to shorter convergence

times and lower penalty costs. These results highlight the effectiveness of the DRL approach not only in generating optimized schedules but also in delivering practical benefits such as speed, flexibility, and reusability.

6.1 Discussion on the algorithm performance

The performance of the proposed DQN method is compared with a simulation-based GA and two established classic DPs. The evaluation is conducted across three dimensions: (1) total penalties; (2) makespan; and (3) CPU time. The penalty and makespan serve as indicators of the algorithm's effectiveness, while CPU time reflects its efficiency. The results demonstrate that the DQN consistently generated the most optimal production schedules on the tasks for training (Task 1 and Task 4), achieving lower total penalties and makespan. In the rescheduling tasks using the pre-trained DQN agents (Task 2, Task 3, Task 5, and Task 6), the DQN outperforms the baseline in producing superior production schedules. Such generalization ability comes from the state and action design. In this study, the state representation does not directly encode the specific quantities and due dates of the PC elements. Instead, the state focuses solely on the real-time job progress on the fixed mold plates. Therefore, when rescheduling orders of the same type, the pre-trained DQN agent is not affected by order quantities and deadlines. Similarly, the size and structure of the action space are not linked to the specific quantities of PC elements, so the DQN agent's action selection remains stable across varying order volumes. However, such design has limitations under complex rescheduling orders and diverse due dates such as Task 6. Thus, the performance of DQN is slightly inferior to that of GA.

Although the DQN does not produce a better schedule than GA for Task 6, its outcome remains better than the baseline heuristics. This phenomenon can be attributed to the differences in optimization mechanisms, adaptability, and generalization capability between the two methods. Specifically, GA employs a population-based global search with inherent exploration and diversification mechanisms, enabling it to better adapt to task-specific penalty structures—such as the dominance of tardiness costs observed in Task 6—by effectively minimizing total penalties. In contrast, the pre-trained DQN relies on value functions learned from the training environment, which may limit its adaptability when facing states that deviate from the training distribution. While this may result in slightly higher penalties compared to GA, DQN still achieves the shortest makespan and significantly outperforms GA in computational efficiency, demonstrating its scalability and suitability for real-time rescheduling. Importantly, the rescheduling results further show that DQN is capable of dynamically adapting to changes in precast element quantities and due dates, highlighting its practical value in flexible and responsive production-construction coordination.

From a CPU time perspective, the DQN outperforms both classic DPs and GA in schedule generation time. DQN's fast generation comes from the pre-trained policy, which enables the DQN to directly select PC elements with the highest Q-values without iterative computation. Meanwhile, the proposed simulation-based GA outperforms the baseline in all six task scenarios, suggesting that the improved GA can be effectively applied to parallel production scheduling. However, a notable limitation of GA lies in its inability to be reused across different tasks without re-iteration. Each new scheduling problem requires the GA to undergo a full optimization cycle, resulting in significantly longer CPU times compared to both DQN and dispatching-rule-based approaches. In two case studies, there are clear gaps in the schedule generation time. There are two reasons for such time diversity. First, in the case study 1, the process times for each type of PC elements are recorded as integer. Consequently, it's enough to set each timestep in the simulation to equal one real hour. In contrast, for case study 2, where the recorded process times are precise to 0.1 hour, the simulation must be accordingly configured to reflect this finer granularity. A comparative analysis of the schedules from both cases reveals that the number of steps required in the simulation environment for the first case is significantly fewer than those needed for the second case. As a result, the CPU time to

generate optimal schedules for the second case will be longer. Second, when the simulation environment functions as a chromosome decoder, it necessitates additional computational costs. At each timestep, it must search the entire action space and select actions that align with the gene sequence. Since the case study 1 has fewer element types compared to case study 2, and the two DPs used in this study need the simulation to decode and generate the final schedule, an increase in the variety of elements leads to a larger action space, which lowers the schedule generation efficiency. However, pre-trained DQN agent does not need to search the entire action space, the neural network could select the action with highest Q-value. Consequently, DQN require less time to generate optimal schedule plans.

6.2 Discussion on the generated schedules

The schedules generated by the DPs and proposed methods are presented in Table 12. The numbers in each sequence represent the code of the PC element in the order. For the task with same PC types, the job sequence given by EDD changes according to different due dates, whereas the job sequence determined by SPT depends only on the production time required by different types of elements, making the SPT sequence constant. The schedule plans provided by DQN and GA will also redefine the manufacturing order based on the specific demands of the orders. In contrast, the schedules produced by DQN and GA dynamically adjust the job sequence based on the specific requirements of each order. Unlike DPs, DQN and GA do not

Table 12. Production sequence of case studies

Task	Method	Sequence
Task 1	EDD	[2, . . . 2, 0, . . . 0, 1, . . .1]
	SPT	[2, . . . 2, 1, . . . 1, 0, . . .0]
	GA	[1, 0, 2, 2, 2, 0, 2, 2, 0, 2, 2, 0, 0, 0, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
	DQN	[2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 0, 2, 2, 2, 2, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Task 2	EDD	[2, . . . 2, 1, . . . 1, 0, . . .0]
	SPT	[2, . . . 2, 1, . . . 1, 0, . . .0]
	GA	[1, 2, 0, 1, 0, 1, 0, 1, 2, 1, 1, 0, 2, 2, 0, 0, 1, 2, 0, 0, 1, 2, 2, 0, 0, 0, 1, 2, 2, 2, 0, 0, 0, 1, 1, 1, 2, 1, 2, 1, 2, 2, 0, 0, 1, 0, 1, 0, 0, 1, 2, 2, 0, 0, 1, 2, 1, 1, 1, 1, 0, 1, 0, 0, 2, 0, 0]
	DQN	[2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 1, 2, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Task 3	EDD	[0, . . . 0, 1, . . . 1, 2, . . .2]
	SPT	[2, . . . 2, 1, . . . 1, 0, . . .0]
	GA	[1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 2, 2, 0, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2]
	DQN	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1, 2, 2, 2, 2, 2, 2]
Task 4	EDD	[3, . . . 4, . . . 5, . . . 8, . . . 0, . . . 1, . . . 2, . . . 6, . . . 7, . . . 9, . . .]
	SPT	[9, . . . 3, . . . 7, . . . 5, . . . 2, . . . 1, . . ., 6, . . . 4, . . . 0, . . . 8, . . .]
	GA	[9, 3, 5, 8, 5, 7, 1, 5, 0, 3, 3, 1, 9, 7, 9, 2, 1, 6, 6, 4, 2, 6, 8, 2, 4, 0, 8, 0, 7, 4]
	DQN	[5, 8, 4, 5, 4, 3, 2, 5, 6, 4, 1, 0, 7, 7, 6, 0, 0, 2, 9, 7, 1, 9, 3, 9, 3, 2, 8, 8, 1, 6]
Task 5	EDD	[3, . . . 4, . . . 5, . . . 8, . . . 0, . . . 1, . . . 2, . . . 6, 7 . . . 9]
	SPT	[9, 3, . . . 7, . . . 5, . . . 2, . . . 1, . . . 4, . . . 6, 0, . . . 8, . . .]
	GA	[5, 5, 5, 5, 2, 5, 3, 2, 0, 3, 9, 5, 7, 7, 0, 0, 6, 1, 1, 1, 4, 1, 4, 2, 8, 4, 1, 4, 8, 0, 1, 1, 0]
	DQN	[5, 5, 5, 5, 4, 1, 4, 8, 3, 3, 1, 8, 2, 6, 7, 7, 9, 5, 1, 4, 0, 5, 2, 1, 0, 0, 1, 1, 1, 4, 2, 0, 0]
Task 5	EDD	[1, . . . 3, . . . 5, . . . 8, . . . 9, . . . 7, . . . 2, . . . 0, . . . 4, . . . 6, . . .]
	SPT	[9, . . . 3, . . . 7, . . . 5, . . . 2, . . . 1, . . . 4, . . . 6, . . . 0, . . . 8, . . .]
	GA	[3, 5, 5, 5, 8, 3, 3, 8, 7, 7, 1, 9, 2, 0, 0, 1, 8, 0, 6, 4, 4, 9, 2, 7, 1, 2, 9, 4, 6, 6]
	DQN	[9, 9, 9, 3, 1, 5, 3, 1, 8, 5, 7, 8, 6, 7, 7, 8, 6, 3, 1, 0, 0, 5, 2, 2, 2, 0, 6, 4, 4, 4]

Source(s): Authors' own work

rely on a static sequencing policy. Instead, DQN uses a neural network-based policy, and GA applies iterative chromosome evolution to redefine the manufacturing order. This flexibility enables both DQN and GA to generate more optimal scheduling plans, tailored to the varying conditions of each task.

Compared to EDD and SPT methods, the optimal schedules generated by DQN or GA incur higher earliness penalties but achieve lower total penalties overall. This discrepancy suggests that to fully utilize the crew, completing certain jobs slightly ahead of schedule can reduce the overall total penalty. This conclusion is reflected in the training process of DQN, as shown in Figure 7–9. During the exploration phase of DQN training, in pursuit of higher rewards, the DQN agent continuously adjusts the balance between earliness and tardiness penalties. At the end of the training, the policy output by DQN tends to adopt the early completion of some PC elements. Although this approach may slightly increase the earliness penalty, it meanwhile reduces more tardiness penalty, thereby decreasing the total penalties. An illustrative example is shown in Figure 10. Assume there are 5 crew members, with two types of jobs requiring 2 and 3 crew members respectively, and different due dates. Figure 10a illustrates the job sequence according to EDD; Job 1 is completed before the due date, while Job 2 finishes later than the due date. Additionally, this schedule does not make full use of the crew resources. Figure 10b presents a more optimal schedule, where both Jobs are processed simultaneously, ensuring full crew utilization and completing both jobs before their respective due dates. This illustrative example explains why schedules generated by the DQN are superior to those created by DPs.

6.3 Discussion on the generalization ability

In the design of DRL, generalization ability has consistently been a major challenge. It is difficult to use a single DRL policy to address all production scheduling problems. In the context of precast facilities, however, incoming orders typically consist of a limited and standardized set of PC element types, with standardized dimensions as well. This paper focuses on scheduling and rescheduling precast orders under parallel production lines and limited resources conditions. The generalization ability of the proposed method is demonstrated by the fact that the pre-trained policy can be directly applied to new orders with varying amounts of precast elements and different due dates. However, as the initial conditions change, the pre-trained policy may not generate optimized schedules. To address this, the state design proposed in this paper incorporates completion rates and progress percentages to describe the production status of all precast elements, thereby mitigating the influence of the actual number of precast orders and their due dates in the state representation. This approach enhances the DRL algorithm's ability to maintain high generalization when faced with different orders. The action design in the proposed method directly assigns jobs to production lines, which ties the action space to the number of production lines and the types of PC elements. For precast orders with the same type of precast elements, the action space remains fixed. Therefore, the proposed method can serve as a pre-trained policy that can be directly applied to precast orders that differ from the training tasks.

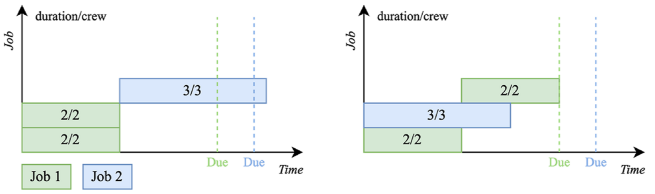


Figure 10. Illustrative example for different job sequences. Source: Authors' own work

6.4 Discussion on the method implementation and social impact

The proposed DRL-based scheduling method provides a practical and adaptable solution that can be directly implemented in precast factories with similar production structures. The implementation process involves two main stages: initial model training and policy deployment. During the training phase, production configurations—such as mold plate availability, labor crew settings, and PC order characteristics—are used to simulate the production environment. Once the model is trained, the pre-trained DQN agent can be deployed to handle ongoing and future orders without retraining. Production managers only need to input the new order data, and the system will automatically generate optimized scheduling plans through inference from the DQN policy.

In terms of system integration, the proposed DRL framework can be incorporated into existing factory management systems or production planning tools. Python-based interfaces and modular environment design allow for straightforward Application Programming Interface integration or standalone scheduling applications. With appropriate customization, factories using different mold configurations or product types can also retrain the model with their specific parameters using the methodology outlined in this study.

The practical implications of the proposed method extend to various construction participants:

(1) Production managers can apply the pre-trained policy to instantly generate feasible and optimized production schedules. This reduces reliance on manual rule-based planning and enhances adaptability in the face of new or changing orders, improving decision-making efficiency in real-time factory operations.

(2) Project managers and contractors benefit from increased reliability in the supply of PCs. The improved schedule consistency enables better coordination between off-site production and on-site assembly, helping to maintain construction timelines and reduce downtime.

(3) Labor planners are supported by more balanced and transparent labor allocation. The model considers crew availability and task duration, which helps to avoid workforce bottlenecks and improve scheduling fairness, ultimately contributing to safer and more sustainable working conditions.

(4) Policy makers and sustainability advocates may find the method useful for promoting efficient resource utilization and environmental responsibility. The reduction in idle time, early production, and unnecessary storage aligns with green building practices and sustainable construction regulations.

From the social and policy impact perspectives, optimizing precast production scheduling presents both opportunities and challenges that extend beyond operational efficiency. Socially, effective scheduling can enhance workforce stability by providing predictable labor demands, thereby improving job security and worker satisfaction. However, excessive optimization may inadvertently intensify workloads or reduce employment opportunities through automation, potentially undermining labor welfare. Furthermore, while streamlined production can minimize prolonged construction-related disruptions (e.g. noise and traffic congestion), compressed schedules risk extending work into antisocial hours, generating community grievances. From a policy standpoint, production scheduling must rigorously comply with labor regulations, occupational safety standards, and environmental directives. Algorithmic scheduling models must therefore embed these legal and ethical constraints to prevent violations. Notably, optimized scheduling aligns with broader sustainability objectives by reducing material waste and energy consumption, potentially qualifying projects for green building certifications or policy incentives.

7. Limitations and future directions

While this study presents a novel and practical DRL-based approach for parallel precast production scheduling, several limitations must be acknowledged. These limitations pertain to four key areas: (1) algorithmic scalability and efficiency, particularly in handling large action spaces and bootstrapping issues inherent in DQN; (2) the narrow focus of optimization objectives, which currently considers only makespan and penalties for earliness and tardiness; (3) the simplified modeling of resource constraints, which excludes potential variability in material availability and process disruptions; and (4) the reliance on a fixed mold production mode, which may not generalize to more automated and efficient flow shop production environments. Recognizing these limitations provides valuable direction for future research aimed at enhancing the robustness, generalizability, and industrial applicability of DRL-based scheduling systems in the precast construction sector.

7.1 Limitation of the algorithm

This study employs a single-agent DQN framework with an action masking mechanism to optimize production scheduling. However, several limitations exist concerning the model's expressiveness and scalability. As a value-based method, DQN must evaluate all feasible actions, which becomes challenging as the scale of production increases. Although the proposed action space design effectively avoids the exponential growth commonly associated with complex combinatorial job assignments, it still maintains a strong dependency on the number of production lines and PC element types. In larger or more complex scenarios, the action space can still become significantly large. Even with action masking to filter invalid options, the agent must compute Q-values for a substantial number of valid actions at each decision step. This leads to longer training times and increases the risk of convergence to local optima. Additionally, traditional DQN suffers from bootstrapping flaws, which can lead to biases in Q-values computation, making the DQN training challenging to converge and decrease the rescheduling performance.

7.2 Limitation of the optimization objectives

Since the limited existing research on DRL applications in parallel precast production scheduling optimization, this study focuses on two primary objectives: minimizing makespan and reducing total penalties associated with earliness and tardiness. Cases collected from an actual factory are utilized for performance validation. While these are critical metrics in practice, real-world applications often involve broader objectives. Factors such as energy consumption, carbon emissions, mold changeover costs, and labor overtime expenses are also essential for sustainable and cost-effective production. Future studies should incorporate multi-objective optimization to better reflect the complexity of real-world decision-making in precast factories.

7.3 Limitation of the constraints

This study models constraints related to the number of available mold plates and specialized labor crews. While these constraints represent key bottlenecks in most PC factories, other resource limitations can arise in practice. Although the surveyed factory possesses an independent concrete center and a reinforcement cage manufacturing center, ensuring resource supply in most situations, materials may still become scarce during peak demand periods. Additionally, the process times used in this study are treated as fixed and deterministic, which does not reflect real-world uncertainties caused by labor fatigue, equipment malfunctions, or supply delays. Consequently, future work should incorporate stochastic process times and additional resource constraints to create more robust and realistic scheduling models.

7.4 Limitation of the production mode

The cases for validation are based on fixed mold production mode, which remains widely used due to its lower site requirements and straightforward implementation. However, this method has the drawback of lower production efficiency, which is constrained by the number of mold plates and skilled crews. Field observations revealed that some factories are transitioning toward mechanized flow shop production, which leverages automation to improve throughput and resource utilization. Therefore, future work should extend the current DRL framework to address the unique scheduling complexities and optimization opportunities in parallel flow shop production environments.

7.5 Future directions

To bridge existing gaps in the field, future research could focus on integrating multi-agent deep reinforcement learning and developing more sophisticated DRL frameworks to enhance the training process and address complex challenges. The application of multi-agent DRL shows promise in reducing the complexity of high-dimensional action spaces, thus simplifying neural network architectures and streamlining the DRL training. Enhanced frameworks, such as Double DQN, Noisy Networks, and Prioritized Experience Replay, could be employed to tackle bootstrapping problems in intricate scenarios. Additionally, refining current algorithms through simulation-based GA could tailor solutions for parallel production tasks, while the advancement of multi-objective optimization algorithms leveraging DRL offers a pathway to addressing more nuanced objectives. There exists a substantial potential to enhance labor efficiency and address uncertainties, thereby rendering optimization strategies increasingly relevant and adaptable within actual production environments. Hence, it is crucial for forthcoming studies to explore not merely parallel fixed mold production but to also broaden their investigation to encompass parallel flow shop production, thereby enriching the research domain with multifaceted optimization methodologies. To address the improvement of pre-trained DQN policies in unforeseen scenarios, several effective strategies could be incorporated, including domain randomization to expose the policy to varied scenarios during training; data augmentation methods to enrich the training dataset with diverse state-action experiences; robust adversarial training to explicitly handle challenging or novel states; regularization techniques (e.g. dropout, entropy regularization) to mitigate overfitting; and meta-learning or transfer learning approaches that enhance policy generalization by training across multiple related environments. Employing these approaches can improve the DQN policy's adaptability and robustness in unforeseen cases.

8. Conclusion

This study proposed an automatic parallel precast production scheduling method using DQN to minimize production makespan and penalties of earliness and tardiness. The proposed DQN framework considered various constraints encountered in real-world factories, including resource constraints, labor constraints, and working hour limitations. A simulation environment was developed to replicate the precast production process under these constraints. Building on this, an improved simulation-based GA was presented for comparison with the DQN method. The proposed DQN method is designed to obtain a pre-trained policy that can directly generate optimal production schedules with minimized earliness and tardiness penalties as well as shortened makespan. The stability and feasibility of the DQN method were validated by empirical experiments conducted in actual factories, including several rescheduling tasks. When compared with GA and traditional DPs like EDD and SPT, the DQN method outperformed in terms of effectiveness and efficiency in generating schedules. The proposed DRL framework also exhibits rescheduling capabilities that can accommodate diverse and dynamic order requirements. Real-world case data collected from a surveyed factory were used to further validate the practical applicability of the proposed approach.

Future research could explore the use of innovative DRL frameworks to improve production scheduling. Research on parallel flow production is another direction that merits attention. As the Internet of Things continues to drive automation in precast production, the application of DRL methods is expected to have a transformative impact on the industry's future development. Additionally future work should consider diversifying scheduling objectives, such as incorporating carbon emissions into the optimization framework. Finally, integrating production scheduling with construction scheduling could lead to the development of a unified scheduling system, supporting optimal decision-making across the entire industrial supply chain.

Acknowledgments

The authors wish to acknowledge the financial support from the Australian Research Council (ARC), Australian Government (Nos: DP200100057, IH200100010; FT220100017; IC240100020; DP250101775).

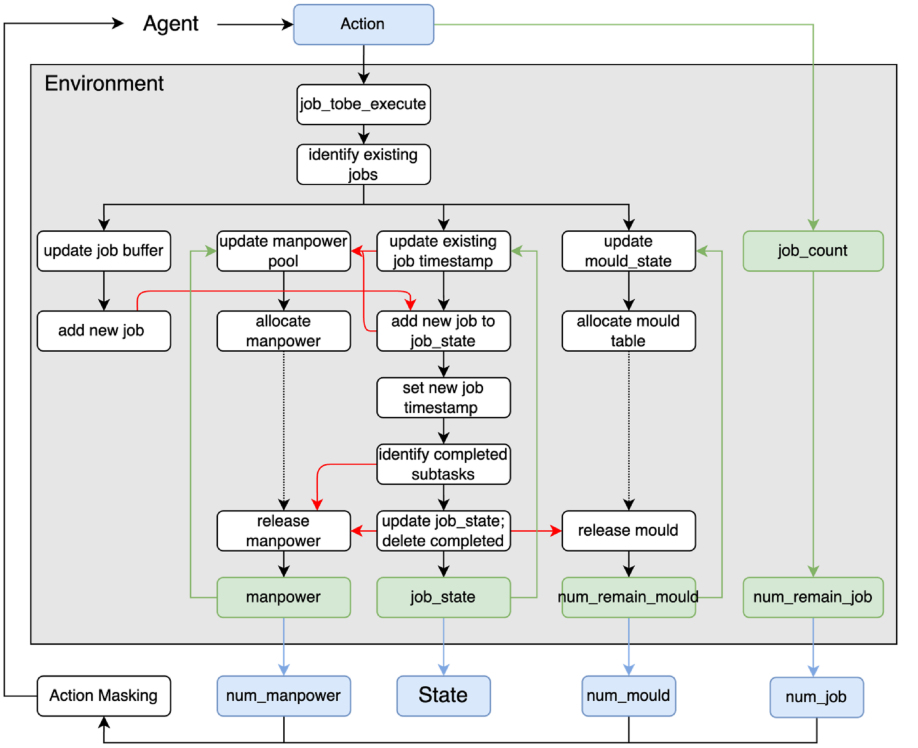
Appendix 1

Table A1. Nomenclature

Abbreviations	Meanings
PC	Precast Component
PCPS	Precast Component Production Scheduling
DP	Dispatching Rule
EDD	Earliest Due Date
SPT	Shortest Process Time
GA	Genetic Algorithm
DRL	Deep Reinforcement Learning
EA	Evolutionary Algorithm
NP-hard problem	As hard as Non-deterministic Polynomial problem
CP	Constrained Programming
DQN	Deep Q-Network
MDP	Markov Decision Process
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
RBRS	Regret-based Biased Random Sampling
SSGS	Serial Schedule Generation Scheme

Source(s): Authors' own work

Appendix 2
Flow chart of agent-environment interaction



Source(s): Authors' own work

Appendix 3
Illustrative example of state transition.

For better illustrate the state transition, here presents an example of how State S is determined in real simulation. Let's assume a precast order with 2 types of elements sent to 3 parallel production lines. The total number of each type is 5 and 10 respectively; and due dates is 72 h and 120 h respectively. At the beginning moment $t = 0$, there are no PC being processed; and thus the S_0 is:

$$S_0 = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

Each roll in S represents a single production line, the -1 values represent that currently no job is assigned to any production line. Then the agent choose action $a_0 = [1, 2]$, which means 1 and 2 PC elements of two types are assigned respectively. Assuming the first process times (mold assembling) of the PCs are 2 and 3 respectively, then after action execution, the S_1 is:

$$S_1 = \begin{bmatrix} 0 & \frac{1}{5} & \frac{1}{72} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 1 & \frac{2}{10} & \frac{1}{120} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 1 & \frac{2}{10} & \frac{1}{120} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The first column of S represents the PC codes. The second column represents the current completion/processing ratio of each type of PC. The third column represents the relative value between the current time and due dates. The last six columns represent the completion rates of each production process.
Source(s): Authors' own work

Appendix 4

Table A2. Algorithm parameter settings

DQN parameters	
Discount factor	0.95
Epsilon_start	0.95
Epsilon_end	0.01
Epsilon_decay	500
Batch size	256
Learning rate	0.0001
Weight decay	0.01

Source(s): Authors' own work

References

Alshboul, O., Al-shboul, K., Shehadeh, A. and Tatari, O. (2025a), "Advancing equipment management for construction: introducing a new model for cost, time and quality optimization", *Construction Innovation-england*, Vol. 5 No. 8.

Alshboul, O., Shehadeh, A. and Almasabha, G. (2025b), "Reliability of information-theoretic displacement detection and risk classification for enhanced slope stability and safety at highway construction sites", *Reliability Engineering and System Safety*, Vol. 256, 110813, doi: 10.1016/j.ress.2025.110813.

Anvari, B., Angeloudis, P. and Ochieng, W.Y. (2016), "A multi-objective Ga-based optimisation for holistic manufacturing, transportation and assembly of precast construction", *Automation in Construction*, Vol. 71, pp. 226-241, doi: 10.1016/j.autcon.2016.08.007.

Benjaoran, V. and Dawood, N. (2006), "Intelligence approach to production planning system for bespoke precast concrete products", *Automation in Construction*, Vol. 15 No. 6, pp. 737-745, doi: 10.1016/j.autcon.2005.09.007.

Booth, S., Knox, W.B., Shah, J., Niekum, S., Stone, P. and Allievi, A. (2023), "The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications", *Proceedings of the AAAI Conference on Artificial Intelligence, Proceedings of the Aaai Conference on Artificial Intelligence*, Vol. 37 No. 5, pp. 5920-5929, doi: 10.1609/aaai.v37i5.25733.

Chan, W.T. and Hu, H. (2001), "An application of genetic algorithms to precast production scheduling", *Computers and Structures*, Vol. 79 No. 17, pp. 1605-1616, doi: 10.1016/s0045-7949(01)00036-0.

Chan, W.T. and Hu, H. (2002a), "Constraint programming approach to precast production scheduling", *Journal of Construction Engineering and Management*, Vol. 128 No. 6, pp. 513-521, doi: 10.1061/(asce)0733-9364(2002)128:6(513).

Chan, W.T. and Hu, H. (2002b), "Production scheduling for precast plants using a flow shop sequencing model", *Journal of Computing in Civil Engineering*, Vol. 16 No. 3, pp. 165-174, doi: 10.1061/(asce)0887-3801(2002)16:3(165).

Chang, C.G. and Han, M.Y. (2021), "Production scheduling optimization of prefabricated building components based on dde algorithm", *Mathematical Problems in Engineering*, Vol. 2021, pp. 1-11, doi: 10.1155/2021/6672753.

Chen, R., Yang, B., Li, S. and Wang, S. (2020), "A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem", *Computers and Industrial Engineering*, Vol. 149, 106778, doi: 10.1016/j.cie.2020.106778.

Dan, Y.R. and Liu, G.W. (2024), "Integrated scheduling optimization of production and transportation for precast component with delivery time window", *Engineering Construction and Architectural Management*, Vol. 31 No. 8, pp. 3335-3355, doi: 10.1108/ecam-09-2022-0871.

Dan, Y.R., Liu, G.W. and Fu, Y. (2021), "Optimized flowshop scheduling for precast production considering process connection and blocking", *Automation in Construction*, Vol. 125, 103575, doi: 10.1016/j.autcon.2021.103575.

Dawood, N.N. (1995), "Scheduling in the precast concrete industry using the simulation modeling approach", *Building and Environment*, Vol. 30 No. 2, pp. 197-207, doi: 10.1016/0360-1323(94)00039-u.

Du, J. and Leung, J. Y.-t. (1990), "Minimizing total tardiness on one machine is Np-hard", *Mathematics of Operations Research*, Vol. 15 No. 3, pp. 483-495, doi: 10.1287/moor.15.3.483.

Du, Y. and Li, J.Q. (2024), "A deep reinforcement learning based algorithm for a distributed precast concrete production scheduling", *International Journal of Production Economics*, Vol. 268, 109102, doi: 10.1016/j.ijpe.2023.109102.

Du, J., Dong, P. and Sugumaran, V. (2020), "Dynamic production scheduling for prefabricated components considering the demand fluctuation", *Intelligent Automation and Soft Computing*, Vol. 26, pp. 715-723, doi: 10.31209/2020.100000167.

Du, J., Dong, P., Sugumaran, V. and Castro-lacouture, D. (2021), "Dynamic decision support framework for production scheduling using a combined genetic algorithm and multiagent model", *Expert Systems*, Vol. 38 No. 1, e12533, doi: 10.1111/exsy.12533.

Du, Y., Li, J.-q., Chen, X.-l., Duan, P.-y. and Pan, Q.-k. (2022a), "Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem", *Ieee Transactions on Emerging Topics in Computational Intelligence*, Vol. 7 No. 4, pp. 1036-1050, doi: 10.1109/tetci.2022.3145706.

Du, Y., Li, J., Li, C. and Duan, P. (2022b), "A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 35 No. 4, pp. 5695-5709, doi: 10.1109/tnnls.2022.3208942.

Du, J., Xue, Y., Sugumaran, V., Hu, M. and Dong, P. (2023), "Improved biogeography-based optimization algorithm for lean production scheduling of prefabricated components", *Engineering Construction and Architectural Management*, Vol. 30 No. 4, pp. 1601-1635, doi: 10.1108/ecam-04-2021-0311.

Hu, Y., Wang, W., Jia, H., Wang, Y., Chen, Y., Hao, J., Wu, F. and Fan, C. (2020), "Learning to utilize shaping rewards: a new approach of reward shaping", *Advances in Neural Information Processing Systems*, Vol. 33, pp. 15931-15941.

Hua, Z., Liu, Z., Yang, L. and Yang, L. (2022), "Improved genetic algorithm based on time windows decomposition for solving resource-constrained project scheduling problem", *Automation in Construction*, Vol. 142, pp. 1-14, doi: 10.1016/j.autcon.2022.104503.

Huang, S. and Ontañón, S. (2020), "A closer look at invalid action masking in policy gradient algorithms", *Arxiv Preprint Arxiv:2006.14171*, Vol. 8 No. 3.

Jang, S. and Lee, G. (2018), "Process, productivity, and economic analyses of bim–based multi-trade Prefabrication — a case study", *Automation in Construction*, Vol. 89, pp. 86-98, doi: 10.1016/j.autcon.2017.12.035.

Kaelbling, L.P., Littman, M.L. and Moore, A. W. (1996), "Reinforcement learning: a survey", *Journal of Artificial Intelligence Research*, Vol. 4, pp. 237-285, doi: 10.1613/jair.301.

Khalili, A. and Chua, D.K. (2014), "Integrated prefabrication configuration and component grouping for resource optimization of precast production", *Journal of Construction Engineering and Management*, Vol. 140 No. 2, 04013052, doi: 10.1061/(asce)co.1943-7862.0000798.

Kim, J.-m., Zhou, Y.-d. and Lee, D.-h. (2017), "Priority scheduling to minimize the total tardiness for remanufacturing systems with flow-shop-type reprocessing lines", *The International Journal of Advanced Manufacturing Technology*, Vol. 91 Nos 9-12, pp. 3697-3708, doi: 10.1007/s00170-017-0057-z.

Kim, T., Kim, Y.W. and Cho, H. (2020), "Dynamic production scheduling model under due date uncertainty in precast concrete construction", *Journal of Cleaner Production*, Vol. 257, 120527, doi: 10.1016/j.jclepro.2020.120527.

Kim, T., Kim, Y.W., Lee, D.M. and Kim, M. (2022), "Reinforcement learning approach to scheduling of precast concrete production", *Journal of Cleaner Production*, Vol. 336, 130419, doi: 10.1016/j.jclepro.2022.130419.

Ko, C.-h. (2010), "Production control in precast fabrication: considering demand variability in production schedules", *Canadian Journal of Civil Engineering*, Vol. 38 No. 2, pp. 191-199, doi: 10.1139/l10-123.

Ko, C.H. and Wang, S.F. (2010), "Ga-based decision support systems for precast production planning", *Automation in Construction*, Vol. 19 No. 7, pp. 907-916, doi: 10.1016/j.autcon.2010.06.004.

Ko, C.H. and Wang, S.F. (2011), "Precast production scheduling using multi-objective genetic algorithms", *Expert Systems with Applications*, Vol. 38 No. 7, pp. 8293-8302, doi: 10.1016/j.eswa.2011.01.013.

Kolisch, R. and Sprecher, A. (1997), "Psplib - a project scheduling problem library: or software - Orsep operations research software exchange program", *European Journal of Operational Research*, Vol. 96, pp. 205-216.

Kurinov, I., Orzechowski, G., Hamalainen, P. and Mikkola, A. (2020), "Automated excavator based on reinforcement learning and multibody system dynamics", *IEEE Access*, Vol. 8, pp. 213998-214006, doi: 10.1109/access.2020.3040246.

Leu, S.S. and Hwang, S.T. (2001), "A Ga-based model for maximizing precast plant production under resource constraints", *Engineering Optimization*, Vol. 33 No. 5, pp. 619-642, doi: 10.1080/03052150108940936.

Leu, S.S. and Hwang, S.T. (2002), "Ga-based resource-constrained flow-shop scheduling model for mixed precast production", *Automation in Construction*, Vol. 11 No. 4, pp. 439-452, doi: 10.1016/s0926-5805(01)00083-8.

Li, H. and Love, P.E.D. (1998), "Site-level facilities layout using genetic algorithms", *Journal of Computing in Civil Engineering*, Vol. 12 No. 4, pp. 227-231, doi: 10.1061/(asce)0887-3801(1998)12:4(227).

Li, Y., Gu, W., Yuan, M. and Tang, Y. (2022), "Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network", *Robotics and Computer-Integrated Manufacturing*, Vol. 74, 102283, doi: 10.1016/j.rcim.2021.102283.

Lin, R. and Liao, C.-j. (2013), "Batch scheduling problem for a machinery factory with fixed-position layout", *International Journal of Production Research*, Vol. 51 No. 3, pp. 910-926, doi: 10.1080/00207543.2012.693216.

Lin, C.C., Deng, D.J., Chih, Y.L. and Chiu, H.T. (2019), "Smart manufacturing scheduling with edge computing using multiclass deep Q network", *IEEE Transactions on Industrial Informatics*, Vol. 15 No. 7, pp. 4276-4284, doi: 10.1109/tii.2019.2908210.

Lin, J., Li, Y.-y. and Song, H.-b. (2022), "Semiconductor final testing scheduling using Q-learning based hyper-heuristic", *Expert Systems with Applications*, Vol. 187, 115978, doi: 10.1016/j.eswa.2021.115978.

Liu, J., Liu, Y., Shi, Y. and Li, J. (2020), "Solving resource-constrained project scheduling problem via genetic algorithm", *Journal of Computing in Civil Engineering*, Vol. 34 No. 2, 04019055, doi: 10.1061/(asce)cp.1943-5487.0000874.

Liu, Z.S., Liu, Z.S., Liu, M. and Wang, J.J. (2021), "Optimization of flow shop scheduling in precast concrete component production via mixed-integer linear programming", *Advances in Civil Engineering*, Vol. 2021 No. 1, 6637248, doi: 10.1155/2021/6637248.

Liu, W., Tao, X., Mao, C. and He, W. (2023), "Scheduling optimization for production of prefabricated components with parallel work of serial machines", *Automation in Construction*, Vol. 148, 104770, doi: 10.1016/j.autcon.2023.104770.

Luo, S., Zhang, L. and Fan, Y. (2021), "Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning", *Computers and Industrial Engineering*, Vol. 159, 107489, doi: 10.1016/j.cie.2021.107489.

Ma, Z.L., Li, S.Y., Wang, Y. and Yang, Z.Q. (2021), "Component-level construction schedule optimization for hybrid concrete structures", *Automation in Construction*, Vol. 125, 103607, doi: 10.1016/j.autcon.2021.103607.

Mao, C., Shen, Q., Shen, L. and Tang, L. (2013), "Comparative study of greenhouse gas emissions between off-site prefabrication and conventional construction methods: two case studies of residential projects", *Energy and Buildings*, Vol. 66, pp. 165-176, doi: 10.1016/j.enbuild.2013.07.033.

Mao, W.Q., Ran, K.X., Wang, T.K., Yu, A. Y., Lv, H.Y. and Chen, J.H. (2024), "An optimization model for just-in-time (jit) delivery of precast components considering 3d loading constraints, real-time road conditions and assembly time", *Engineering Construction and Architectural Management*, Vol. 32 No. 5, pp. 3259-3284, doi: 10.1108/ecam-04-2023-0372.

Martin, X.A., Herrero, R., Juan, A. A. and Panadero, J. (2024), "An agile adaptive biased-randomized discrete-event heuristic for the resource-constrained project scheduling problem", *Mathematics*, Vol. 12, p. 1873, doi: 10.3390/math12121873.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D. (2015), "Human-level control through deep reinforcement learning", *Nature*, Vol. 518 No. 7540, pp. 529-533, doi: 10.1038/nature14236.

Moerland, T.M., Broekens, J. and Jonker, C.M. (2020), "A framework for reinforcement learning and planning", Arxiv Preprint Arxiv:2006.15009, 127.

Pan, Z., Wang, L., Wang, J. and Lu, J. (2021), "Deep reinforcement learning based optimization algorithm for permutation flow-shop scheduling", *Ieee Transactions on Emerging Topics in Computational Intelligence*, Vol. 7 No. 4, pp. 983-994, doi: 10.1109/tetci.2021.3098354.

Peiris, A., Hui, F.K.P., Duffield, C. and Ngo, T. (2023), "Production scheduling in modular construction: Metaheuristics and future directions", *Automation in Construction*, Vol. 150, 104851, doi: 10.1016/j.autcon.2023.104851.

Peng, F. and Zheng, L. (2023), "Integrating real-time manufacturing data into a novel serial two-stage adaptive alternate genetic fireworks algorithm for solving stochastic Type-ii simple assembly line balancing problem", *Complex and Intelligent Systems*, Vol. 9 No. 6, pp. 7075-7102, doi: 10.1007/s40747-023-01091-7.

Podolski, M. (2022), "Effective allocation of manpower in the production of precast concrete elements with the use of metaheuristics", *Journal of Civil Engineering and Management*, Vol. 28 No. 4, pp. 247-260, doi: 10.3846/jcem.2022.16383.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M. and Dormann, N. (2021), "Stable-baselines 3: reliable reinforcement learning implementations", *Journal of Machine Learning Research*, Vol. 22, pp. 1-8.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T. and Silver, D. (2020), "Mastering atari, Go, chess and shogi by planning with a learned model", *Nature*, Vol. 588 No. 7839, pp. 604-609, doi: 10.1038/s41586-020-03051-4.

Silver, D., Singh, S., Precup, D. and Sutton, R.S. (2021), "Reward is enough", *Artificial Intelligence*, Vol. 299, 103535, doi: 10.1016/j.artint.2021.103535.

Sivamayil, K., Rajasekar, E., Aljafari, B., Nikolovski, S., Vairavasundaram, S. and Vairavasundaram, I. (2023), "A systematic study on reinforcement learning based applications", *Energies*, Vol. 16 No. 3, p. 1512, doi: 10.3390/en16031512.

Sivamayilvelan, K., Rajasekar, E., Vairavasundaram, S., Balachandran, S. and Suresh, V. (2024), "Flexible recommendation for optimizing the debt collection process based on customer risk using deep reinforcement learning", *Expert Systems with Applications*, Vol. 256, 124951, doi: 10.1016/j.eswa.2024.124951.

Sutton, R.S. and Barto, A.G. (2018), *Reinforcement Learning: an Introduction*, Mit Press.

Tam, V.W., Fung, I.W., Sing, M.C. and Ogunlana, S.O. (2015), "Best practice of prefabrication implementation in the Hong Kong public and private sectors", *Journal of Cleaner Production*, Vol. 109, pp. 216-231, doi: 10.1016/j.jclepro.2014.09.045.

Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J.P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T.L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., Mckinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C. and Silver, D. (2019), "Grandmaster level in starcraft Ii using multi-agent reinforcement learning", *Nature*, Vol. 575 No. 7782, pp. 350-354, doi: 10.1038/s41586-019-1724-z.

Wang, Z. and Hu, H. (2018), "Dynamic response to demand variability for precast production rescheduling with multiple lines", *International Journal of Production Research*, Vol. 56 No. 16, pp. 5386-5401, doi: 10.1080/00207543.2017.1414970.

Wang, Hu, H. and Gong, J. (2018), "Framework for modeling operational uncertainty to optimize offsite production scheduling of precast components", *Automation in Construction*, Vol. 86, pp. 69-80, doi: 10.1016/j.autcon.2017.10.026.

Wang, Liu, Y.S., Hu, H. and Dai, L. (2021), "Hybrid rescheduling optimization model under disruptions in precast production considering real-world environment", *Journal of Construction Engineering and Management*, Vol. 147 No. 4, 04021012, doi: 10.1061/(asce)co.1943-7862.0001976.

Wang, L., Zhao, Y. and Yin, X. (2023), "Precast production scheduling in off-site construction: mainstream contents and optimization perspective", *Journal of Cleaner Production*, Vol. 405, 137054, doi: 10.1016/j.jclepro.2023.137054.

Xie, L.L., Chen, Y.J., Wu, S.S., Chang, R.D. and Han, Y.L. (2024), "Knowledge extraction for solving resource-constrained project scheduling problem through decision tree", *Engineering Construction and Architectural Management*, Vol. 31 No. 7, pp. 2852-2877, doi: 10.1108/ecam-04-2022-0345.

Xie, L.-l., Li, D., Wu, S. and Chang, R.-d. (2025), "Solving multi-mode resource-constrained scheduling problem of prefabricated construction using genetic algorithm", *Engineering Construction and Architectural Management*, Vol. 8 No. 2, doi: 10.1108/ecam-03-2023-0232.

Xiong, F.L., Chu, M.L., Li, Z., Du, Y. and Wang, L.T. (2021), "Just-in-time scheduling for a distributed concrete precast flow shop system", *Computers and Operations Research*, Vol. 129, 105204, doi: 10.1016/j.cor.2020.105204.

Yang, Z. and Lu, W. (2023), "Facility layout design for modular construction manufacturing: a comparison based on simulation and optimization", *Automation in Construction*, Vol. 147, 104713, doi: 10.1016/j.autcon.2022.104713.

Yang, Z.T., Ma, Z.L. and Wu, S. (2016), "Optimized flowshop scheduling of multiple production lines for precast production", *Automation in Construction*, Vol. 72, pp. 321-329, doi: 10.1016/j.autcon.2016.08.021.

Yao, Y., Tam, V.W.Y., Wang, J., Le, K.N. and Butera, A. (2024), "Automated construction scheduling using deep reinforcement learning with valid action sampling", *Automation in Construction*, Vol. 166, 105622, doi: 10.1016/j.autcon.2024.105622.

Zhou, J. and Ren, D. (2020), "A hybrid model of external environmental benefits compensation to practitioners for the application of prefabricated construction", *Environmental Impact Assessment Review*, Vol. 81, 106358, doi: 10.1016/j.eiar.2019.106358.

Further reading

Park, I.B., Huh, J., Kim, J. and Park, J. (2020), "A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities", *IEEE Transactions on Automation Science and Engineering*, Vol. 17, pp. 1420-1431, doi: 10.1109/tase.2019.2956762.

Corresponding author
Vivian WY Tam can be contacted at: vivianwytam@gmail.com