Automated assembly sequence planning and scheduling in precast building projects using reinforcement learning

Ajay Kumar Agrawal[1], Yang Zou[1], Hongyu Jin[1], Mohammed Abdelmegid[2], Vicente Gonzalez[3]

[1]Department of Civil and Environmental Engineering, University of Auckland, New Zealand
[2]School of Civil Engineering, University of Leeds, UK
[3]Department of Civil and Environmental Engineering, University of Alberta, Canada

aagr657@aucklanduni.ac.nz, yang.zou@auckland.ac.nz, hongyu.jin@auckland.ac.nz, M.Abdelmegid@leeds.ac.uk, vagonzal@ualberta.ca

Abstract

Efficient construction of precast concrete buildings (PCB) with high quality and safety requires adequate planning and scheduling of the assembly of precast components (PC). Traditional manual assembly sequence planning and scheduling (ASPS) methods often result in sub-optimal and error-prone schedules, especially in complex projects. Existing metaheuristic-based ASPS methods lack generalizability and capability to handle dynamic conditions in construction projects. Reinforcement Learning (RL) offers a promising solution to these limitations. However, existing RL-based construction scheduling studies overlook project uncertainties and rely on value-based methods, which are computationally inefficient for large-scale problems. This study proposes a novel method for optimal ASPS of PCB projects using RL integrated with Monte-Carlo Sampling (MCS). The method uses a Temporal Graph Network (TGN) and Multi-Layer Perceptron (MLP) to create embeddings of the relevant input information, while the Proximal Policy Optimization (PPO) RL algorithm is employed for generating the assembly schedules using the embeddings. Reward shaping is utilized to generate schedules that simultaneously minimize time, cost, and constraint violations. Preliminary results from a case study on a hypothetical PC project with 34 precast concrete walls demonstrate the proposed method to provide stable learning and generate optimum assembly schedules. The proposed method outperforms existing value-based RL approaches for construction scheduling, demonstrating superior training stability, solution quality, and execution efficiency.

Keywords –

Assembly sequence planning and scheduling; Precast building construction; Reinforcement learning; Proximal policy optimization

1 Introduction

The growing housing demand has created the need to build 96000 homes every day worldwide [1]. Prefabricated Housing Construction (PHC), an innovative construction method, provides a high-speed and sustainable solution to this challenge [2]. It involves offsite manufacturing of components like columns and walls, which are transported and assembled onsite. Among prefabrication methods, precast concrete buildings (PCB) are widely adopted, using precast concrete elements [2]. Efficient assembly sequencing planning and scheduling (ASPS), considering constraints like resource and material availability, is vital for PCB project success [3]. ASPS refers to the process of defining the sequence and timeline of assembly of precast components (PC). Conventionally, ASPS is done manually by practitioners based on their experience, leading to suboptimal and error-prone results, especially in large and complex projects [4]. Thus, there is a growing focus on automating ASPS in PCB projects.

Existing studies in this direction have primarily utilized metaheuristic methods such as Genetic Algorithm (GA) and Simulated Annealing (SA) [5]. However, such methods lack generalizability to different types of projects [6], and cannot capture the uncertainties and dynamic situations in the construction projects [7].

To this end, RL has been used recently to address construction planning problems [7]. RL is a machine learning technique in which an agent learns to make the best decisions by mimicking humans' trial-and-error learning process. Existing RL-based construction planning techniques have certain limitations, hindering their applicability in ASPS problems. First, they are primarily designed for conventional on-site construction projects. Second, ASPS involves sequencing numerous PCs, creating a large activity set that existing methods struggle to handle due to dimensionality challenges.

Finally, the construction processes' uncertainties were often overlooked. PCB processes can face uncertainties for various reasons, such as component characteristics and weather conditions [8], which need to be taken into account in ASPS.

To address these limitations, this study proposes a novel RL-based method to automatically generate the optimum assembly schedules for PCB, considering installation constraints and uncertainties in the activity durations and weather conditions.

Key innovations of this study are as follows:

a) A novel architecture is proposed using the Proximal Policy Optimization (PPO) algorithm [9], an actor-critic RL algorithm that can effectively handle large state and action spaces compared to the algorithms used in existing construction planning studies.
b) State, action space, and reward function, which play a crucial role in RL, are uniquely defined considering optimization objectives as minimizing time, cost constraint violations [3].
c) An approach to handle uncertainties through Monto Carlo Sampling (MCS) is incorporated with RL.

2 Literature review

2.1 Assembly Sequence Planning and Scheduling in PCB Projects

Initial studies in this direction utilized structural support and topological relationships between the building components to generate assembly sequences [10]. However, schedule optimization was not done. Later, researchers utilized Building Information Models and optimization algorithms to develop optimum assembly sequences. Optimization of ASPS has been done from various perspectives, such as maximization of structural stability [11], prioritization of heavier and larger components, minimization of installation time [3]–[5], [12], and reduction in interference between components, among others [12], [13]. Most of the studies utilized GA for optimization [3], [4], [11], [13]. To avoid the risk of slow convergence in large-scale problems, Liu et al. [5] integrated GA with SA. Other researchers used modified versions of GA to ensure population diversity [4]. However, existing studies failed to simultaneously consider assembly time, cost, and constraint violation as schedule optimization objectives. They lack the ability to swiftly adapt to dynamic conditions and uncertainties on the construction projects, as the solutions developed by GA only apply to the initial conditions of the optimization [14].

2.2 RL for construction planning

In RL, an agent learns to find the policy to make the best decisions by iteratively interacting with an environment, as shown in Figure 1. At each time step, the agent selects an action from the set of possible actions (known as action space), and the environment rewards the agent based on the selected action. Over the iterations, the agent improves the decision-making by maximizing the reward. In contrast to algorithms such as GA, RL algorithms can handle dynamic situations more efficiently as they automatically learn the best optimization strategy based on the interaction with the environment [6].

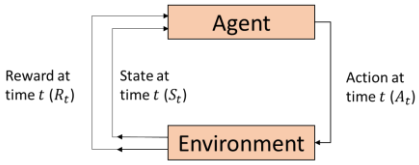

Figure 1. RL Model

Soman et al. [15] used Q-learning (a class of RL algorithms that maximizes the Q-function to find the best policy by estimating cumulative rewards for each state-action pair) to generate construction lookahead schedules automatically. Kedir et al. [7] utilized deep learning to estimate Q-function. Yao et al. [6] utilized Deep Q-learning (DQN) and sampling constraint-free actions to achieve faster convergence. However, they addressed precedence and resource constraints only. ASPS in PCB requires consideration of several constraints, such as the size and weight of the PC and interference between PCs. Moreover, the uncertainties of the construction environment were not considered, and only time-based schedule optimization was performed.

The Q-learning algorithm becomes computationally expensive for large-scale problems, for example, resource-constrained construction planning and scheduling [16]. The complexity further escalates in uncertain environments, where state spaces become continuous due to variables like activity durations and weather conditions. DQN algorithm is faster than Q-learning, as the Q-function estimation is done using deep learning. However, it is prone to suffer from instability in such cases of continuous state spaces [17]. Actor-critic algorithms, a class of RL algorithms, combine the strength of value and policy-based approaches. These algorithms use an actor to update the policy directly and a critic to evaluate the value function. Proximal Policy Optimization (PPO), a state-of-the-art actor-critic algorithm, balances exploration and stability [9], making it effective for handling complex, dynamic, and large-scale problems like ASPS. Despite its potential, PPO has not been extensively applied in this domain.

2.3 Problem statement

Based on the identified research gaps, this study proposes a method for ASPS in PCB projects using the PPO algorithm to generate assembly schedules that:

a) minimize time, cost, and constraint violations associated with the assembly. These objectives are selected based on their frequent use in the existing literature [3], [12], [13].
b) incorporate the uncertainty in the PCB projects. Two primary sources of uncertainties are considered: inherent variations in the activity durations and external uncertainty due to weather, both of which are prevalent in PCB projects [8].

3 Methodology

3.1 Problem formulation and mathematical modeling

Consider that the PCB consists of n elements to be scheduled. Each of these elements belongs to the component type $CT = [1,2,3, \dots i]$. The component types correspond to different categories in which the components can be divided, e.g., beams, columns, walls, etc. Each element e_k requires renewable resources $r_k = [r_{ka}, r_{kb}, r_{kc,\dots,} r_{km}]$, where r_{ka} denotes the requirement of resources of the category a. $R = [R_a, R_b, R_c, \dots, R_m]$ denotes the maximum availability of renewable resources in categories $[a, b, c, \dots, m]$. Each element e_k requires time t_k for assembly. The time requirement t_k for assembly of each element is assumed to follow a triangular distribution based on the findings of previous studies [18]. Weather conditions, including rain, wind, and temperature, are considered to impact the assembly durations [8]. The distribution of the values of these three weather conditions can be taken from a weather forecast [19]. The impact of weather conditions on the assembly durations can be identified using general codes of practice and relevant laws and regulations [8].

3.1.1 Objectives

$$minimize\ D_{total}, C_{total}\ and\ CV_{total} \quad (1)$$

Where D_{total} is the total scheduled duration of the project, C_{total} is the total cost (including direct cost and indirect cost) of the project and CV_{total} denotes the total number of constraint violations in the generated schedule.

3.1.2 Constraints

The key constraints for ASPS were identified based on the literature. The constraints are divided into two categories: Hard constraints and soft constraints.

(a) Hard constraints: The constraints that must not be violated as they ensure the structural integrity of the building and adherence to scheduling logic.
  (i) Structural support constraints: The PC should be assembled according to structural support requirements [11].
  (ii) Resource availability constraint: The allocated resources must be within the limits of the R [8].
  (iii) Component availability: A PC can be assembled only after its delivery on-site [8]. A PC delivery schedule is used to consider component availability.
(b) Soft constraints: The constraints whose violation should be minimized as much as possible.
  (i) Weight constraint: The heavier the PC, the earlier it should be assembled [3].
  (ii) Size constraint: The larger the space occupied by the PC, the earlier it should be assembled [3], [12].
  (iii) Interference constraint: Interference between the new PC and the pre-installed PC should be minimized [3].

3.2 RL model for ASPS using PPO

3.2.1 Network architecture

The network architecture of the proposed model for automated ASPS using PPO is depicted in **Error! Reference source not found.**.

3.2.1.1 State:

The state represents the combination of information required by the agent for decision-making. In this study, the state information at a timestep includes the following information:

(a) The construction schedule generated till the timestep is represented in the form of a graph. Each graph node represents the elements and their properties, and the edges represent their relationships. The properties of the nodes are represented in **Error! Reference source not found.**. The first property represents the status (not started, in progress, completed) of the element. The second property corresponds to CT. The third and fourth properties represent the resources needed. They can be customized based on the number of categories of resources. The next two properties focus on the completed and remaining duration of elements till the current timestep. The last two properties represent the weight and the area of the elements, respectively.
(b) The availability of PC on-site, represented using a binary vector of length equal to n.

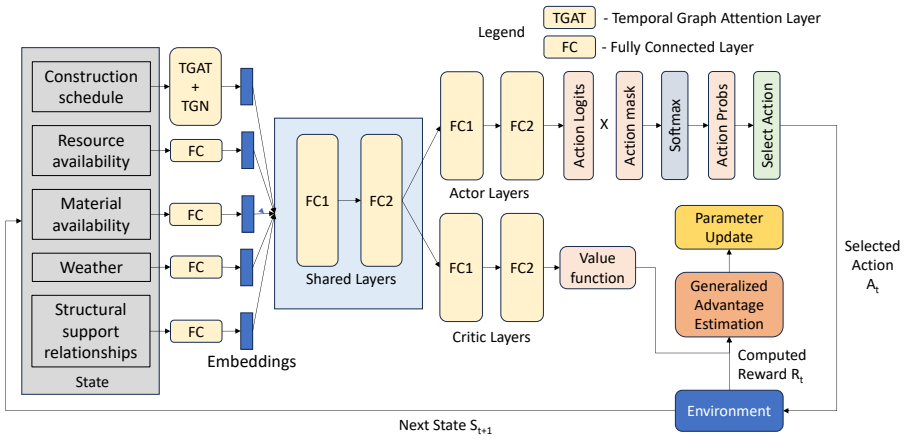

Figure 2 . Model architecture

Element status	Element type	Crane needed	Labor needed	Elapsed duration	Remaining duration	Element weight	Element area

Figure 3 . Node features

(c) The availability of resources of different categories, represented using a vector of length equal to R.

(d) Structural support relationships in the form of binary vector S of shape $[n, n]$. A value of $S_{ij} = 1$ represents that the element i is supported by the element j.

(e) Weather is represented in the form of a vector of dimension $[3, T]$, where T denotes the length of the forecast period for which the weather data is considered. Each column represents a particular day, and three rows represent the sampled values of rain, temperature, and wind.

3.2.1.2 Action:

Considering the n elements to be scheduled, the action space is a matrix A of size 2^n, where each row represents a binary vector of length n, indicating which elements should be started (1 for start, 0 otherwise). In PCB, the number of elements installed simultaneously is limited by the project's available cranes, so actions exceeding this limit are removed.

3.2.1.3 Reward:

As this study intends to address three optimization objectives, i.e., time, cost, and constraint violations simultaneously, the reward structure used in this study consists of the following components addressing these objectives:

(a) Time:

$$R_{t1} = r_{w1} * N_s + r_{w2} * N_f - r_{w3} * t$$

$$R_{t2} = r_{w4}$$

Where R_{t1} is the reward given to each timestep. It includes a positive reward for the number of elements started (N_s), the number of elements finished (N_f), and negative reward for each timestep (t). A large positive reward of R_{t2} is added to finish all the elements. r_{w1}, r_{w2}, r_{w3}, and r_{w4} are scalar reward coefficients.

(b) Cost:

$$R_c = -r_{w5} * [\Delta t * (R_{ct} + I_{ct})]$$

Where R_c is the cost reward per time step, Δt is the time difference between the current and previous timestep, R_{ct} and I_{ct} are resource and indirect costs per unit of time, and r_{w5} is the reward coefficient.

(c) Constraint violation:

$$R_{con} = -r_{w6} * N_{sc}$$

Where R_{con} is the reward for constraint violations, N_{sc} is the number of soft constraint violations in that timestep, and r_{w6} is the reward coefficient. The violations of weight and area constraints are computed based on the weight and area values in the node features. For interference constraint violations, an adjacency matrix and an interference matrix are used [4]. The adjacency matrix represents the adjacent elements. The interference matrix is a $[n, n]$ matrix, where each element is a 1D vector IC_{ij} of length 6, which is as follows:

$$IC_{ij} = [d_{x+}, d_{x-}, d_{y+}, d_{y-}, d_{z+}, d_{z-}]$$

Where:

IC_{ij} = a binary vector of interferences if the element j is installed before the element i,

$$d_* = \begin{cases} 1, if\ \ i\ can\ move\ in\ that\ direction \\ 0, otherwise \end{cases}$$

For example:

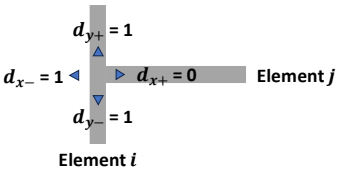

Figure 4. Plan of connection between two elements

3.2.1.4 Environment:

(a) Initialization:
The environment is initialized with constraints such as resource availability, structural support constraints, element weights, etc. The MCS is used to handle uncertainties in activity durations and weather conditions. MCS uses repeated random sampling of a stochastic variable from its distribution to approximate the desired quantity [20]. It finds suitability in solving stochastic optimization problems due to its simplicity and ability to approximate well with a small number of samples [20]. We sampled installation durations and weather conditions from their distributions at each episode's start. The episode is then run using these values. This way, the RL agent learns to perform well over the distribution of installation durations and weather conditions.

(b) Scheduling Logic:
At each timestep, the agent selects an action, and the environment updates the schedule, calculates rewards, and returns the updated state and reward. The elements in the selected action are started first, and resources are assigned to them. Then, the earliest finishing element is finished, and its resources are released. The rewards are calculated, and the next state is sent to the agent. The episode ends when all elements are scheduled.

3.2.1.5 Network architecture

The state information, including the resource availability vector, material availability vector, weather vector, and structural support relationship vectors, are embedded using Fully Connected (FC) layers. The construction schedule, which is represented in the form of a graph, is embedded using a combination of Temporal Graph Attention Network (TGAT) [21] and Temporal Graph Network (TGN) [22]. At each timestep, the graph evolves as the node features and their relationships change. The conventional GNNs used in existing RL-based construction scheduling studies [6] fail to capture such temporally evolving dynamic graphs [22]. TGNs provide an efficient way to deal with such dynamic and time-dependent networks [22]. TGN, when used with TGAT, encodes time features in addition to the node features and uses a self-attention mechanism [23] to identify important parts of the graph to generate the embedding. TGN also incorporates a memory module using Gated Recurrent Units to retain information from previous timesteps, enabling the network to understand how the schedule evolves over time. The actor-critic network includes two shared FC layers and two separate FC layers.

3.2.1.6 Action masking

An action mask is created for actions that violate hard constraints. It filters out such invalid actions and improves training efficiency by reducing the number of possible actions.

4 Case study

To validate the proposed method, a hypothetical project consisting of ASPS of 34 precast walls is used based on existing studies [4], [8]. The project consists of walls of varying thicknesses. Figure 5 shows the walls in the project along with their ID.

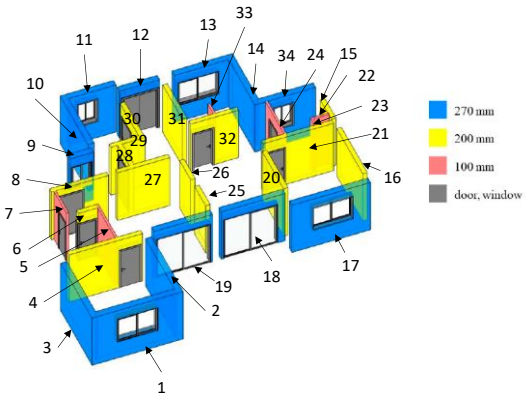

Figure 5. Case study project (Picture credit [4])

As the weights and areas of the walls were not available in Yang et al. [4], they were assumed based on the relative sizes of walls in Figure 5. Similarly, the distribution of installation duration of the walls was not available. Therefore, all the walls were assumed to have the same triangular distribution of durations, derived from Yuan et al. [18]. The assumed installation duration distribution for each wall is (0.47, 0.625, 0.78) hrs. The details of the walls are shown in Figure 5, Table 1, and Table 2. The project is assumed to be constructed in Auckland, New Zealand (NZ). The weather forecast of Auckland is taken from online sources [24], [25]. The impact of weather conditions on construction activities is considered based on local industry standards. Based on the weather forecast, the temperature and rain values are

found to have no impact, and the impact of wind speed is considered based on recommendations from the Crane Association of NZ [26]. Lifting is assumed to involve two mobile cranes, with worker requirements based on Yuan et al. [8]. The larger walls (Area >= 2.5m^2) are assumed to require 6 workers, and others require 4 workers. Costs for the crane and workers are derived from Rawlinsons New Zealand Construction Handbook, 2014 [27]. All the elements were assumed to be already available.

Table 1 Wall weights

Wall ID	Weight (kg)	Wall ID	Weight (kg)
6, 7, 22, 23, 33	50	13	225
5, 9, 15	75	1, 17	250
24, 25, 26, 28	100	11, 34	275
8, 12, 18, 19, 29, 30, 32	150	2, 3, 10, 14	300
4, 16, 20, 21, 27, 31	175		

Table 2: Wall areas

Wall ID	Area (m^2)
22, 23, 25, 26, 33	0.5
6, 7, 9, 15, 28	0.75
5, 24, 30, 32, 29, 31	1.25
8, 12, 27	2
2, 3, 10, 13, 14, 16, 20	2.5
11, 18, 19, 34	2.75
1, 4, 17, 21	3

The validation experiment included two facets: (1) a comparison between the performance of the developed approach with existing DQN-based construction scheduling methods and (2) a sensitivity analysis to understand the impact of scales of reward coefficients. For the first aspect, the comparison was done between the results obtained using DQN+GNN (DQN RL algorithm and GNN for schedule embedding [6]), and PPO+TGN+TGAT (proposed in this study). The key hyperparameters for the three settings are described in Table 3. For the second facet, 7 scenarios of reward coefficient were created, as described in Table 4.

Table 3: Key Hyperparameters for training

Hyperparameter	DQN+ GNN	PPO+ TGN+ GAT
PPO clip epsilon	NA	0.25
PPO entropy coeff.	NA	0.08
Learning rate	0.0003	0.0003
Gamma	0.99	0.99
GAE lambda	NA	0.95
Graph network layers	2	2
Epsilon decay	1000	NA
Epsilon start, end	1, 0.01	NA

Table 4: Sensitivity analysis scenarios

Scenario	r_{w1}	r_{w2}	r_{w3}	r_{w4}	r_{w5}	r_{w6}
S1	1000	1000	2000	20000	5	1000
S2	1000	1000	2000	20000	5	10
S3	1000	1000	2000	20000	0.05	1000
S4	1000	1000	2000	20000	0.05	10
S5	10	10	20	200	5	1000
S6	10	10	20	200	0.05	1000
S7	10	10	20	200	5	10

The algorithm was run on a Windows 11 computer, with Intel® Xeon® Gold 6242R CPU @ 3.10GHz processor and NVIDIA Quadro RTX 5000 16.0 GB GPU for 5000 episodes. The high number of episodes was to consider the uncertainties in the problem, such that the agent can learn over the distribution of duration and weather conditions based on MCS. The results of training on Scenario S1 are demonstrated in Figure 6**Error! Reference source not found.Error! Reference source not found.**. The graphs show the 100-episode average of computed rewards, costs, schedule durations, and constraint violations. Since each episode is trained on different initial conditions, averaging over 100 episodes provides a clearer trend. While existing studies developed assembly sequences focusing on single objectives, our method effectively generated schedules minimizing cost, time, and constraint violations simultaneously. For PPO+TGN+TGAT, the agent successfully discovered a policy that maximizes the reward, achieving greater consistency after approximately 2000–3000 episodes. Similarly, the number of constraint violations decreased and stabilized around the same point. The time and cost showed only a mild reduction. A maximum of two elements could be installed parallelly due to the availability of only 2 cranes. Re-sequencing the elements did not reduce the duration and cost, as the duration distribution of the elements was the same. Further, the schedule duration and cost exhibited higher variability, likely due to the inherent variability in sampled durations and weather conditions, which have a greater impact on these metrics.

Our method significantly outperforms the DQN+GNN [6] based method used in existing studies, delivering superior solution quality, enhanced training stability, and improved execution time. As evident from Figure 6, the total reward and constraint violations identified using PPO+TGN+TGAT were significantly better than DQN+GNN-based solutions. Although the time and cost were quite close, PPO+TGN+TGAT showed better convergence as compared to DQN+GNN. Moreover, for all four criteria, PPO+TGN+TGAT demonstrated superior stability of training. DQN utilizes bootstrapping by estimating Q-function, which can be highly unstable [17], especially with large state space created in the current problem due to uncertainty. Since the parameter updates in DQN solely rely on Mean

Squared Error loss, small overestimation or underestimation of Q-values can lead to large gradient updates, causing instability in the training. PPO directly optimizes the policy using a clipped loss function, limiting the extent of updates in each iteration and providing better stability in learning [9]. The TGN+TGAT network incorporates the temporal aspect of the schedule and keeps a memory embedding of the actions taken in the previous timesteps, handling the dynamically changing graph better. As a result, the objectives could be minimized to a greater extent as compared to the DQN+GNN-based method. To train the 5000 episodes, DQN+GNN took around 12 hours, while PPO+TGN+TGAT required nearly 6.8 hours. DQN requires the management of a replay buffer and target network updates, which are not needed in PPO, resulting in lower execution time for PPO+TGN+TGAT in the validation experiment. However, further work is needed to reduce computational demands considering the near-real-time application requirements.

The results of sensitivity analysis using scenarios in Table 4 are demonstrated in Figure 7. The results show the impact of variation in reward coefficient scales. Since the time and cost varied within a very narrow range, the variation in the total reward mainly depended upon r_{w6}. As a result, the total reward for S2, S4, and S7 was found to be mostly consistent. For the rest 4 of the scenarios, an evident increment in the reward was found. S1 performed generally well across all three objectives as the reward coefficients were decided to provide relatively similar weightage to all three objectives.

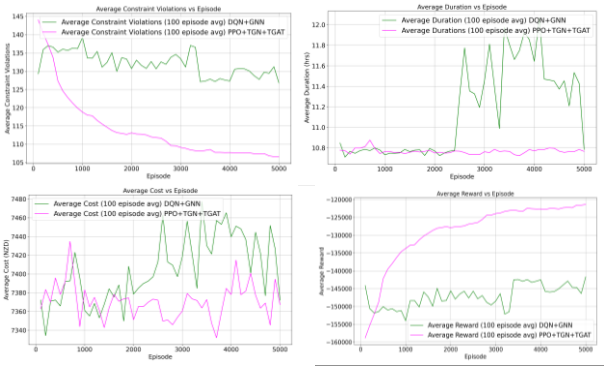

Figure 6. Validation results on scenario S1

5 Conclusion and Outlook

This study proposes a novel method for automated multi-objective ASPS optimization in PCB projects under weather and activity duration uncertainties using RL. PPO, a popular actor-critic RL algorithm, is combined with MCS to train an agent to generate assembly schedules minimizing time, cost, and constraint

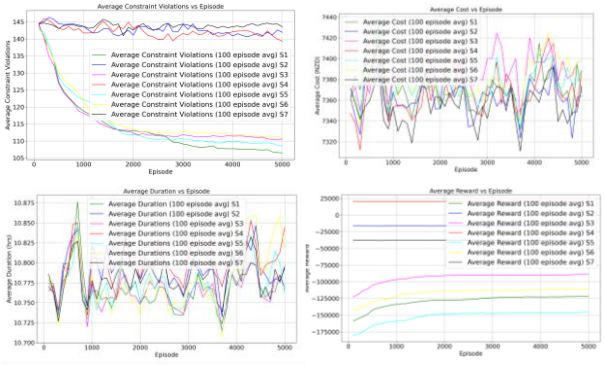

Figure 7. Reward coefficient sensitivity analysis results

violations. Activity durations and weather conditions are sampled at each episode via MCS, enabling the agent to learn optimal policies under uncertainties. A novel PPO network architecture is proposed for ASPS, utilizing TGAT and TGN to create the embeddings of the schedule generated at each timestep. Additional state information is encoded using FC layers, while the actor-critic network consists of 2 shared and 2 separate FC layers. The method is validated on a hypothetical project involving the assembly of 34 precast walls. The case study results show that the proposed method effectively identifies assembly schedules that minimize cost, time, and constraint violations while maximizing rewards under uncertainties. It outperformed the conventional DQN+GNN-based method used in existing RL-based construction scheduling studies.

The study has limitations that need to be addressed to improve the robustness and industry applicability of the proposed method. First, validation using real-world, complex PCB project data is required. A systematic identification of optimal reward coefficients for better results should be explored. Finally, the performance of the developed method should be compared with other RL algorithms, such as Deep Deterministic Policy Gradient (DDPG), Double DQN, and Soft Actor Critic (SAC), etc.

6 References

[1] 4 practical solutions to the global housing crisis. World Economic Forum. Jun. 10, 2024. On-line: https://www.weforum.org/stories/2024/06/global-housing-crisis-practical-solutions/, Accessed: 12/13/2024.

[2] Y. Fang, S. Gao, Y. Jiang, and S. Li. BIM and lean construction in prefabricated housing construction in China. *International Journal of Lean Six Sigma*, 14(7): 1329–1353, 2023.

[3] Y. Wang, Z. Yuan, and C. Sun. Research on assembly sequence planning and optimization of

precast concrete buildings. *Journal of Civil Engineering and Management*, 24(2): 2018.

[4] B. Yang, S. Jiang, M. Dong, D. Zhu, and Y. Han. Graph Database and Matrix-Based Intelligent Generation of the Assembly Sequence of Prefabricated Building Components. *Applied Sciences*, 13(17): 2023.

[5] C. Liu, F. Zhang, H. Zhang, Z. Shi, and H. Zhu. Optimization of assembly sequence of building components based on simulated annealing genetic algorithm. *Alexandria Engineering Journal*, 62: 257–268, 2023.

[6] Y. Yao, V. W. Y. Tam, J. Wang, K. N. Le, and A. Butera. Automated construction scheduling using deep reinforcement learning with valid action sampling. *Automation in Construction*, 166: 105622, 2024.

[7] N. S. Kedir, S. Somi, A. R. Fayek, and P. H. D. Nguyen. Hybridization of reinforcement learning and agent-based modeling to optimize construction planning and scheduling. *Automation in Construction*, 142: 104498, 2022.

[8] Z. Yuan *et al.* Simulation and optimization of prefabricated building construction considering multiple objectives and uncertain factors. *Journal of Building Engineering*, 86: 108830, 2024.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. arXiv. Aug. 28, 2017.

[10] H. Liu, Z. Lei, H. X. Li, and M. Al-Hussein. An Automatic Scheduling Approach: Building Information Modeling-based Onsite Scheduling for Panelized Construction, 1666–1675, 2014.

[11] V. Faghihi, K. F. Reinschmidt, and J. H. Kang. Construction scheduling using Genetic Algorithm based on Building Information Model. *Expert Systems with Applications*, 41(16): 7565–7578, 2014.

[12] L. Huang, R. Pradhan, S. Dutta, and Y. Cai. BIM4D-based scheduling for assembling and lifting in precast-enabled construction. *Automation in Construction*, 133: 103999, 2022.

[13] S. Bae, H. Cha, and S. Jiang. Building an Information Modeling-Based System for Automatically Generating the Assembly Sequence of Precast Concrete Components Using a Genetic Algorithm. *Applied Sciences*, 14(4): Art. no.4 2024.

[14] E. Gazioğlu and A. S. Etaner-Uyar. Experimental analysis of a statistical multiploid genetic algorithm for dynamic environments, *Engineering Science and Technology, an International Journal*, 35: 101173, 2022.

[15] R. K. Soman and M. Molina-Solana. Automating look-ahead schedule generation for construction using linked-data based constraint checking and reinforcement learning. *AUTOMATION IN CONSTRUCTION*, 134: 2022.

[16] R. Pellerin, N. Perrier, and F. Berthaut. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 280(2): 395–416, 2020.

[17] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*, Cambridge, Massachussetts: The MIT Press. 2018.

[18] Z. Yuan, Y. Chang, Y. Chen, Y. Wang, W. Huang, and C. Chen. Simulating and optimizing precast wall lifting in prefabricated building construction. *Engineering, Construction and Architectural Management*. 2022.

[19] E. Mohamed, P. Jafari, A. Chehouri, and S. AbouRizk. Simulation-Based Approach for Lookahead Scheduling of Onshore Wind Projects Subject to Weather Risk. *Sustainability*, 13(18): 2021.

[20] T. Homem-de-Mello and G. Bayraksan. Monte Carlo sampling-based methods for stochastic optimization. *Surveys in Operations Research and Management Science*, 19(1): 56–85, 2014.

[21] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan. Inductive Representation Learning on Temporal Graphs. arXiv. Feb. 19, 2020.

[22] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein. Temporal Graph Networks for Deep Learning on Dynamic Graphs. arXiv. Oct. 09, 2020.

[23] A. Vaswani *et al.* Attention Is All You Need. arXiv. Aug. 02, 2023.

[24] Auckland Central Weather Forecast and Observations - MetService New Zealand,. On-line: https://www.metservice.com/towns-cities/regions/auckland/locations/auckland, Accessed: 12/30/2024.

[25] Wind, waves, weather & tide forecast Auckland Airport - Windfinder,. On-line: https://www.windfinder.com/forecast/auckland, Accessed: 12/30/2024.

[26] Safety Wind Speed,. Crane Association of New Zealand (INC). On-line: https://www.cranes.org.nz/uploads/2/0/5/7/20572552/wind4.pdf, Accessed: 12/30/2024.

[27] *Rawlinsons New Zealand Construction Handbook*, Rawlinsons Publications. 2014.