2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Dynamic Crane Scheduling with Reinforcement Learning for a Steel Coil Warehouse

Sang-Hyun Cho†, Woo-Jin Shin†, Jeongsun Ahn, Sanghyun Joo, and Hyun-Jung Kim*, *Senior Member, IEEE*

***Abstract*— This paper tackles the dynamic crane scheduling problem in a steel coil warehouse, involving tasks such as coil storage, retrieval, and shuffling. Tasks arrive dynamically with precedence relations, while multiple cranes share a track, necessitating collision avoidance. We aim to minimize the average task waiting time by allocating tasks to cranes and optimizing their execution sequence. Unlike prior research focusing on static scenarios or rule-based heuristics, we introduce a real-time, reinforcement learning-based algorithm. We propose a policy network based on graph neural networks to effectively handle precedence relations and global information. Experimental results demonstrate its superiority over traditional heuristics, such as dispatching rules in dynamic scenarios.**

I. INTRODUCTION

This study addresses a dynamic multi-crane scheduling problem that arises in a steel coil warehouse. The motivation for this problem stems from iron and steel enterprises, where the production of steel products involves several stages, including iron-making, steel-making, continuous casting, and hot/cold rolling. Between these production stages, there is a need for warehouses to store the output products from the preceding stage and retrieve the products required for the subsequent stage.

The coil warehouse, in particular, serves as a repository for completed or partially finished steel coil products. An example of a coil warehouse is depicted in Fig. 1. Inside the warehouse, coils are stored in multiple rows and can be stacked on top of each other. Handling of the coils is facilitated by overhead cranes operating on a shared track. Cranes should move to designated positions, lift coils, transport them to other locations, and drop them off without colliding with each other. For the transportation of coils between the warehouse and production stages, trucks can be positioned at both ends of the warehouse, allowing them to either receive coils from the cranes or hand over coils to the cranes. Trucks dynamically arrive at the warehouse, resulting in real-time requirements for incoming and outgoing coils within the warehouse. To enhance the overall efficiency of steel production, efficient scheduling of cranes is essential, and it is challenging to meet dynamic requirements while avoiding collisions between cranes.

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education [2022R1A2C4001978, 2022M3J6A1063021], and AI Platform Center for Manufacturing at KAIST(A0801038001).

The authors are with the Department of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea.

†These authors contributed equally.

*Corresponding author. E-mail: hyunjungkim@kaist.ac.kr

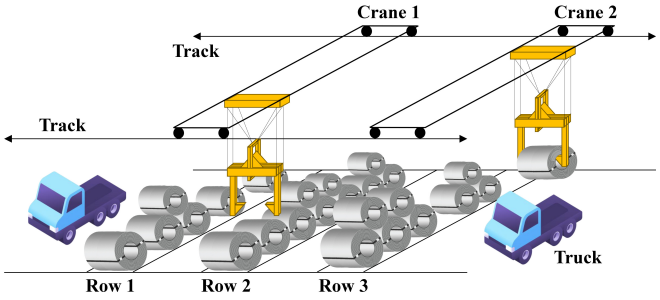

Fig. 1. A steel coil warehouse.

Incoming coils can be stored in empty slots, or outgoing coils from the upper level can be retrieved directly by cranes. On the other hand, when storing coils in slots already occupied or retrieving coils from the lower level, shuffling tasks must be carried out initially to relocate the blocking coils. Due to these blocking issues, required sequences among tasks emerge, leading to the establishment of precedence relations between them.

Steel manufacturing companies' warehouse typically operates in two primary phases [1]. The first phase involves organizing tasks for storing, retrieving, and shuffling products according to the requirements of incoming and outgoing products. During this phase, positions of storage and shuffling of products, as well as the precedence relations among tasks, are determined. Once these tasks are defined, the subsequent phase focuses on crane scheduling. This includes allocating tasks to multiple cranes and determining their execution sequence for efficient operation of cranes.

Compared to warehouses storing products like steel slabs, which can be stacked to greater heights, coils are limited to stacking up to two levels due to stability concerns [2]. Consequently, determining positions and precedence relations for coils is relatively straightforward. However, coil warehouses typically operate with multiple cranes, making crane scheduling more complicated [3]. Therefore, in this study, we focus on resolving crane scheduling, assuming that the positions and precedence relations of tasks are predetermined by the first phase process.

The entire process of steel production operates continuously, and as a result, the transportation of coils between the warehouse and production stages occurs dynamically. Trucks arrive at the warehouse either loaded with coils or empty to receive coils, triggering the real-time scheduling of cranes accordingly. However, previous studies on crane scheduling problems have mainly focused on static settings to derive

high-quality solutions at the expense of longer computation times. Consequently, these studies face challenges when applied to dynamic environments that require quick decision-making. Moreover, in dynamic scheduling, dispatching rules are commonly used to make rapid decisions but often result in unsatisfactory solution quality.

Recent advancements in machine learning, particularly reinforcement learning (RL), have demonstrated their effectiveness in solving combinatorial optimization problems [4]–[8]. Consequently, we propose an RL-based algorithm that offers advantages in both rapid scheduling and solution quality while being adaptable to dynamic scenarios. To effectively embed global information and dynamic features about tasks, cranes, and precedence relations, we introduce a policy network based on graph neural networks (GNNs) that are trained in an end-to-end fashion. In our approach, the agent allocates a task for an idle crane at each state, with the objective of minimizing the average waiting time of tasks. Through experiments, we confirm that our approach outperforms handcrafted heuristics, such as dispatching rules, previously used in dynamic crane scheduling.

II. Related Works

Crane scheduling is a critical aspect of various industries, including container terminals, factories, and warehouses, where cranes are extensively used as material handling equipment. Efficient scheduling of cranes is vital for overall system productivity and is known to be highly complicated. Research in crane scheduling initially focused on static problems, with significant attention on quay cranes and yard cranes in container terminals. Subsequently, research expanded to problems in factories and warehouses. In recent years, there has been a growing interest in tackling the challenges of crane scheduling in dynamic environments.

The quay crane scheduling problem involves determining crane schedules for loading and unloading containers to and from vessels berthed at a port. Kim and Park's pioneering work [9] for multiple quay cranes has encouraged extensive subsequent research in this field. The primary objective of quay crane scheduling is to minimize the makespan, and numerous exact solution methods and heuristics have been developed [10]–[14]. However, unlike warehouse cranes, quay cranes operate at single locations, making it difficult to adapt those studies to the problems in warehouses.

For yard cranes, the scheduling problem involves addressing crane operations for container transfer between yards and trucks. The nature of tasks depends on the yard layout, and they can be either single-location or delivery-based tasks. In cases of single-location tasks, the problem resembles the quay crane scheduling problem, and various studies have been conducted [15]–[18]. When cranes perform delivery-based tasks, they are typically operated as twin cranes, emphasizing cooperative operations between them [19]–[22].

For warehouses of steel coils, Zapfel and Wasner [23] initially studied the crane scheduling problem, followed by research for a single crane in [2], [24]–[26]. Subsequent studies in [27], [28] addressed multi-crane scheduling in static environments. Additionally, research has been conducted on other various warehouses, including studies on slab yard cranes [1], [29]–[31] and automated storage/retrieval systems [32], [33], as well as factory cranes [34]–[37].

While research on static problems has seen significant progress, studies addressing dynamic environments have been relatively scarce. Recently, a few research efforts have focused on dynamic factory environments. For steel-making shops, dynamic crane scheduling problems are studied in [38], [39], considering the realistic characteristics of the steel-making process. In addition, there are studies utilizing RL, aiming to develop approaches for factory cranes in flexible manufacturing systems [40]–[42]. In these approaches, agents select one of the given heuristic rules at each state.

This study addresses the dynamic crane scheduling problem arising in coil warehouses and proposes an RL-based approach capable of providing real-time superior solutions for dynamically occurring requests. Our approach differs from previous research on static settings in its understanding of system dynamicity and its ability to deliver high-quality solutions quickly. Additionally, it distinguishes itself from dispatching rules and existing RL approaches as it allows each crane to perform any possible task flexibly rather than being restricted to predefined rules.

III. Problem Description

The warehouse consists of R rows, where rows 1 and R serve as the I/O points positioned by trucks, and rows 2 to $R-1$ are storage locations for coils. There are multiple cranes operating on a single track, and the track is positioned in the direction that crosses the rows (refer to Fig. 1). Each row can accommodate just one crane, and all cranes are capable of accessing every row. Since the movement of the hoist within a single crane is independent of crane collision, we only consider crane movements in the direction of the track for the sake of simplification. Moving a crane across one row takes $\hat{t}$ units of time, and handling time $\bar{h}$ is required for cranes to pick up or drop coils.

Individual coil transportation within the warehouse is defined as a task. Storage tasks involve moving coils from the I/O point to storage locations, retrieval tasks contain the opposite movement, and shuffling tasks entail moving coils within the storage area. For each task τ, source and target locations (l^s_τ and l^f_τ) are specified, and a duration $d_\tau (= |l^s_\tau - l^f_\tau| \cdot \hat{t} + 2 \cdot \bar{h})$ is required to complete task τ. The arrival time of task τ is denoted as a_τ and is not known in advance. Moreover, we assume that the time intervals between task arrivals follow an exponential distribution.

There can be a precedence relation between two tasks, and when task τ precedes task τ', it is denoted as $\tau \prec \tau'$. Depending on the locations of two tasks, the relation can be defined in one of four different types, as illustrated in Fig. 2: In the first type, the succeeding task can be lifted up after the preceding task has been lifted up; in the second, the succeeding one can be dropped off after the preceding has been dropped off; the third allows the succeeding to be dropped off at the position where the preceding was lifted

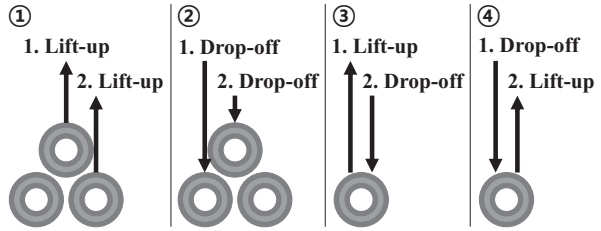

Fig. 2. Four types of precedence relations between tasks.

up, and the last one indicates that the task can be lifted up again after it has been dropped off.

The waiting time of a task is the gap between its arrival time and completion time. Our objective is to minimize the average waiting time of tasks, which is computed as follows, where T is a set of tasks and s_τ is the start time of task τ.

$$\sum_{\tau \in T} (s_\tau + d_\tau - a_\tau)/|T| \tag{1}$$

IV. Proposed Approach

In this section, we describe our RL-based approach. We define the state as a graph and develop a policy network based on graph attention networks v2 (GATv2) [43]. Through a policy gradient method, we train the network to maximize the cumulative reward, which aligns with our objective of minimizing the average waiting time.

A. States, Actions, and Rewards

1) States: To effectively transfer the status of a warehouse to the agent, we define the state x_t at time step t as follows.

Definition 1: $x_t = (G_t(V_t, E_t), g_t)$ where

- $G_t(V_t, E_t)$ indicates a graph where V_t and E_t are the set of nodes and edges, respectively.
- $V_t = V_t^a \cup V_t^b$ represents the node set at time step t, defined as the union of two sets as defined below.
- $V_t^a = \{v_1^a, ..., v_\tau^a, ..., v_n^a\}$ is the set of nodes for arrived and unfinished tasks until t, where n is the number of those tasks. $h_\tau^a = [l_\tau^s, l_\tau^f, d_\tau, ty_\tau, as_\tau^a, st_\tau^a, rt_\tau^a, wt_\tau]$ denotes the feature vector of node v_τ^a, for task τ, where l_τ^s and l_τ^f are source and target locations, d_τ is duration, ty_τ is the one-hot vector that indicates a type of the task among storage, retrieval, or shuffling, as_τ^a is 1 if the task is assigned to a crane, otherwise 0, st_τ^a is 1 if the task has started, otherwise 0, rt_τ^a is the remaining time for the task to complete, and wt_τ is the waiting time. If τ is not assigned, then rt_τ^a and wt_τ are both set to 0.
- $V_t^b = \{v_1^b, ..., v_c^b, ..., v_m^b\}$ is the set of nodes for cranes at t, where m is the number of all cranes. $h_c^b = [l_c, as_c^b, st_c^b, rt_c^b, l_c^s, l_c^f]$ denotes the feature vector of node v_c^b, for crane c, where l_c is the current position, as_c^b is 1 if there is an assigned task, otherwise 0, st_c^b is 1 if there is an assigned task that has started, otherwise 0, rt_c^b represents the remaining time to complete the assigned task, and l_c^s and l_c^f are source and target locations of the assigned task, respectively. If there is no assigned task, then rt_c^b, l_c^s, and l_c^f are set to 0.
- $E_t = E_t^1 \cup E_t^2 \cup E_t^3 \cup E_t^4$ indicates the set of directed edges at t. E_t^1 and E_t^2 are the sets of edges connecting from v_τ^a to v_c^b and from v_c^b to v_τ^a for all τ and c, respectively. E_t^3 is the set of edges from v_τ^a to $v_{\tau'}^a$ for all precedence relations $\tau \prec \tau'$. E_t^4 includes the opposite-direction edges for the edges in E_t^3, specifically designed to enable bidirectional information flow. Only for E_t^3 and E_t^4, each edge has an attribute of the one-hot vector, representing one of the four types of precedence relations depicted in Fig. 2.
- $g_t = [uf_t, ua_t, um_t]$ represents the global feature vector of current tasks at t. uf_t, ua_t, and um_t denote the number of unfinished tasks, unassigned tasks, and available tasks (unassigned tasks among which preceding tasks have all been assigned), respectively.

The state includes dynamic features, and the number of tasks can vary, causing changes in the number of nodes. Our network model is based on GNNs, where embedding is defined for pairs of nodes. This aspect allows us to apply the identical model to graphs of different sizes.

Furthermore, to ensure that the model effectively captures precedence relations, we decompose the set of edges into four categories, each processed with different parameters within the model. We also define edge attributes to allow the model to distinguish the types of relations, expediting the learning process. Additionally, to enhance generalization performance, we define global features that summarize the overall information of tasks at the current state.

2) Actions: The action space is defined as a set of task-crane pairs. An action (τ, c) assigns task τ to crane c. Tasks are executed by cranes in the sequence they are assigned. Thus, a task can only be handled if all preceding tasks have already been assigned. Any action involving a task with at least one preceding task not yet assigned is masked, meaning it cannot be chosen by the agent. Similarly, actions involving a crane that is not currently idle are also masked.

3) State transition and rewards: The environment receives an action and provides the next state along with a reward. The action schedules the crane to start the task at the earliest possible time, calculated by considering factors like crane travel time, task precedence relations, and crane collisions. Cranes avoid collisions based on the sequence tasks are assigned, ensuring subsequent cranes move along collision-free paths if conflicts occur. The clock for the next state is set to the time when there is at least one available task and one idle crane. Up to that time, each crane executes its allocated task. Finally, the environment returns the current information as the next state with a reward.

The reward r_t for an action (τ, c) at time step t is the negative value of the waiting time for task τ, determined by the following equation.

$$r_t = -(s_\tau + d_\tau - a_\tau) \tag{2}$$

Here, the task start time s_τ is set to the earliest possible time mentioned earlier and calculated through the simulation in our environment.

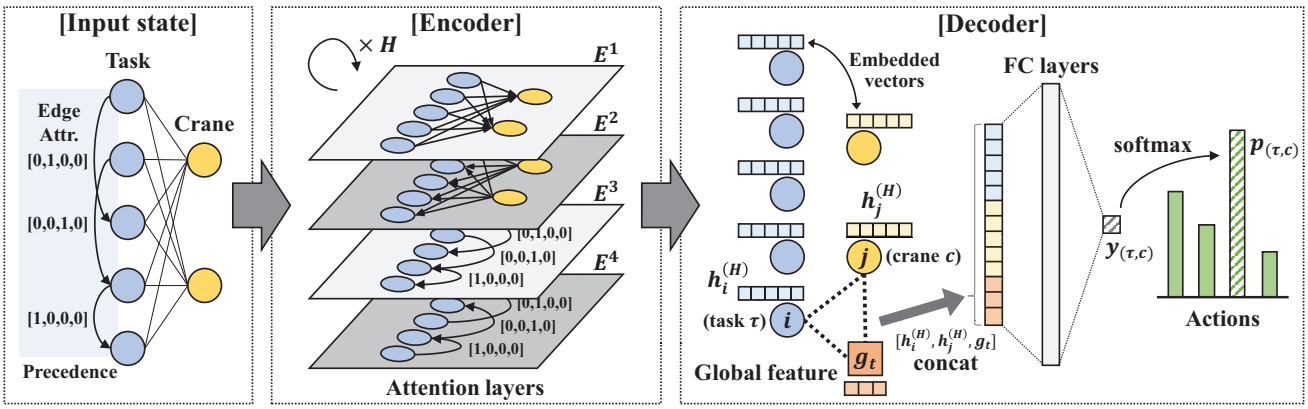

Fig. 3. The structure of our policy network.

B. Policy Network

The proposed policy network consists of an encoder that embeds the state and a decoder that generates the selection probability of actions based on the embedded features (refer to Fig. 3).

1) Encoder: The encoder generates hidden vectors for each node in the state x_t represented by the graph. First, for the initial features of nodes in V_t^a and V_t^b, the following process transforms them into vectors of the dimension d^h.

$$h_i = \text{ReLU}(W_0 \cdot h_i^0) \tag{3}$$

Here, h_i^0 is any of h_τ^a and h_c^b, and W_0 is a learnable matrix. For nodes in V_t^a and V_t^b, we utilize distinct W_0.

After that, to effectively handle each of edge relationships and incorporate edge attributes, we construct four different sub-graphs, each composed of sets E_t^k and V_t. For each sub-graph, we apply the following attention layers (GATv2):

$$e^k(h_i, h_j) = a^{k^T}\text{LeakyReLU}(W_1^k \cdot [h_i, h_j, e_{ij}]) \tag{4}$$

$$\alpha_{ij}^k = \text{softmax}_j(e^k(h_i, h_j)) = \frac{\exp(e^k(h_i, h_j))}{\sum_{j' \in N_i^k} \exp(e^k(h_i, h_{j'}))} \tag{5}$$

$$h_i^{k(l)} = \text{LeakyReLU}\left(\sum_{j \in N_i^k} \alpha_{ij}^k \cdot W_2^k h_j\right) \tag{6}$$

where e_{ij} is the one-hot vectored edge attribute and only used for $k = 3$ and 4. W_1^k, W_2^k, and a^k are learnable parameters, N_i^k is the set of node i and its neighbor nodes coming into i according to E^k, and $[\cdot, \cdot]$ indicates vertical concatenation.

With $h_i^{k(l)}$ for all k, we can obtain the final hidden vector for each node as follows:

$$h_i^{(l)} = \text{ReLU}(h_i \oplus W_3 \cdot [h_i^{1(l)}, h_i^{2(l)}, h_i^{3(l)}, h_i^{4(l)}]) \tag{7}$$

where l represents the number of repetitions of the above process, $\oplus$ is element-wise addition, and W_3 is a learnable matrix. The processes described in (4)-(7) are repeated total H times, obtaining final hidden vectors for all nodes in both V_t^a and V_t^b.

2) Decoder: Decoding transforms the final hidden vectors into scalar values, representing the selection probability for actions. To incorporate the global features, we propose the decoding layer as follows.

$$y_{(\tau,c)} = W_5 \cdot \text{ReLU}(W_4 \cdot [h_i^{(H)}, h_j^{(H)}, g_t]) + \text{mask}_{(\tau,c)} \tag{8}$$

$$p_{(\tau,c)} = \text{softmax}_{(\tau,c)}(y_{(\tau,c)}) \tag{9}$$

Here, i and j denote task τ and crane c nodes, respectively, g_t is the global feature, $\text{mask}_{(\tau,c)}$ is masking process, and W_4 and W_5 are learnable matrices. The agent uses selection probability $p_{(\tau,c)}$ to sample actions during training. Once the policy is fully trained, it makes decisions by selecting actions with the highest probability, deriving high-quality decisions.

C. Training Algorithm

To train our policy, we utilize REINFORCE [44], known as one of the most effective policy gradient methods. We set the baseline b_k as the cumulative reward from step k, divided by the number of repetitions M. This baseline, as introduced in [45], is computed with the same task arrival sequence to reduce the variance caused by the randomness in the arrival, stabilizing the learning process in dynamic problems. The detailed procedure is outlined in Algorithm 1.

Algorithm 1 Policy gradient method

for each update step **do**
  $\Delta_\theta \leftarrow 0$, sample B instances
  for each instance **do**
    Run episodes $i = 1, ..., M$:
    $\{s_1^i, a_1^i, r_1^i, ..., s_N^i, a_N^i, r_N^i\} \sim \pi_\theta$
    Compute total reward: $R_k^i = \sum_{k'=k}^N \gamma^{k'-k} r_{k'}^i$
    for $k = 1$ to N **do**
      Compute baseline: $b_k = \frac{1}{M}\sum_{i=1}^M R_k^i$
      for $i = 1$ to M **do**
        $\Delta_\theta \leftarrow \Delta_\theta + \frac{1}{BM}\nabla_\theta \log \pi_\theta(s_k^i, a_k^i)(R_k^i - b_k)$
      end for
    end for
  end for
  $\theta \leftarrow \theta + \alpha\Delta_\theta$
end for

TABLE I

EXPERIMENTAL RESULTS ON VALIDATION INSTANCES

Method	2 cranes						3 cranes					
	N=100		N=150		N=200		N=100		N=150		N=200	
	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap
SPT-NN	163.4	0.98	187.5	1.09	194.0	1.19	298.6	1.38	332.3	1.82	357.1	2.08
GNS-NN	171.5	1.08	198.4	1.21	204.1	1.31	333.2	1.66	358.7	2.04	392.3	2.38
GNS-MTD	163.8	0.99	194.6	1.17	197.3	1.23	287.9	1.30	318.9	1.71	347.9	2.00
FCFS-NN	172.8	1.10	198.5	1.22	199.7	1.26	336.5	1.69	373.1	2.17	416.9	2.60
FCFS-MTD	150.4	0.82	178.3	0.99	187.0	1.12	282.2	1.25	324.7	1.76	355.0	2.06
NN	169.6	1.06	191.2	1.13	195.5	1.21	325.9	1.60	352.7	1.99	390.8	2.37
MTD	140.4	0.70	167.9	0.87	171.3	0.94	266.5	1.13	285.4	1.42	326.3	1.82
ZC-NN	101.6	0.23	110.9	0.24	110.6	0.25	185.1	0.48	174.5	0.48	185.5	0.60
ZC-MTD	92.4	0.12	101.3	0.13	101.6	0.15	142.2	0.13	137.7	0.17	138.9	0.20
SimECT	100.2	0.21	110.6	0.23	110.0	0.24	148.0	0.18	145.5	0.24	145.3	0.25
Proposed RL	**82.5**	**0**	**89.6**	**0**	**88.4**	**0**	**125.3**	**0**	**117.8**	**0**	**115.9**	**0**

V. NUMERICAL EXPERIMENTS

The experimental results of our RL approach are proposed in comparison to various algorithms.

A. Settings

1) Instance Generation: Instances have 2 or 3 cranes, with R set to 20 or 30, respectively, and a total of 100, 150, or 200 tasks (N). Storage, retrieval, and shuffling ratios among tasks remain consistent. Source and target locations, excluding I/O points, are randomly set, with precedence relations specified when tasks share locations. Arrival intervals follow an exponential distribution, with means of 115 or 130 for 2 or 3 cranes, respectively, and the number of new tasks at each arrival follows a normal distribution, with a mean of 7 and a standard deviation of 1. At time 0, there are 10 initial tasks. Both $\hat{t}$ and $\bar{h}$ are set to 1. These parameters are carefully chosen to balance task arrival and completion rates while using the best-performing heuristic.

2) Model Parameters: d^h, H, M, α, and γ are set to 64, 3, 10, 5×10^{-3}, and 1 respectively. Training lasts for 500 epochs, each with one gradient update. Since the simulation computation is the bottleneck, training is performed on the CPU, Intel Core i7-12700 with 128.00 GB of RAM. B is set to 20, matching the number of processors on the CPU.

3) Comparison Algorithms: Our approach is compared with heuristics applicable for dynamic settings such as the nearest neighbor (NN) [19], minimum travel distance (MTD) [46], the shortest processing time (SPT) [47], the greatest number of successors (GNS) [48], and first-come first-served (FCFS) [19]. We also designed several heuristics for more comparison: zone control (ZC) and the earliest completion time rule using simulations (SimECT). ZC is a commonly used operational strategy in real fields, while SimECT is a heuristic that utilizes exact completion times of tasks obtained through simulation. Details follow:

- NN: For all task τ and crane c, selects the pair with minimum $|l^s_\tau - l_c|$ as the action.
- MTD: For all task τ and crane c, selects the pair with minimum $|l^s_\tau - l_c| + d_\tau$ as the action.
- SPT-NN: For pairs of tasks with minimum d_τ and all cranes, apply NN.
- GNS-NN(-MTD): For pairs of tasks with the greatest number of successors and all cranes, apply NN(MTD).
- FCFS-NN(-MTD): For pairs of tasks with minimum a_τ and all cranes, apply NN(MTD).
- ZC-NN, ZC-MTD: For pairs of tasks using the left(right) I/O point and the leftmost(rightmost) crane, and other tasks and all cranes, apply NN or MTD.
- SimECT: Using simulation, for all tasks and cranes, select the pair that can complete the task earliest, considering crane conflicts.

TABLE II

GENERALIZATION ABILITY (2 CRANES TRAINED, 3 CRANES TESTED)

Method	N =100		N =150		N =200	
	Obj.	Gap	Obj.	Gap	Obj.	Gap
ZC-MTD	142.2	0.11	137.7	0.16	138.9	0.17
w/o global Feat.	133.5	0.04	124.0	0.05	125.7	0.06
Proposed RL	**128.3**	**0**	**118.2**	**0**	**119.0**	**0**

We also compared our approach with the mixed integer programming (MIP) approach using ILOG CPLEX 22.1. The static version of our problem was formulated as MIP based on [49]. We generated 30 instances with parameters from A.1), each featuring 2 cranes and only 10 tasks. With a 10-minute time limit per instance, only 4 out of 30 produced feasible solutions, with an average relative gap of 52%. Given the challenges of achieving feasible solutions within given solution times, we excluded further comparisons with MIP in subsequent experiments. Detailed formulations and tables are omitted due to space constraints.

B. Experimental Results

1) Overall performance: In Table I, we present the overall performance of the proposed RL approach and other heuristics. We conducted test experiments for scenarios with 2 and 3 cranes, each with N = 100, 150, and 200. The RL approach was trained on instances of N = 100 only, for both scenarios. The 'Obj.' column shows the average objective value over 100 validation instances for each method, with the 'Gap' indicating the ratio compared to the RL approach.

As shown in Table I, our RL approach consistently outperforms all others. Heuristics based on zone control or

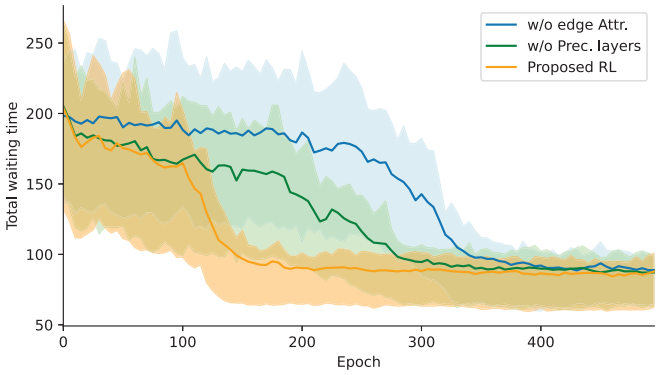

Fig. 4. Ablation study of model components on performance during training (in the range of the 1st to 3rd quartiles).

TABLE III
MODEL ABLATION TEST RESULTS

Method	N =100		N =150		N =200	
	Obj.	Gap	Obj.	Gap	Obj.	Gap
w/o edge Attr.	86.7	0.05	93.2	0.04	93.5	0.06
w/o Prec. layers	85.5	0.04	91.8	0.02	91.4	0.03
Proposed RL	**82.5**	**0**	**89.6**	**0**	**88.4**	**0**

simulation perform relatively well among the comparison algorithms, with ZC-MTD having the best performance, although it remains more than 12% less competitive than the RL approach. We also observe that when the number of tasks or cranes grows, the gap between the RL approach and others gets bigger. Notably, our approach takes only milliseconds to make a decision, demonstrating the potential to make high-quality real-time decisions in dynamic environments. The training lasted about 50 and 80 hours for 2 cranes and 3 cranes scenarios, respectively.

2) Generalization performance: To evaluate our approach's generalization performance, we trained it with the scenario of 2 cranes and $N = 100$ and tested it in the scenario involving 3 cranes. The results are presented in Table II. We can confirm that the proposed approach outperforms the best comparative heuristic, ZC-MTD, by at least 11%, indicating superior generalization performance.

The model labeled 'w/o global Feat.' is trained without global features in the network model, resulting in an approximate 5% performance degradation compared to the proposed approach. This confirms the significant role of global features in enhancing the model's generalization capability. In the absence of global features, the model relies solely on node connections to estimate the graph's scale. However, this process may yield inaccurate results when the number of crane nodes deviates from the training data. Therefore, the direct inclusion of graph scale information through global features emerges as a crucial factor with a substantial impact.

3) Evaluation of model components: Fig. 4 depicts the performance of RL approaches during the training for the 2-crane scenario with $N = 100$. 'w/o edge Attr.' is a model that excludes the use of edge attributes, and 'w/o Prec. layers' is a model without edge attributes as well as attention layers related to precedence relations. It is worth noting that 'w/o Prec. layers' has a significantly lower number of learnable parameters compared to others. As can be seen in Fig. 4, we find that the proposed model exhibits the fastest convergence. Furthermore, it is evident that incorporating edge attributes with attention layers for precedence relations is crucial, as 'w/o edge Attr.' converges more slowly than 'w/o Prec. layers'. Although the types of precedence relations can be inferred using task locations included in the node features, the proposed model benefits from explicit information through edge attributes, resulting in faster convergence and performance improvement. The relevant evaluation results are summarized in Table III.

TABLE IV
TEST RESULTS FOR VARYING INSTANCE GENERATION PARAMETERS

Method	ω×1.5		ω×2.0		μ×1.5		μ×2.0	
	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap
ZC-MTD	112.7	0.11	154.6	0.24	200.5	0.13	241.1	0.05
Proposed RL	**101.7**	**0**	**124.3**	**0**	**178.2**	**0**	**229.7**	**0**

4) Robustness on instance parameters: We evaluate the performance of our RL approach by varying parameters for the instance generation, and the results are presented in Table IV. As the number of initial tasks (ω) increases, the performance gap between the RL approach and the best heuristic, ZC-MTD, grows. This suggests that the RL approach can make better decisions when it has more available actions to choose at each state. On the other hand, an increase in arrival size (μ) reduces the performance gap, indicating that as scenarios become more extreme with a rapidly increasing number of tasks, the difference in strategies between methods diminishes further. Similar results are observed in instances where the task arrival interval is shortened. Overall, these findings confirm that our RL approach consistently performs well under various conditions.

VI. CONCLUSION

In this study, we proposed an RL-based approach to address the dynamic multi-crane scheduling problem in a coil warehouse. The state of the warehouse environment was represented using a graph, and a policy network utilizing graph attention networks was developed to effectively process it. The network was designed to handle precedence relations and global information to accelerate training convergence and improve generalization performance. Through numerical experiments, we confirmed that our RL approach outperforms other handcrafted heuristics and maintains strong performance on unseen instances. Also, the RL approach can derive high-quality decisions within a short time, making it applicable to online warehouse operations.

For future work, we plan to explore approaches based on fine-tuning to achieve significantly higher performance in out-of-distribution scenarios. Additionally, we aim to extend the problem scope through location assignment, making our research applicable to general warehouse settings.

REFERENCES

[1] A. Dohn and J. Clausen, "Optimising the slab yard planning and crane scheduling problem using a two-stage heuristic," *International Journal of Production Research*, vol. 48, no. 15, pp. 4585–4608, 2010.

[2] Y. Yuan and L. Tang, "Novel time-space network flow formulation and approximate dynamic programming approach for the crane scheduling in a coil warehouse," *European Journal of Operational Research*, vol. 262, no. 2, pp. 424–437, 2017.

[3] A. Beham, S. Raggl, J. Karder, B. Werth, and S. Wagner, "Dynamic warehouse environments for crane stacking and scheduling," *Procedia Computer Science*, vol. 200, pp. 1461–1470, 2022.

[4] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" *arXiv preprint arXiv:1803.08475*, 2018.

[5] Y.-D. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, and S. Min, "Pomo: Policy optimization with multiple optima for reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 188–21 198, 2020.

[6] Y.-D. Kwon, J. Choo, I. Yoon, M. Park, D. Park, and Y. Gwon, "Matrix encoding networks for neural combinatorial optimization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5138–5149, 2021.

[7] W. Kool, H. van Hoof, J. Gromicho, and M. Welling, "Deep policy dynamic programming for vehicle routing problems," in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 2022, pp. 190–213.

[8] M. Kim, J. Park, and J. Park, "Sym-nco: Leveraging symmetricity for neural combinatorial optimization," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1936–1949, 2022.

[9] K. H. Kim and Y.-M. Park, "A crane scheduling method for port container terminals," *European Journal of Operational Research*, vol. 156, no. 3, pp. 752–768, 2004.

[10] L. Moccia, J.-F. Cordeau, M. Gaudioso, and G. Laporte, "A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal," *Naval Research Logistics*, vol. 53, no. 1, pp. 45–59, 2006.

[11] M. Sammarra, J.-F. Cordeau, G. Laporte, and M. F. Monaco, "A tabu search heuristic for the quay crane scheduling problem," *Journal of Scheduling*, vol. 10, pp. 327–336, 2007.

[12] C. Bierwirth and F. Meisel, "A fast heuristic for quay crane scheduling with interference constraints," *Journal of Scheduling*, vol. 12, pp. 345–360, 2009.

[13] C. Bierwirth and F. Meisel, "A follow-up survey of berth allocation and quay crane scheduling problems in container terminals," *European Journal of Operational Research*, vol. 244, no. 3, pp. 675–689, 2015.

[14] Z. Zhang, M. Liu, C.-Y. Lee, and J. Wang, "The quay crane scheduling problem with stability constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 1399–1412, 2018.

[15] W. Ng, "Crane scheduling in container yards with inter-crane interference," *European Journal of Operational Research*, vol. 164, no. 1, pp. 64–78, 2005.

[16] W. Li, Y. Wu, M. Goh, W. Li, Y. Wu, and M. Goh, *A continuous-time model for multiple yard crane scheduling with last-minute job arrivals*. Springer, 2015.

[17] P. Guo, W. Cheng, Y. Wang, and N. Boysen, "Gantry crane scheduling in intermodal rail-road container terminals," *International Journal of Production Research*, vol. 56, no. 16, pp. 5419–5436, 2018.

[18] F. Zheng, X. Man, F. Chu, M. Liu, and C. Chu, "Two yard crane scheduling with dynamic processing time and interference," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3775–3784, 2018.

[19] A. H. Gharehgozli, G. Laporte, Y. Yu, and R. De Koster, "Scheduling twin yard cranes in a container block," *Transportation Science*, vol. 49, no. 3, pp. 686–705, 2015.

[20] D. Briskorn, S. Emde, and N. Boysen, "Cooperative twin-crane scheduling," *Discrete Applied Mathematics*, vol. 211, pp. 40–57, 2016.

[21] D. Kress, J. Dornseifer, and F. Jaehn, "An exact solution approach for scheduling cooperative gantry cranes," *European Journal of Operational Research*, vol. 273, no. 1, pp. 82–101, 2019.

[22] H. Fan, W. Peng, M. Ma, and L. Yue, "Storage space allocation and twin automated stacking cranes scheduling in automated container terminals," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14 336–14 348, 2021.

[23] G. Zäpfel and M. Wasner, "Warehouse sequencing in the steel supply chain as a generalized job shop model," *International Journal of Production Economics*, vol. 104, no. 2, pp. 482–501, 2006.

[24] R. J. Rei, M. Kubo, and J. P. Pedroso, "Simulation-based optimization for steel stacking," in *Modelling, Computation and Optimization in Information Systems and Management Sciences: Second International Conference MCO 2008, Metz, France-Luxembourg, September 8-10, 2008. Proceedings*. Springer, 2008, pp. 254–263.

[25] L. Tang, X. Xie, and J. Liu, "Scheduling of a single crane in batch annealing process," *Computers & Operations Research*, vol. 36, no. 10, pp. 2853–2865, 2009.

[26] L. Tang, X. Xie, and J. Liu, "Crane scheduling in a warehouse storing steel coils," *IIE Transactions*, vol. 46, no. 3, pp. 267–282, 2014.

[27] X. Xie, Y. Zheng, and Y. Li, "Multi-crane scheduling in steel coil warehouse," *Expert Systems with Applications*, vol. 41, no. 6, pp. 2874–2885, 2014.

[28] G. N. Maschietto, Y. Ouazene, M. G. Ravetti, M. C. de Souza, and F. Yalaoui, "Crane scheduling problem with non-interference constraints in a steel coil distribution centre," *International Journal of Production Research*, vol. 55, no. 6, pp. 1607–1622, 2017.

[29] X. Cheng, L. Tang, and P. M. Pardalos, "A branch-and-cut algorithm for factory crane scheduling problem," *Journal of Global Optimization*, vol. 63, pp. 729–755, 2015.

[30] G. Zhao, J. Liu, L. Tang, R. Zhao, and Y. Dong, "Model and heuristic solutions for the multiple double-load crane scheduling problem in slab yards," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1307–1319, 2019.

[31] X. Wang, M. Zhou, Q. Zhao, S. Liu, X. Guo, and L. Qi, "A branch and price algorithm for crane assignment and scheduling in slab yard," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1122–1133, 2020.

[32] Y. Kung, Y. Kobayashi, T. Higashi, M. Sugi, and J. Ota, "Order scheduling of multiple stacker cranes on common rails in an automated storage/retrieval system," *International Journal of Production Research*, vol. 52, no. 4, pp. 1171–1187, 2014.

[33] S. Heshmati, T. A. Toffolo, W. Vancroonenburg, and G. V. Berghe, "Crane-operated warehouses: Integrating location assignment and crane scheduling," *Computers & Industrial Engineering*, vol. 129, pp. 274–295, 2019.

[34] X. Xie, Y. Li, and Y. Zheng, "Multiple crane scheduling in batch annealing process with no-delay constraints for machine unloading," in *2012 IEEE International Conference on Information and Automation*. IEEE, 2012, pp. 597–601.

[35] B. Peterson, I. Harjunkoski, S. Hoda, and J. N. Hooker, "Scheduling multiple factory cranes on a common track," *Computers & Operations Research*, vol. 48, pp. 102–112, 2014.

[36] R. Bürgy and H. Gröflin, "The blocking job shop with rail-bound transportation," *Journal of Combinatorial Optimization*, vol. 31, pp. 152–181, 2016.

[37] J.-Q. Li, Y. Du, K.-Z. Gao, P.-Y. Duan, D.-W. Gong, Q.-K. Pan, and P. Suganthan, "A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 2153–2170, 2021.

[38] J. Li, A. Xu, and X. Zang, "Simulation-based solution for a dynamic multi-crane-scheduling problem in a steelmaking shop," *International Journal of Production Research*, vol. 58, no. 22, pp. 6970–6984, 2020.

[39] F. Yuan, K. Feng, S.-j. Lin, and A.-j. Xu, "A study on DAA-based crane scheduling models for steel plant," *International Journal of Production Research*, vol. 59, no. 20, pp. 6241–6251, 2021.

[40] Y. Du, J. Li, C. Li, and P. Duan, "A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[41] Z. Xu, D. Chang, M. Sun, and T. Luo, "Dynamic scheduling of crane by embedding deep reinforcement learning into a digital twin framework," *Information*, vol. 13, no. 6, p. 286, 2022.

[42] Z. Xu, D. Chang, T. Luo, and Y. Gao, "Intelligent scheduling of double-deck traversable cranes based on deep reinforcement learning," *Engineering Optimization*, pp. 1–17, 2022.

[43] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *arXiv preprint arXiv:2105.14491*, 2021.

[44] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[45] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing

clusters," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 270–288.
[46] H. J. Carlo and F. L. Martínez-Acevedo, "Priority rules for twin automated stacking cranes that collaborate," *Computers & Industrial Engineering*, vol. 89, pp. 23–33, 2015.
[47] M. L. Pinedo, *Scheduling*. Springer, 2012, vol. 29.
[48] K.-K. Yang, "A comparison of dispatching rules for executing a resource-constrained project with estimated activity durations," *Omega*, vol. 26, no. 6, pp. 729–738, 1998.
[49] P. Legato, R. Trunfio, and F. Meisel, "Modeling and solving rich quay crane scheduling problems," *Computers & Operations Research*, vol. 39, no. 9, pp. 2063–2078, 2012.