A Topology-Aware Optimization Framework for Conveyor System Scheduling

Guodong Wang
School of Computer Science and Technology, Qingdao University, Qingdao, Shandong, China

Abstract: In modern industrial settings, conveyor systems often involve multiple devices working together, changing tasks, and complex scheduling needs. A common challenge is the lack of effective modeling of how devices are connected and how their working states depend on each other. Traditional scheduling methods usually make decisions based only on local information or the state of a single device, which makes it hard to respond quickly and efficiently at the system level. To address this, we propose a dynamic scheduling method for conveyor systems based on graph neural networks (GNNs). First, industrial cameras capture real-time image sequences to estimate the speed of each conveyor belt and extract visual features. These are used to build small time-based graphs, which are then combined into a full system graph. The GNN learns the relationships and dependencies between devices to create a global state representation. Using this representation, a reinforcement learning-based policy network is trained to automatically adjust the belt speeds. A visual feedback loop is also introduced to enable continuous online learning. Additionally, we design a method to dynamically split and compress the system graph to improve modeling efficiency and scheduling accuracy. Experiments show that this method offers better flexibility, stability, and energy efficiency under complex conveyor conditions.

Keywords: Conveyor System Scheduling; Topology-Aware Optimization; Graph Neural Networks; Reinforcement Learning.

1. Introduction

In typical scenarios like ports, smart warehouses, and mining transport, conveyor systems serve as key infrastructure for logistics, responsible for coordinating multiple devices and continuously moving materials [1–3]. As the system size grows and tasks become more complex, these systems show highly dynamic behavior, strong structural dependencies, and unstable states. This puts higher demands on the Intelligence and adaptability of scheduling strategies [4]. However, most existing scheduling methods rely on fixed rules or static models. They make decisions based on single devices or local parameters and do not model the connections and dependencies between devices as a whole. As a result, it's hard for them to maintain overall system efficiency and fast response when tasks change frequently or devices are tightly connected. Therefore, building a scheduling model that understands the system's structure and can adapt to dynamic changes is a key challenge for improving the intelligence of conveyor systems [5].

Although some recent studies have started to use graph-based modeling and intelligent algorithms to improve conveyor system scheduling, most existing methods still face major limitations [6,7]. On one hand, current graph models are often static or rule-based, making them unable to capture the dynamic changes in system topology and state during operation. This makes it difficult to handle real-world scenarios where tasks change frequently and working conditions vary rapidly. On the other hand, the sensing and modeling layers are not tightly integrated. System status is often determined by a small number of sensors or local signals, ignoring important visual cues such as material distribution or pile-ups on the conveyor surface. This leads to scheduling strategies that are not responsive to the full system state [8]. Moreover, most scheduling strategies still rely on traditional optimization methods, which struggle to coordinate multiple devices effectively in high -dimensional and complex environments [9]. Therefore, there is an urgent need for a unified scheduling framework that combines visual perception, dynamic graph modeling, and policy learning to ensure efficient and stable operation of conveyor systems under complex conditions.

To address the above issues, this paper proposes a dynamic scheduling method for conveyor systems that combines visual sensing, graph neural network (GNN) modeling, and reinforcement learning [10]. First, industrial cameras are used to continuously capture image sequences of conveyor belts. By applying feature matching and speed estimation techniques, the system estimates the operational status of each device and builds time-based subgraphs to extract sequential state information. Then, based on multiple conveyor belts, a system-level graph is constructed to represent the device network. A GNN [11] is used to model the connections and state transitions between devices, creating a global topological state representation. On top of this, a reinforcement learning policy network is designed to adaptively generate speed control actions based on the system state. A feedback loop is also introduced using visual information, allowing the system to continuously learn and improve its scheduling policy over time. To improve modeling efficiency and inference speed, a dynamic graph partitioning and compression mechanism is proposed. This mechanism identifies less critical regions based on activity level and state change density and models them in a more abstract way to reduce unnecessary computation. Experimental results show that this method achieves better scheduling flexibility, system stability, and energy efficiency in various typical conveyor scenarios, demonstrating its effectiveness and practical value.

The main contributions of this paper are as follows:

• We propose a scheduling modeling framework for conveyor systems that integrates visual perception and graph neural networks. By jointly modeling image-based temporal subgraphs and the device-level system graph, we achieve, for the first time, a unified representation of structural topology and state dependencies in conveyor scenarios;

• We design a dynamic topology partitioning and graph compression mechanism based on operating speed and image-derived state density. This approach effectively reduces redundant modeling costs while improving the modeling accuracy and inference efficiency in key regions;

• We develop a graph-embedding-driven reinforcement learning policy network. Through a closed-loop execution framework based on visual feedback, the system enables adaptive optimization of multi-device scheduling strategies;

• We conduct experiments across several typical conveyor task scenarios. Results show that our method outperforms existing approaches in terms of scheduling flexibility, energy efficiency, and system stability, demonstrating strong potential for practical deployment and broader application.

The remainder of this paper is organized as follows: **Sec.2** presents the related work, including mainstream scheduling methods for conveyor systems, graph neural network modeling techniques, and the application of reinforcement learning in industrial control. **Sec.3** provides a detailed description of the proposed method, covering the visual perception module, graph construction process, dynamic topology partitioning mechanism, and the design of the reinforcement learning policy network. **Sec.4** introduces the experimental setup and evaluation metrics, followed by performance validation under various typical conveyor scenarios. **Sect.5** discusses the applicability, potential extensions, and limitations of the proposed method. Finally, **Sec.6** concludes the paper and outlines future research directions.

2. Related Work

2.1 Scheduling Methods for Conveyor Systems

Traditional conveyor system scheduling mainly relies on rule-based control strategies, such as start-stop threshold settings, predefined speed tables, and static task scheduling. These approaches are typically implemented using programmable logic controllers (PLCs), which offer clear control logic and high execution efficiency. However, they lack the ability to perceive changes in system status, making them ineffective under complex or dynamic operating conditions [1] . In recent years, some studies have attempted to reframe the scheduling optimization problem as a modeling and learning task. Techniques such as machine learning, optimization algorithms—including genetic algorithms, ant colony optimization, and reinforcement learning—have been introduced to improve scheduling performance [9,12]. Nevertheless, these methods often rely on simplified system structures or assume static working conditions, making them difficult to apply to real-world conveyor systems involving multiple devices and strong coupling [13].

2.2 Application of Visual Perception in Conveyor Systems

Visual perception technologies are increasingly used in manufacturing and logistics systems, mainly for tasks such as object detection, surface defect inspection, and state estimation [14]. In conveyor scenarios, cameras can serve as noncontact sensors, providing richer status information than traditional sensors—for example, detecting material accumulation, spillage, or abnormal halts on the belt surface [15]. However, most current visual applications in conveyor systems are limited to static image analysis. They lack modeling of temporal continuity and multidevice collaborative states, making it difficult for visual information to directly support system-level scheduling and control. Moreover, there is often a weak coupling between visual data and system structure. As a result, image-derived information is rarely transmitted effectively to the scheduling decision layer, limiting its value in closed-loop optimization [16].

2.3 Graph-Based Modeling and Graph Neural Networks

Graph Neural Networks (GNNs) have gained wide popularity in recent years for modeling structured data and have demonstrated strong capabilities in structural reasoning across various fields such as traffic scheduling, industrial process control, and smart grids [17]. In conveyor systems, the devices are connected through explicit material flow paths and scheduling dependencies, making them naturally suitable for graph-based modeling [18]. Some studies have attempted to model conveyor devices as graph nodes and use techniques such as Graph Convolutional Networks (GCNs) or Graph Attention Networks (GATs) to capture information flow and state coordination between devices [8]. However, most of these approaches still rely on static graph structures and lack the ability to model topological changes that occur with dynamic device states. In addition, there is often no effective integration between graph structures and visual state information, resulting in incomplete representations of the overall system state [19].

2.4 Applications of Reinforcement Learning in Scheduling Optimization

Reinforcement Learning (RL), as an optimization approach for interactive decision-making problems, has been widely applied in areas such as job shop scheduling, robotic control, and energy management [11,20]. In conveyor system scheduling, RL can be used to learn the optimal mapping from system states to control actions (e.g., speed configurations), offering strong adaptability in policy learning [21]. However, traditional RL methods often face performance bottlenecks when dealing with high-dimensional state spaces, coordinated decision-making across multiple devices, and constraints imposed by system topology. In recent years, some studies have explored combining GNNs with RL (e.g., Graph-RL, GNN-RL) to enhance the structural awareness of policy learning. Nevertheless, applications of such methods in industrial conveyor scenarios are still in the early stages, with limited efforts to systematically integrate visual data, dynamic graph structures, and multi-objective scheduling tasks [22,23].

In summary, existing research still shows critical gaps in the following three areas: (1) A lack of deeply integrated methods that combine visual perception with graph-based modeling, preventing the formation of a closed-loop structural representation from sensing to modeling; (2) The absence of scheduling optimization mechanisms that account for dynamic changes in system topology; (3) Limited use of reinforcement learning in conveyor scenarios for training and feedback of multi-device coordination strategies. To address these limitations, this paper proposes a multi-level structural embedding model that integrates graph neural networks with visual perception. A dynamic topology partitioning and compression mechanism is introduced to adapt to structural variations, and a reinforcement learning policy network is employed for speed scheduling control. This approach enhances the system's perception coverage, modeling depth, and decision-making intelligence, overcoming the key shortcomings of existing scheduling methods.

3. Proposed Method

3.1 Vision-Based Non-Contact Speed Estimation and Temporal Subgraph Modeling

As shown in Fig 1-A, to achieve accurate and non-contact sensing of conveyor belt operating states, we propose a vision-guided speed estimation approach that integrates visual perception with graph-based modeling. The method consists of four main components: image acquisition and physical speed estimation, temporal graph construction from image sequences, graph structure generation, and feature extraction using a graph neural network.

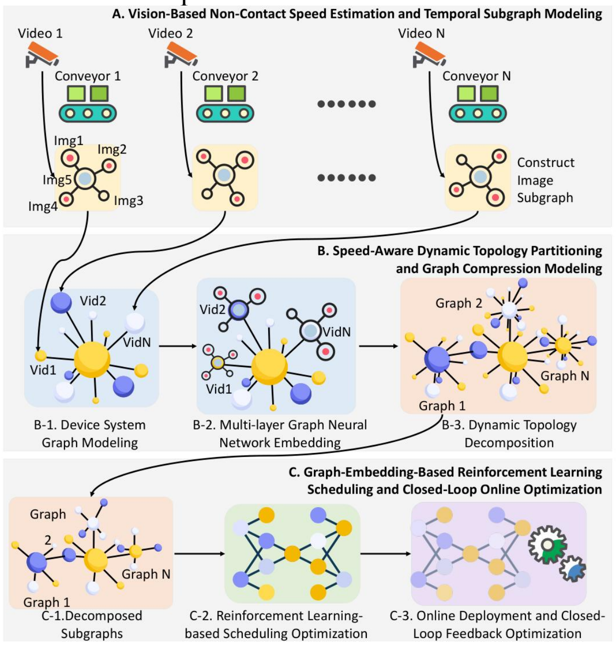
A. Vision-Based Non-Contact Speed Estimation and Temporal Subgraph Modeling
Video 1
Video 2
Video N
Conveyor 1
Conveyor 2
Conveyor N
Img1
Img2
Img5
Img4
Img3
Construct Image Subgraph
B. Speed-Aware Dynamic Topology Partitioning and Graph Compression Modeling
Vid2
VidN
Vid1
Vid2
VidN
Vid1
Graph 2
Graph N
Graph 1
B-1. Device System Graph Modeling
B-2. Multi-layer Graph Neural Network Embedding
B-3. Dynamic Topology Decomposition
C. Graph-Embedding-Based Reinforcement Learning Scheduling and Closed-Loop Online Optimization
Graph 2
Graph N
Graph 1
C-1.Decomposed Subgraphs
C-2. Reinforcement Learning-based Scheduling Optimization
C-3. Online Deployment and Closed-Loop Feedback Optimization

Figure 1. Overall Framework of the Proposed Conveyor Scheduling Method

3.1.1 Visual speed estimation mechanism

An industrial camera is installed above each conveyor belt, positioned vertically to capture the belt's motion area. The camera continuously captures an image sequenceI1, I2,…IT at a fixed frame rate f (frames per second).

To convert pixel distances in the image into real-world physical distances, a calibration board is used to obtain the unit conversion coefficient k, defined as:

$$k = L_{\text{real}} / L_{\text{pixel}} \left(\text{unit:~mm/pixel}\right) \tag{1}$$

Where L_{real} is the actual physical distance measured on the calibration board (in mm), and L_{pixel} is the corresponding pixel length in the image.

As shown in Fig 2, for two consecutive frames I_t and I_{t+1}, the SIFT algorithm is used to extract sets of image feature points F_t and F_{t+1}, respectively. A set of matched feature point pairs $\{\left(p_i^t, p_i^{t+1}\right)\}_{i=1}^{SN}$ is then obtained, where SN is the number of matches. The feature extraction and matching are defined as:

$$F_t = \text{SIFT}\left(I_t\right), F_{t+1} = \text{SIFT}\left(I_{t+1}\right) \tag{2}$$

$$\{\left(p_i^t, p_i^{t+1}\right)\}_{i=1}^{SN} = \text{Match}\left(F_t, F_{t+1}\right) \tag{3}$$

$$\text{Match}\left(F_t, F_{t+1}\right) \triangleq \{(p,q) \mid \frac{\left\|d_1(p)\right\|}{\left\|d_2(p)\right\|} < \tau\}, \quad \tau = 0.75 \tag{4}$$

The pixel displacement for each matched pair is computed as:

$$\Delta p_i = \mid p_i^{t+1} - p_i^t \mid_2 \tag{5}$$

The average pixel displacement is given by:

$$\overline{\Delta p} = \frac{1}{SN} \sum_{i=1}^{SN} \Delta p_i \tag{6}$$

With the conversion coefficient k and the time interval between frames, the average physical displacement $\overline{\Delta x}$ and the estimated instantaneous speed v_t are calculated as:

$$\overline{\Delta x} = k \cdot \overline{\Delta p}, \quad v_t = \frac{\overline{\Delta x}}{\Delta t} = f \cdot k \cdot \overline{\Delta p} \tag{7}$$

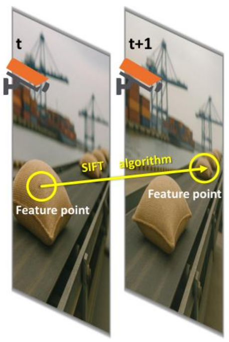

Figure 2. SIFT-Based Visual Speed Estimation Pipeline

To reduce noise in the speed sequence $v_1, v_2, ..., v_T$, a sliding window filter with length w is applied. The smoothed speed is computed as:

$$v'_t = \frac{1}{w}\sum_{i=t-w+1}^{t} v_i \quad (8)$$

The final output v'_t is used as the speed feature input for the belt node in the device system graph.

3.1.2 Temporal subgraph construction and modeling from image sequences

Within a fixed time window τ, an image sequence $\mathrm{I}_1, \mathrm{I}_2, ..., \mathrm{I}_T$ is collected, where each image It is represented as a graph node Nt with node features defined as:

$$N_t = \left[\phi(I_t), t\right] \quad (9)$$

Here, $\phi(It)$ denotes the deep feature vector extracted from image I_t using a convolutional neural network (e.g., ResNet).

A temporal subgraph $G_{img} = (N, E)$ is constructed from the image sequence, where $\mathrm{N} = N_{i=1}^{T}$ is the set of nodes and E is the edge set composed of two types of edges:

Temporal Edges connect consecutive frames:

$$E_T = \{(N_t, N_{t+1}) | 1 \le t < T\} \quad (10)$$

Similarity Edges connect pairs of nodes with cosine similarity above a threshold δ:

$$w_{ij} = \cos(N_i, N_j) = \frac{N_i \ N_j}{|N_i| \cdot |N_j|}, \quad (11)$$

$$E_S = \{(N_i, N_j) w_{ij} > \delta\} \quad (12)$$

The final graph structure is defined as:

$$G_{\mathrm{img}} = (\boldsymbol{N}_{\mathrm{img}}, \ E_T \cup E_S) \quad (13)$$

The graph G_{img} is then fed into a graph neural network (GNN), denoted as f_{GNN} , for embedding learning. Let $\boldsymbol{N}_{img} \in R^{T \times d}$ denote the initial node feature matrix. The graph convolutional update at layer $l+1$ is expressed as:

$$\boldsymbol{N}_{\boldsymbol{img}}^{(\boldsymbol{l}+1)} = f_{GNN}\left(G_{img}, \boldsymbol{N}_{\boldsymbol{img}}\right) = \sigma\left(\mathrm{D}'^{-1/2}_{img} A'_{img} \mathrm{D}'^{-1/2}_{img} N^{(l)}_{img} W^{(l)}_{img}\right) (14)$$

Here, $A'_{img} = \ +$ is the adjacency matrix with self-loops, $\mathrm{D}'^{-1/2}_{img}$ is the corresponding degree matrix, $W^{(l)}_{img}$ is the learnable weight matrix at layer l , and $\sigma(\cdot)$ is a nonlinear activation function.

Finally, a $READOUT$ operation aggregates the node embeddings to produce a subgraph-level representation:

$$\boldsymbol{z}_{\boldsymbol{img}} = READOUT\left(\{\boldsymbol{N}_{\boldsymbol{t}}^{(\boldsymbol{L})}\}_{t=1}^{T}\right) \quad (15)$$

Where $READOUT$ (·) is implemented as mean pooling.

This embedding $\boldsymbol{z}_{\boldsymbol{img}}$ captures both the temporal dependencies and structural patterns of the image sequence and is used as one of the input features for the conveyor belt node in the system -level device graph. It provides dynamic visual information to support downstream modeling and scheduling optimization.

3.2 Speed-Aware Dynamic Topology Partitioning and Graph Compression Modeling

As shown in Fig 1-B, to address the challenges of heterogeneous device states and strong scheduling dependencies in conveyor systems, we propose a speed-aware dynamic topology partitioning method. Combined with graph compression strategies, this approach enables structural abstraction and semantic aggregation for inactive regions, thereby improving the efficiency and reasoning capacity of system-level graph modeling.

3.2.1 Device system graph modeling

As shown in Fig 1-B-2, each conveyor belt in the system is represented as a node Np within the current time window. The node feature is defined as the concatenation of the smoothed speed value and the image subgraph embedding:

$$N_p = \left[v_p, \boldsymbol{z}_{\boldsymbol{p}}^{\boldsymbol{img}}\right] \quad (16)$$

Where, vp denotes the estimated smoothed speed of belt p, and z_{img}^{p} is the embedding from its corresponding temporal image subgraph. The operator [.] represents feature concatenation.

The device system graph is defined as $G_{sys} = (N_{sys}, A)$, where $N_{sys} = N_{p=1}^{BN}$ represents the set of all BN conveyor nodes, and A denotes the set of directed edges representing scheduling depende1ncies. If device p transfers material to device q, a directed edge ($N_p \rightarrow N_q$) is created. The edge weight a_{pq} is dynamically generated based on feature similarity or scheduling relevance. In this paper, the edge weights are computed as:

$$a_{pq} = MLP\left(\left[N_p, N_q\right]\right) \quad (17)$$

Where $[.]$ denotes feature concatenation and $MLP(\cdot)$ is a multi-layer perceptron used to learn the latent relationship strength between nodes.

The entire system graph $Gsys$ is input to a graph neural network (f_{GNN}) for joint modeling.

The initial node feature matrix is denoted as $\boldsymbol{N}_{sys} \in R^{M\times d}$, where M is the number of devices. Each node includes both speed and visual embedding features.

The graph convolutional update process is defined as:

$$\boldsymbol{N}_{\boldsymbol{sys}}^{(\boldsymbol{l}+1)} = f_{GNN}\left(G_{sys}, \mathrm{N}_{\mathrm{sys}}\right) == \sigma\left(\mathrm{D}'_{\mathrm{sys}}{}^{-1/2} A'_{sys} \mathrm{D}'_{\mathrm{sys}}{}^{-1/2} N_{sys}^{(l)} W^{(l)}\right) \quad (18)$$

Where $A'_{sys} = A_{sys} +$ is the adjacency matrix with self-loops, $\mathrm{D}'_{\mathrm{sys}}$ is the corresponding degree matrix, $W^{(l)}$ is the learnable weight at layer l, and $\sigma(\cdot)$ is a non-linear activation function.

A $READOUT$ operation is applied to obtain the global system-level state embedding:

$$z_{sys} = READOUT\left(\{\boldsymbol{N}_{\boldsymbol{i}}^{(\boldsymbol{L})}\}_{i=1}^{M}\right) \quad (19)$$

Where $READOUT$ $(\cdot)$ is implemented using mean pooling. The resulting vector $zsys$ serves as the input for the subsequent scheduling control module.

3.2.2 Dynamic topology partitioning strategy

To enhance the responsiveness of the system graph structure to operational activity, As shown in Fig 1-B-3, we introduce a dualcriterion strategy based on speed awareness and subgraph density perception for dynamic graph partitioning:

Speed-Based Subregion Partitioning Let θv denote the activity threshold. For each node vi, the activity status is defined as:

$$Active\left(v_{\mathrm{i}}\right) = \begin{cases} 1, & \text{if } v_i \geq \theta_v \text{or locate data scheduling boundary} \\ 0, & v_i < \theta_v \text{ and operating under noload} \end{cases} \quad (20)$$

Based on this rule, the active node set V_{act} and inactive node set V_{inact} are determined, resulting in an initial speed-driven subgraph partition.

Image Subgraph Density-Aware Analysis For each conveyor belt's corresponding image subgraph G_i^{img} , we compute the number of nodes ni and the inter-frame variation rate Δvar .

If the following condition is met:

$$n_i > \delta_n \ or \ \Delta_{var} > \delta_s, \quad (21)$$

The complete graph structure is preserved. Otherwise, the subgraph is subject to compressed modeling.

3.2.3 Multi-strategy graph compression modeling

Based on the previously described region partitioning, the device system graph Gsys is divided into two types of subgraphs:

Active region subgraph: $G_{act} = \left(\boldsymbol{N}_{act}, \boldsymbol{E}_{act}\right)$.

Inactive region subgraph: $G_{inact} = \left(\boldsymbol{N}_{inact}, \boldsymbol{E}_{inact}\right)$.

Here, $\boldsymbol{N}_{act}$ and $\boldsymbol{N}_{inact}$ represent the sets of active and inactive nodes, respectively, and $\boldsymbol{E}_{inact}$ and $\boldsymbol{E}_{inact}$ are the corresponding edge sets.

To reduce the modeling complexity in inactive regions, we apply a differentiable graph pooling mechanism—DiffPool [24]—on G_{inact}, enabling joint structural and semantic compression.

Let the node feature matrix of G_{inact} be **N**inact ∈ RN×d and the adjacency matrix be **E**inact ∈ $R^{N\times N}$. DiffPool uses two graph neural networks:

A GNN to generate the node-to-cluster assignment (soft pooling matrix):

$$\boldsymbol{S} = GNN_{pool}\left(\boldsymbol{N}_{\boldsymbol{inact}}, \boldsymbol{E}_{\boldsymbol{inact}}\right) \in R^{N\times K}, \quad (22)$$

A GNN to extract node embeddings:

$$\boldsymbol{Z}_{\boldsymbol{inact}} = GNN_{embed}\left(\boldsymbol{N}_{\boldsymbol{inact}}, \boldsymbol{E}_{\boldsymbol{inact}}\right) \in R^{N\times d'}, \quad (23)$$

The super node features and their new connectivity are updated as:

$$\boldsymbol{N}_{\boldsymbol{inact}}^{'} = \boldsymbol{S} \ \ \boldsymbol{Z}_{\boldsymbol{inact}} \in R^{K\times d'} \quad (24)$$

$$\boldsymbol{E}_{\boldsymbol{inact}}^{'} = \boldsymbol{S} \ \ \boldsymbol{E}_{\boldsymbol{inact}} \boldsymbol{S} \in R^{K\times K}, \quad (25)$$

Where, $\boldsymbol{N}_{\boldsymbol{inact}}^{'}$ represents the compressed super node features, and $\boldsymbol{E}_{\boldsymbol{inact}}^{'}$ represents the corresponding adjacency structure. This mechanism supports end-to-end training and effectively preserves both structural and semantic information.

Finally, the full structure of the active region

G_{inact} and the compressed representation of the inactive region ($\boldsymbol{N}_{inact}^{'}$, $\boldsymbol{E}_{inact}^{'}$) are combined to generate the nested system graph state embedding:

$$G_{inact} = \left(\boldsymbol{N}_{inact}^{'}, \boldsymbol{E}_{inact}^{'}\right) \tag{26}$$

3.2.4 Cross-scale nested graph construction

Based on the compressed graph embedding $Ginact$, the final optimized system graph is constructed as:

$$G_{sys}' = \left(G_{act} \cup G_{inact}, E'\right) \tag{27}$$

Where, E' includes the original connections within the active region and additional cross-layer edges between the virtual super nodes (from G_{inact}) and real nodes (from G_{act}).

This cross-scale nested structure preserves topological completeness while achieving representational compression. It serves as the final input to the reinforcement learning module for scheduling policy optimization:

$$z_{final} = f_{GNN}\left(G_{sys}'\right) \tag{28}$$

3.3 Graph-Embedding-Based Reinforcement Learning Scheduling and Closed-Loop Online Optimization

As shown in Fig 1-C, to optimize scheduling in conveyor systems under complex topologies and dynamic task conditions, we formulate the scheduling control problem as a Reinforcement Learning (RL) task. A closed-loop control framework is designed based on graph-embedded system states, policy generation, and real-time feedback. The system takes the global state embedding extracted via a graph neural network as input, generates speed control actions through a policy network, and continuously improves the scheduling strategy using feedback rewards—thus forming a closed-loop of state perception → policy decision → feedback learning for intelligent scheduling.

3.3.1 Problem formulation: RL-based MDP definition

We model the conveyor scheduling process as a Markov Decision Process (MDP), defined by the 5-tuple $\left(S, A, P, R, \gamma\right)$, where:

State Space S: The system state $s_t \in R^d$ is represented by the global graph embedding z_{final} obtained from the GNN, which integrates device topology, visual features, and historical speed information:

$$s_t = z_{final} \tag{29}$$

Action Space A: The action $at \in Rn$ represents the speed configuration of n conveyor belts:

$$a_t = \left\{v_{1t}, v_{2t}, \cdots, v_{nt}\right\} \tag{30}$$

Transition Probability $P(s_{t+1} \mid s_t, a_t)$: Determined by the dynamics of the physical system, including transport delays, material accumulation, and execution noise. These are difficult to model explicitly and are approximated implicitly through reinforcement learning.

Reward Function $R\left(s_t, a_t\right)$: Encourages the alignment between scheduling actions and recommended speeds, defined as:

$$r_t = -\sum_{i=1}^{n}\left(v_{it} - v_{it}'\right)^2 \tag{31}$$

Where, v_{it}' is the recommended speed for belt i , derived from visual estimation and load expectations. A material accumulation detection module is introduced to assess blockage risks on the belt. The detection function is defined as:

$$Pile_t^i = I\left[\frac{A_{fg}^i}{A_{total}^i} > \delta\right], \quad \delta \in \left(0,1\right), \tag{32}$$

Where A_{fg}^i is the foreground (material) area, and A_{total}^i isthe total area of the image. $I\left[\cdot\right]$ is the indicator function. When $Pile_t^i$ = 1, it indicates a risk of material accumulation.

The recommended speed is adjusted based on the pile-up status:

$$v_{it}' = \begin{cases} \min\left(v_{it}^{his}, v_i^{max}\right), & if \quad Pile_t^i = 0 \\ \lambda \cdot v_{it}^{his} & if \quad Pile_t^i = 1, \lambda \in \left(0,1\right) \end{cases} \tag{33}$$

Where v_{it}' is the recommended speed, v_{it}^{his} is the historical stable speed, v_i^{max} is the maximum allowable speed, and λ is a slowdown factor applied under pile-up risk.

Discount Factor γ : Reflects the decay of future rewards, set as $\gamma = 0.95$.

3.3.2 Policy function modeling and network architecture

The reinforcement learning policy function $\pi_\theta(a_t \mid s_t)$ is implemented using a multi-layer perceptron (MLP) with the following structure:

Input layer: The input is the system state vector $s_t \in R^d$.

Two hidden layers: Each consists of ReLU activation and Batch Normalization modules.
Output layer: Applies the tanh activation function to produce normalized actions $a_t' \in (-1,1)^n$.
Action Mapping and Clipping: The normalized action a_t' is then mapped to the actual velocity range and clipped to satisfy system constraints:

$$a_t = clip\left(\frac{v_{max}-v_{min}}{2}\cdot a_t' + \frac{v_{max}+v_{min}}{2}, v_{min}, v_{max}\right) \quad (34)$$

Here, v_{min} and v_{max} denote the system's minimum and maximum allowable speeds, respectively.
The final output of the policy network is the speed control action:

$$a_t = \pi_\theta(s_t) \quad (35)$$

3.3.3 Policy optimization algorithm: PPO-based training mechanism
We adopt the Proximal Policy Optimization (PPO) algorithm to train the policy network. In each training iteration, the agent interacts with the environment to collect a trajectory (s_t, a_t, r_t, s_{t+1}). The advantage estimate $\mathrm{AE_t}$ is used to construct the PPO loss function:

$$\mathrm{L_{PPO}} = E_t\left[\min\left(\rho_t(\theta)\times \mathrm{AE_t}, clip\left(\rho_t(\theta), 1-Thold, 1+Thold\right)\times \mathrm{AE_t}\right)\right] \quad (36)$$

Where, $\rho_t(\theta) = \frac{\pi_\theta(a_t s_t)}{\pi_{\theta_{old}}(a_t s_t)}$ is the probability ratio between the new and old policies, and $Thold$ is the clipping threshold to ensure stable training.
To estimate the long-term return from each state, we introduce an independent Critic network $V_\phi(s_t)$ to learn the value function. The Critic is implemented as a two-layer MLP, and the loss is defined as:

$$\mathrm{L_{critic}} = E_t\left[\left(V_\phi(s_t) - R_t\right)^2\right] \quad (37)$$

Where $R_t = \sum_{l=0}^{T-t}\gamma^l(r_{t+l})$ is the actual cumulative return.
The advantage function is computed using Generalized Advantage Estimation (GAE):

$$A_t' = \sum_{l=0}^{T-t}\gamma^l\left(r_{t+l} + \gamma V(s_{t+l+1}) - V(s_{t+l})\right) \quad (38)$$

Where $V(s_t) = V_\phi(s_t)$ is the state value predicted by the Critic.
3.3.4 Online deployment and closed-loop feedback optimization
As shown in Fig 1-C-3, the trained policy is deployed in the control module and executed in a loop with a cycle of 1–60 seconds, performing the following closed-loop steps:
State Perception: The visual sensing module continuously captures images, estimates belt speed and load, performs material pile-up detection, constructs the system graph Gt, and extracts the current state st.
Policy Decision: The policy network generates the corresponding speed control action at.
Action Execution: The control command is sent to the equipment controller via the Modbus communication bus.
Reward Feedback: The reward rt is calculated based on the deviation between the executed action and the recommended speed.
Policy Update: The experience tuple (s_t, a_t, r_t, s_{t+1}) is stored in a buffer. PPO is used to periodically update the policy parameters.
Safety Mechanism: If abnormal outputs from the policy are detected, a fallback strategy is triggered (e.g., static speed or manual override).
This scheduling approach effectively integrates graph-based modeling with reinforcement learning, enabling structure-aware, adaptive, and online-optimized control, and is suitable for multi-device collaborative scheduling scenarios.

4. Experiments

4.1 Experimental Setup

4.1.1 Experimental platform and environment configuration
The proposed method was validated on the following experimental platform: the computing setup includes an Intel Xeon 6226R processor, 128GB RAM, and an NVIDIA RTX 3090 GPU. The software environment consists of Python 3.9, PyTorch 2.0, and the DGL framework for model training and inference.
The control system is built on a PLC communicating via the Modbus TCP protocol, with industrial-grade servo drives executing control commands. The visual sensing system uses a Basler acA1920-40gc industrial camera, capturing at 30 FPS, with a calibrated measurement accuracy of ±0.2 mm.
4.1.2 Dataset and evaluation metrics
Dataset: A real-world dataset was collected by

deploying sensors and industrial cameras on a factory production line. It includes 30 hours of conveyor system operation, with scheduling records covering 8 conveyor belts, 104 task transitions, and nearly 50,000 image frames.

Evaluation Metrics: Mean Squared Error (MSE) of Speed: Measures the deviation between the control output and the target speed; Control Latency: The average time from issuing a control action to observing the corresponding acceleration/deceleration on the belt; Makespan: The total time required to complete a batch of tasks under each scheduling method; Action Stability: The standard deviation of speed changes across successive control actions, indicating the smoothness of the policy output.

4.2 Baseline Methods

To comprehensively evaluate the performance of the proposed method, we compare it against four representative baselines: Rule-Based Baseline (RB): A manually designed control strategy using fixed-speed rules based on expert experience. Traditional Reinforcement Learning (DQN): A Deep Q-Network-based approach using the mean pixel intensity of images as the input state representation. GNN-RL without Graph Structure: A method that uses global feature vectors without explicit graph modeling, combining GNN-based representation with PPO training. GNN-Vision-RL (Ours): The full proposed framework that integrates vision-based graph modeling and graph neural reinforcement learning.

4.3 Quantitative Results and Analysis

Table 1 shows the comparison results of different methods across all evaluation metrics on the test set.

Table 1. Performance Comparison on the TestSet

Method	MSE ↓	Latency/ms ↓	Makespan/s ↓	Stability ↓
RB	0.043	124	923.1	0.037
DQN	0.028	96	845.6	0.025
GNN-RL	0.023	89	818.3	0.021
GNN-Vision-RL (Ours)	**0.011**	**62**	**776.5**	**0.014**

The results demonstrate that our proposed method outperforms all baselines across the board, particularly in terms of speed control accuracy (MSE) and task completion time (Makespan). These improvements highlight the effectiveness of visual subgraph modeling and dynamic graph compression in enhancing both the stability and responsiveness of the scheduling policy.

4.4 Visualization and Ablation Study

4.4.1 Policy output visualization

Fig.3 illustrates the variation trends of target conveyor belt speeds over time under different scheduling strategies within a typical task-switching interval. To simulate the system’s response to sudden task changes, the interval from t = 20 to 40 seconds (highlighted in gray) is selected as the disturbance window. We compare each method’s ability to adjust speeds during abrupt state transitions and evaluate system recovery behavior.

As shown in the Fig. 3, the traditional Rule-Based control strategy lacks global awareness of system state and relies solely on fixed thresholds. This results in slow responses and delayed speed adjustments when a task change occurs, making it unsuitable for fine-grained control.

The DQN method demonstrates some ability to adjust dynamically; however, since its state input is limited to local statistical features, it cannot model inter-device structural dependencies. This leads to significant speed oscillations and instability in response to task disturbances.

The GNN-RL method incorporates graph structural information into its modeling, which allows for better representation of device connections compared to DQN. As a result, it achieves faster and smoother speed adjustments. Nonetheless, its reliance on static state inputs without fine-grained visual perception causes noticeable fluctuations during task disturbances, and its recovery speed is slightly inferior to our method.

In contrast, the proposed GNN-Vision-RL approach can quickly detect and respond to state changes by jointly modeling visual subgraph embeddings and the system- level graph structure. This significantly enhances the precision of scheduling policy under complex task transitions. The resulting speed curves are smoother, more continuous, and exhibit faster recovery, clearly outperforming the other methods. These results validate the proposed method’s advantages in responsiveness, robustness, and overall stability under dynamic scheduling scenarios.

4.4.2 Module ablation study

To verify the importance of each component in the proposed scheduling system, we conduct the following ablation experiments:

No Graph Modeling: The device system graph is removed. Global features are computed using average pooling over image features only.

No Compression Mechanism: The system graph is constructed as fully connected, without applying dynamic topology partitioning or graph compression.

No Visual Perception: Visual subgraph embeddings are replaced with values from conventional speed sensors, removing the dynamic perception capability enabled by vision.

As shown in Table 2, removing individual modules results in performance degradation, confirming the necessity of subgraph modeling, topology partitioning, and the nested graph structure.

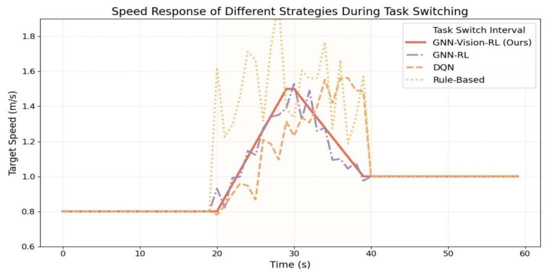

Figure 3. Comparison of Target Speed Curves during Task-Switching Intervals

Table 2. Ablation Study Results

Configuration	MSE	Makespan/s	Stability
GNN-Vision-RL (Ours)	**0.011**	**776.5**	**0.014**
w/o Visual Perception	0.019	812.3	0.020
w/o Graph Modeling	0.022	837.7	0.026
w/o Compression	0.017	791.4	0.019

The results validate the effectiveness of the proposed scheduling optimization framework in improving response speed, control precision, and system stability. The integration of visual perception and graph modeling provides a structured representation of system states, while reinforcement learning—supported by dynamic topology adaptation and online feedback—offers strong adaptability for complex scheduling tasks.

5. Conclusion

This paper addresses the problem of scheduling optimization for conveyor systems under multi-device coordination and dynamic task changes. We propose a novel method that integrates visual perception, graph neural network modeling, and reinforcement learning for dynamic scheduling. Specifically, the system uses non-contact industrial cameras to capture image sequences and construct temporal subgraphs for speed estimation and state modeling. A system-level graph is then built based on scheduling dependencies between devices, and amultilayer graph neural network is used to extract a topology-aware global state embedding. Finally, a reinforcement learning-based policy network is constructed, forming an adaptive closed-loop control framework with online visual feedback and reward evaluation. Experimental results demonstrate that the proposed method outperforms both traditional rule-based strategies and mainstream deep reinforcement learning approaches in terms of scheduling responsiveness, speed control accuracy, and overall system stability. In scenarios involving frequent task switching and state disturbances, our method shows superior robustness and generalization capability. Future work will explore extensions such as multi-task scheduling under complex working conditions, self-recovery mechanisms for device failures, and integration with higher-level MES (Manufacturing Execution Systems), aiming to facilitate practical deployment in industrial conveyor and manufacturing systems.

Acknowledgements

This research was supported by the Central Government Guiding Local Science and Technology Development Project, titled Research and Application of Key Technologies for Full-Process Management Systems and Automation Equipment at Bulk Grain Terminals (Project No. 24-2-4-1-zyyd-hy).

References

[1] K. Wang, Q. Hu, M. Zhou, Z. Zun, and X. Qian, “Multi-aspect applications and development challenges of digital twin-driven management in global smart ports,” Case Studies on Transport Policy, vol. 9, no. 3, pp. 1298–1312, 2021.

[2] C. Zhang, H. Hu, Y. Yang, and N. Wang, “Optimizing the portfolio of smart

technologies to maximize port carbon emission efficiency: a configurational theory approach," Maritime Policy & Management, pp. 1–39, 2025.

[3] L. Tang and Y. Meng, "Data analytics and optimization for smart industry," Frontiers of Engineering Management, vol. 8, no. 2, pp. 157–171, 2021.

[4] W. Mi and Y. Liu, "Development trend and target of smart port," in Smart Ports. Springer, 2022, pp. 189–201.

[5] X. Wang and H. Shi, "Research on intelligent optimization of bulk cargo terminal control system," in Journal of Physics: Conference Series, vol. 1601, no. 5. IOP Publishing, 2020, pp. 052044.

[6] C. F. Fontana, F. Papa, C. L. Marte, L. R. Yoshioka, and C. A. Sakurai, "Intelligent transportation system as a part of seaport terminal management system," International journal of systems applications, engineering & development, vol. 8, pp. 41–46, 2014.

[7] W. Wang, A. Ding, Z. Cao, Y. Peng, H. Liu, and X. Xu, "Deep reinforcement learning for channel traffic scheduling in dry bulk export terminals," IEEE Transactions on Intelligent Transportation Systems, 2024.

[8] X. Song and L. Huang, "Optimization of import business process and system design in bulk port based on internet of things." in ICEIS (4), 2011, pp. 509–512.

[9] S. Liu, Q. Liu, L. Wang, and X. Chen, "Intelligent bulk cargo terminal scheduling based on a novel chaotic-optimal thermodynamic evolutionary algorithm," Complex & Intelligent Systems, vol. 10, no. 6, pp. 7435–7450, 2024.

[10]H. Shi, X. Wang, H. Zhou, and N. Lu, "Research on monitoring and diagnosis model of dry bulk terminal conveying equipment based on edge-cloud collaborative technology," in 6th International Workshop on Advanced Algorithms and Control Engineering (IWAACE 2022), vol. 12350. SPIE, 2022, pp. 359–364.

[11]S. Liu, W. Wang, S. Zhong, Y. Peng, Q. Tian, R. Li, X. Sun, and Y. Yang, "A graph-based approach for integrating massive data in container terminals with application to scheduling problem," International Journal of Production Research, vol. 62, no. 16, pp. 5945–5965, 2024.

[12]W. Yang, X. Bao, Y. Zheng, L. Zhang, Z. Zhang, Z. Zhang, and L. Li, "A digital twin framework for large comprehensive ports and a case study of qingdao port," The International Journal of Advanced Manufacturing Technology, vol. 131, no. 11, pp. 5571–5588, 2024.

[13]B. Greve, N. M. Ahmed, M. Knauer, C. Wanigasekara, and F. S. Torres, "Optimal control of maritime container terminal operations for effective utilization of resources," IEEE Transactions on Intelligent Transportation Systems, 2025.

[14]Y. Li, L. Chen, M. Chen, and X. Qian, "Integrating spatial cognition and slam for improved performance of autonomous material handling robots in dynamic stockyard environments," Industrial Robot: the international journal of robotics research and application, 2025.

[15]J. Xin, R. R. Negenborn, and G. Lodewijks, "Energy-aware control for automated container terminals using integrated flow shop scheduling and optimal control," Transportation Research Part C: Emerging Technologies, vol. 44, pp. 214–230, 2014.

[16]F. Zeng, Q. Wu, and C. Dai, "Speed control simulation system of bulk terminal conveyor belts," in ICTIS 2013: Improving Multimodal Transportation Systems-Information, Safety, and Integration, 2013, pp. 2062–2069.

[17]Y. Lv, Y. Gao, and J. Liu, "Dual strategies-based resilience enhancement in a bulk cargo port under dynamic machinery failure scenarios with reinforcement learning," Ocean & Coastal Management, vol. 260, pp. 107484, 2025.

[18]H. Shi and X. Wang, "Study on optimization of monitoring system of belt conveyor in dry bulk wharf," in 2nd International Conference on Laser, Optics and Optoelectronic Technology (LOPET 2022), vol. 12343. SPIE, 2022, pp. 597–601.

[19]M. Tang, D. Gong, S. Liu, and H. Zhang, "Applying multi-phase particle swarm optimization to solve bulk cargo port scheduling problem," Advances in Production Engineering & Management, vol. 11, no. 4, p. 299, 2016.

[20]A. Ridwan, "Container ship planning optimization using intelligent meta-heuristic algorithms," Industrial Engineering & Management Systems, vol. 21, no. 3, pp. 460– 468, 2022.

[21]L. Henesey, “Multi-agent systems for container terminal management,” Ph.D. dissertation, Blekinge Institute of Technology, 2006.

[22]R.-A. Daineanu, N.-M. Zus, M. Panaitescu, F.-V. Panaitescu, and D. Dinu, “The dynamics of a maritime container terminal complex system: optimization process design,” in Advanced Topics in Optoelectronics, Microelectronics, and Nanotechnologies XI, vol. 12493. SPIE, 2023, pp. 51–60.

[23]M. Jusis, “Method of data synchronization of autonomous port handling processes,” Ph.D. dissertation, Vilniaus universitetas, 2021.

[24]H. Yin, Z. Wang, and N. K. Jha, “A hierarchical inference model for internet-of-things,” IEEE Transactions on Multi-Scale Computing Systems, vol. 4, no. 3, pp. 260–271, 2018.