TOWARDS SOLVING INDUSTRIAL SEQUENTIAL DECISION-MAKING TASKS UNDER NEAR-PREDICTABLE DYNAMICS VIA REINFORCEMENT LEARNING: AN IMPLICIT CORRECTIVE VALUE ESTIMATION APPROACH

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning to plan and schedule is receiving increasing attention for industrial decision-making tasks because of its potential to outperform heuristics, especially under dynamic uncertainty, as well as its efficiency in problem-solving, especially with the adoption of neural networks and the behind GPU computing. Naturally, reinforcement learning (RL) with the Markov decision process (MDP) becomes a popular paradigm. Instead of handling the near-stationary environments like Atari games or the opposite case for open world dynamics with high uncertainty, in this paper, we aim to devise a tailored RL-based approach for the setting in the between: the near-predictable dynamics which often hold in many industrial applications, e.g., elevator scheduling and bin packing, as empirical case studies tested in this paper. We formulate a two-stage MDP by decoupling the data dynamics from the industrial environment. Specifically, we design a bi-critic framework for estimating the state value in stages according to the two-stage MDP. Experimental results show that our engine can adaptively handle different tasks of dynamics data and outperform recent learning-based models and traditional heuristic algorithms.

1 INTRODUCTION

The advent of Industry 4.0 has put forward demanding requirements for resolving the sequential decision-making task in production. The task that involves planning and scheduling has been researched for decades for its potential commercial value in improving the performance of decision algorithms. The planning task, like the bin packing problem (BPP) (Zhao et al., 2020; Zhu et al., 2021; Duan et al., 2022; Zhao et al., 2022; Zhao & Xu, 2022), involves a series of discrete objects under certain constraints to optimize an objective function. And the scheduling task focuses focuses on allocating limited resources to multiple objects to optimize performance indicators under certain constraints, such as the elevator group scheduling problem (EGSP) (Crites & Barto, 1998; zheng et al., 2013; Wei et al., 2020), the vehicle routing problem (VRP) (Nazari et al., 2018), and the job scheduling problem (JSP) (Chen & Tian, 2019).

With the increase of the problem scale, finding the optimal solution in terms of effectiveness and generalization is more challenging. Most traditional solutions in the industry are rule-based, tuning a score function with expert experience for specific tasks but hardly generalizing to similar ones. Others try to formulate the tasks as NP-hard combinatorial optimization problems and then apply heuristics algorithms— such as search algorithm (ELA, 2019; TUR, 2020) and the greedy algorithm (Ramalingam et al., 2017)—but lack real-time response and scalability. Many learning-based methods (Wei et al., 2020; Zhao et al., 2020) are developed as RL, showing its remarkable advantages in sequential decision-making problems. However, existing works still fall behind the industry standards because they are based on low-quality simulators, which intentionally relax the security constraints and weaken the data variability to boost the performance of RL algorithms.

There are two challenges when solving industrial sequential decision-making tasks with RL: unstable environmental dynamics and strict security constraints. Existing works (Hadoux et al., 2014;

Chandak et al., 2020; Chen et al., 2021) formulate these tasks as non-stationary environments and develop classical RL algorithms to learn structural features of the unstable environment, including Meta-RL (Xu et al., 2020; Chen et al., 2021), distribution-based RL (Bellemare et al., 2017) and context detection RL (Padakandla et al., 2020). However, the unstable dynamics present regularities in real-world industrial cases and are potentially predictable (i.e., near-predictable). These works neglect the near-predictable environmental dynamics and thus can not fully exploit the benefits of RL in industrial scenarios. Although Chen et al. (2021) and Wei & Luo (2021) further consider the Constrained Markov Decision Processes (CMDP) (Altman, 1999) and robust constrained Markov decision process (RCMDP) (Russel et al., 2020) for safe RL (Hewing et al., 2020), they tolerate constraints violations and are not up to industry standard. In addition, integrating RL algorithms for non-stationary environments and constraints consideration leads to high state transition uncertainty (Mao et al., 2018) in the training process and thus poor convergence and insufficient performances on our enhanced simulators.

In this paper, we propose a **D**ynamic-**A**ware and **C**onstraints-**C**onfined (**DACC**) RL framework for industrial sequential decision-making tasks. Unlike previous RL-based efforts for industrial cases that formulate the problem as non-stationary CMDP, we first identify the non-stationary but near-predictable environmental dynamics in industrial sequential decision-making tasks and reformulate these tasks as a two-stage MDP. Furthermore, we identify the potential advantage of the two-stage MDP in reducing state transition uncertainty over existing RL methods that use one-stage MDP. Then, we design **DACC**, a bi-critic framework for perceiving the input dynamics and making decision with the guidance of task-related rules under security constraints separately. By estimating the state value in two stages with our bi-critic framework, we reduce the state transition uncertainty caused by the mutually adverse effects of data variability and security constraints. To evaluate our framework's effectiveness and generalization, we conduct experiments on a typical industrial sequential decision-making task and a more restrictive one on our self-developed simulators that consider realistic constraints and real-world data patterns. Results show that **DACC** can adaptively handle different tasks of different data-flow characteristics and outperforms existing rule-based and learning-based algorithms. **The main contributions are as follows:**

1) We identify the industrial sequential decision-making task as a non-stationary but near-predictable environment with strict security constraints and point out the high state transition uncertainty problem in existing RL methods for the task.

2) We innovatively separate the state transition process of these industrial tasks into two stages by decoupling the input data dynamics from the industrial environment. Based on the two-stage MDP, we design a bi-critic framework to capture the regularity of dynamic input data and make decisions with the guidance of task-related rules under security constraints.

3) We apply our framework to two representative yet challenging real-world cases: bin packing and elevator group scheduling problems. Results show that our methods outperform conventional rule-based and state-of-the-art learning-based models. Further comparisons with the Meta-RL methods verify our framework's superiority in grasping the data dynamics of these tasks.

2 Related Work

Rule-based Algorithms for Sequential Decision-Making Tasks. Traditional solutions utilize rule-based algorithms with expert experience for specific tasks. For BPP, the Bottom-Left heuristic (BL) (Chazelle, 1983) searches for Empty Maximum Space (EMS) with the minimum bottom (x) and left (y) coordinates to place the current item; Deepest-Bottom-Left with Fill heuristic (DBLF) (Kang et al., 2012) sorts EMS in order of deepest coordinates (z) and uses bottom and left as tie-breakers; Best Match First Packing heuristic (BMF) (Li & Zhang, 2015) sorts EMS in order of distance to the container's deepest-bottom-left point; and Online BPH (Ha et al., 2017) sorts EMS in order of minimum bottom, deepest, and minimum left. For EGSP, collective control (Strakosch, 1983) assigns the elevator to stop at its nearest call in the running direction; and (Rong et al., 2003) allocates hall calls to the elevator that minimizes the estimated arrival time. Others create policies with matrices of "heuristic scores" and option decisions with the highest score. (Ramalingam et al., 2017) develops a greedy algorithm based on submodularity to maximize its objective function.

Learning-based Algorithms for Sequential Decision-Making Tasks. Reinforcement learning (RL) excels at dealing with sequential decision-making problems. In real-world BPP applications,

the agent usually can not foresee the upcoming boxes, which puts online decision requirements on BPP solvers. This variant of BPP is called online BPP, and Zhao et al. (2020) uses RL and well-designed discrete representations to solve it. While Zhao & Xu (2022) treat the BPP problem as a decision in a continuous space and formulate it into a tree structure. The model combines RL and traditional algorithms for flexible application in various settings. As for EGSP, an early work (Crites & Barto, 1998) uses Q-learning to control elevators to move up and down. A recent work (Wei et al., 2020) explores deep asynchronous actor-critic learning to decide each elevator's next target stop floor. Our model focuses on input-dependent industrial scenarios, where input data are collected from relatively stable processes on the production line with regularity to follow.

RL for Non-stationary Environment. RL algorithms are recently devised for the non-stationary environment, such as distribution-based RL, Meta-RL (learn to learn), and context detection. Distribution-based RL modifies distributions to stabilize the training process and improve performance. Bellemare et al. (2017) use a distribution to represent the value function. Approximating the full distribution mitigates the effects of learning from a non-stationary policy. Meta-RL (Finn et al., 2017; Nagabandi et al., 2018; Xu et al., 2020) learns a large number of tasks and acquires enough prior knowledge to learn new tasks faster. The well-known Meta-RL models are model-based methods suitable for mainly state-dependent simulation scenarios like maze games rather than input-dependent scenarios to some extent. The context detection RL (da Silva et al., 2006; Padakandla et al., 2020; de Oliveira et al., 2006) assesses whether the MDP functions have changed based on sequential observations, creating, updating, and selecting one among several partial models of the environment. Whether to generate a new context depends on expert experience, and the model consumes much memory. CASRL (Chen et al., 2021) considers both meta and context-detection.

3 Our Approach

We propose the **D**ynamic-**A**ware and **C**onstraint-**C**onfined (**DACC**) RL framework for industrial sequential decision-making tasks under near-predictable dynamics and security constraints. We first reformulate the industrial sequential decision-making task as a two-stage MDP and point out the problems with existing RL approaches for these tasks in §3.1. Then, we show the design of our bi-critic framework based on our reformulation, which captures the near-predictable dynamics and confines constraints separately in §3.2.

3.1 Reformulation as a two-stage MDP

A conventional MDP is defined as a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}$ denotes the state transition probabilities, $\mathcal{R}$ is the reward function and γ is the reward discount factor. Generally, $\mathcal{P}(s' \mid s, a)$ takes into the current state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$, and outputs the distribution of the next state $s' \in \mathcal{S}$.

As for near-predictable industrial sequential decision-making tasks, we summarize the two significant characteristics as follows: 1) The generation of new coming decision objects d_{t+1} at time t is only determined by the current data distribution $f_t(d)$(e.g., passenger flow in EGSP), which is task-related and usually presents specific patterns over time in the industry (i.e., near-predictable). For instance, boxes of similar shape and size usually appear in a batch in BPP. And the elevator passenger flow often follows three traffic patterns: up-peak, down-peak, and the two-way traffic pattern (Jansson & Uggla Lingvall, 2015). 2) There are some hard constraints to satisfy and rules to follow in the environment, only related to the current state and the new object to be decided. Therefore, we can divide the state transition into two stages: the state-dependent stage and the input-dependent stage. Here we use the bin packing problem as the context to describe our approach, especially its detailed two-stage state transition process in Fig.1. In the state-dependent stage, the pending current d_t is picked to the position indicated by the action a_t according to the current state $\mathbf{S}_t$, which consists of the current placement state $\mathbf{S}_t^p$, the current box d_t, and a hidden feature $\mathbf{F}_t$. $\mathbf{F}_t$ represents the marginal distribution of future boxes related to $f_t(d), f_{t+1}(d), \cdots, f_{t+\infty}(d)$. Then $\mathbf{S}_t$ transits to $\tilde{\mathbf{S}}_t$, which consists of the next placement state $\mathbf{S}_{t+1}^p$ and $\mathbf{F}_t$. This stage is deterministic such that:

$$\mathcal{P}\left(\tilde{\mathbf{S}}_t \mid \mathbf{S}_t, a_t\right) = \mathcal{P}\left(\mathbf{S}_{t+1}^p, \mathbf{F}_t \mid \mathbf{S}_t^p, d_t, \mathbf{F}_t, a_t\right) = \begin{cases} 1, & \text{if } a_t \text{ corresponds to } \mathbf{S}_{t+1}^p \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

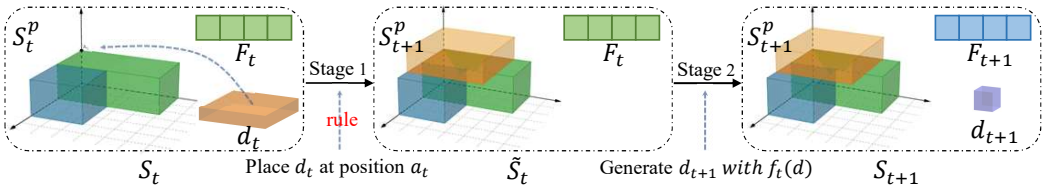

Figure 1: The two-stage state transition process of the BPP. In the state-dependent stage (stage 1), the pending current d_t is picked to the position indicated by the action a_t under hard constraints and rules' guidance. And thus $\mathbf{S}_t$ transits to $\tilde{\mathbf{S}}_t$. In the input-dependent stage (stage 2), a new coming box d_{t+1} is generated with the current box generating distribution $f_t(d)$ and thus $\tilde{\mathbf{S}}_t$ transits to $\mathbf{S}_{t+1}$.

Using $V_1^\pi(\mathbf{S}_t)$ and $V_2^\pi(\tilde{\mathbf{S}}_t)$ to denote the value estimation of $\mathbf{S}_t$ and that of $\tilde{\mathbf{S}}_t$, respectively, we get

$$\begin{aligned} V_1^\pi(\mathbf{S}_t) &= \mathbb{E}_\pi[r + \gamma V_2^\pi(\tilde{\mathbf{S}}_t)] \\ V_1^\pi(\mathbf{S}_t^p, d_t, \mathbf{F}_t) &= \mathbb{E}_\pi[r + \gamma V_2^\pi(\mathbf{S}_{t+1}^p, \mathbf{F}_t)] \end{aligned} \tag{2}$$

As both V_1^π and V_2^π input the identical $\mathbf{F}_t$, this stage focuses on handling the value estimation depending on security constraints and rules. In the input-dependent stage, a new coming box d_{t+1} is generated with the current box generating distribution $f_t(d)$. As $f_t(d)$ is only determined by the data pattern, independent of $\mathbf{S}_t^p$ and $\mathbf{F}_t$, and related to $f_t(d), f_{t+1}(d), \cdots, f_{t+\infty}(d)$. we can get

$$\mathcal{P}\left(\mathbf{S}_{t+1} \mid \tilde{\mathbf{S}}_t\right) = \mathcal{P}\left(d_{t+1}, \mathbf{F}_{t+1} \mid F_t\right) = \mathcal{P}\left(d_{t+1}, \mathbf{F}_{t+1} \mid f_t(d), \mathbf{F}_{t+1}\right) = f_t(d) \tag{3}$$

$$V_2^\pi(\tilde{\mathbf{S}}_t) = V_2^\pi(\mathbf{S}_{t+1}^p, \mathbf{F}_t) = \mathbb{E}_{d_{t+1} \sim f_t(d)}[V_1^\pi(\mathbf{S}_{t+1}^p, d_{t+1}, \mathbf{F}_{t+1})] \tag{4}$$

As the input $\mathbf{S}_t^p$ of V_1^π and V_2^π is the same, while the $\mathbf{F}_t$ of V_2^π is different from the $\mathbf{F}_{t+1}$ of V_1^π, the input-dependent stage focus on handling the value estimation depending on the task-related and time-varying data distribution $f_t(d)$. Hence, by dividing the MDP into two stages and estimating the value separately, we can avoid introducing the uncertainty caused by their interaction, distinguish the uncertain factors from different sources, and accurately make state transitions. However, existing RL algorithms treat two stages as one and approximate the current state value $V_\pi(\mathbf{S}_t)$ with $r + \gamma V(\mathbf{S}_{t+1})$ suffer from high value estimation bias because they neglect the distribution distance between $\mathbf{F}_t$ and $\mathbf{F}_{t+1}$ and mix the unstable factors of the two stages, making the neural network difficult to converge.

REMARK #1: Our derivation identifies that a two-stage MDP can resolve the two main challenges of industrial sequential decision-making tasks, thus alleviating value estimation bias over existing RL approaches that use one-stage MDP on these tasks.

3.2 BI-CRITIC FRAMEWORK FOR RL UNDER NEAR-PREDICTABLE ENVIRONMENTS

Based on the reformulation and derivation, we design a dynamic-aware and constraints-confined (DACC) reinforcement learning framework for industrial sequential decision-making tasks under near-predictable dynamics. Dealing with the two challenges in two stages separately, we overcome the problem of value estimation bias of unstable state transition in dynamic uncertainty.

Overall Framework. The overall framework is shown in Fig.6. For the sake of illustration, we plot the input-dependent stage ($\tilde{\mathbf{S}}_{t-1} \rightarrow \mathbf{S}_t$) of the previous state transition and the state-dependent stage ($\mathbf{S}_t \rightarrow \tilde{\mathbf{S}}_t$) of the current state transition. For the state-dependent stage, we first use the dynamics aware module to extract the latent features of the marginal distribution of future data $\mathbf{F}_t$ with historical data $d_{t-T}, d_{t-T+1}, \cdots, d_t$ and form the current state $\mathbf{S}_t$ with $\mathbf{F}_t$, $\mathbf{S}_t^p$ and d_t. Then we use the primary encoder to encode the current state and output a latent vector (denoted as $\mathbf{z}_t$) representing a multidimensional Gaussian distribution. After that, we estimate the current state value $V_1(\mathbf{S}_t)$ with the primary critic network and use the actor-network to calculate the action vector $\pi(a_t|\mathbf{z_t})$ with the guidance of the constraints and rules module, which considers the hard constraints of industrial tasks and combines task-related expert rules. For the input-dependent stage, we use the dynamics aware module to get $\mathbf{F}_{t-1}$ and form the intermediate state $\tilde{\mathbf{S}}_{t-1}$ with $\mathbf{F}_{t-1}$ and $\mathbf{S}_t^p$. Then we use the secondary encoder to extract its features $\tilde{\mathbf{z}}_{t-1}$ (similar to $\mathbf{z}_t$) and use the secondary

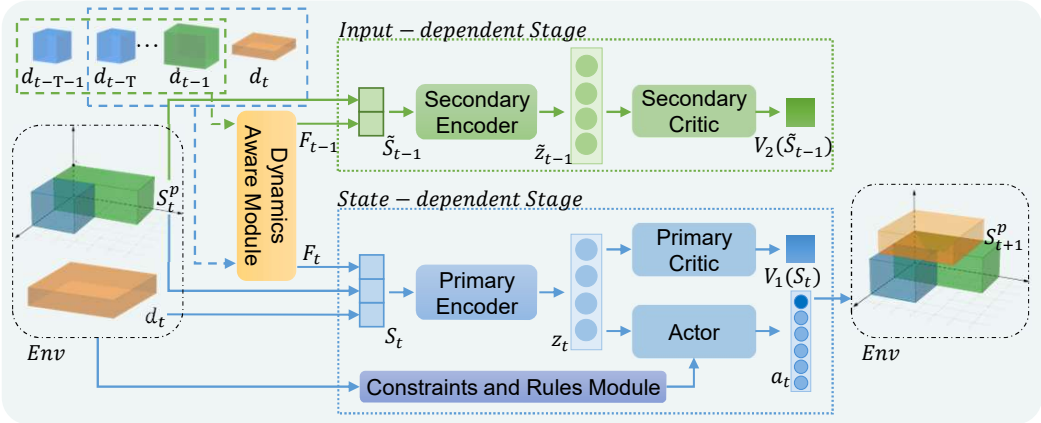

Figure 2: Our bi-critic framework. We illustrate the input-dependent stage of previous time step and the stage-dependent stage of current time step. In the input-dependent stage, the secondary critic focus on handling the value estimation depending on the task-related and time-varying data. In the state-dependent stage, the primary critic focuses on handling the value estimation depending on security constraints and rules. And the actor makes decisions based on the learned features.

critic network to estimate the intermediate state value $V_2(\tilde{\mathbf{S}}_{t-1})$. We can readily resort to off-the-shelf actor-critic-based (Sutton et al., 1999) algorithms (e.g., A2C as used in our experiments for its simplicity and effectiveness, or others like A3C (Mnih et al., 2016), PPO (Schulman et al., 2017)) to train our model. From Eq.2, we calculate the advantage function of A2C as

$$A_t = \alpha r_t + \gamma V_2^{\pi}(\tilde{\mathbf{S}}_t) - V_1^{\pi}(\mathbf{S_t}) \tag{5}$$

From the relation between $V_2^{\pi}(\tilde{\mathbf{S}}_{t-1})$ and $V_1^{\pi}(\mathbf{S}_t)$ shown in Eq.4, we introduce an additional loss function $\mathcal{L}_{KL}$ in our framework to minimize the Kullback-Leibler Divergence between $\tilde{\mathbf{z}}_{t-1}$ and $\mathbf{z}_t$:

$$\mathcal{L}_{KL} = \sum_{i=1}^{k} \left(\log \frac{\tilde{\sigma}_{t-1,(i)}}{\sigma_{t,(i)}} + \frac{\sigma_{t,(i)}^2 + \left(\mu_{t,(i)} - \tilde{\mu}_{t-1,(i)}\right)^2}{2\tilde{\sigma}_{t-1,(i)}^2} - \frac{1}{2} \right) \tag{6}$$

where $\mu_{t,(i)}$ and $\sigma_{t,(i)}^2$ represent the ith component of the mean vector and variance vector of $\mathbf{z}_t$, respectively, while $\tilde{\mu}_{t-1,(i)}$ and $\tilde{\sigma}_{t-1,(i)}^2$ represent those of $\tilde{\mathbf{z}}_{t-1}$, respectively, and k is the dimension of $\mathbf{z}_t$ and $\tilde{\mathbf{z}}_{t-1}$. Thus the overall loss function is defined as

$$\mathcal{L} = {A_t}^2 - A_t \log \pi(a_t|\mathbf{z}_t) + \pi(a_t|\mathbf{z}_t) \log \pi(a_t|\mathbf{z}_t) + \mathcal{L}_{KL} \tag{7}$$

Dynamics Aware Module. To capture the trend of data and learn the features of the marginal distribution of future data, we propose a dynamics aware module, which takes into the historical data $d_{t-T}, d_{t-T+1}, \cdots, d_t$ and outputs the feature vector $\mathbf{F}_t$. As the distribution of data $f_t(d)$ is independent of the current state, we can pre-train the dynamics aware module beforehand. A feasible alternative is to build the dynamics aware module with long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) or Attention (Vaswani et al., 2017) networks and set a pre-training task using the last T data to predict the next three data. For instance, we input the data sequence of $d_{t-T}, d_{t-T+1}, \cdots, d_{t-1}$ to the dynamics aware module and make it predict the d_t, d_{t+1}, d_{t+2}. After that, we use the pre-trained dynamics aware module to output the hidden feature vector $\mathbf{F}_t$.

Constraints and Rules Module. To better satisfy strict constraints and incorporate task-related expert rules in industrial applications, we design a constraints and rules module. It inputs the current state and uses expert rules to calculate an action mask $\mathbf{M}_r$, indicating the expert rules' evaluation of each action in the current state. Then we apply the action mask to the policy function: $\pi' \leftarrow \pi(a_t|\mathbf{z}_t) \circ \mathbf{M}_r$. By explicitly changing the action probability with task-related rules, we avoid excessive exploration of inefficient or illegal actions (e.g., actions that violate constraints) without introducing excessive training complexity. In practical usage, we can simply set the chosen probability of illegal actions as zero to enforce the security constraints.

REMARK #2: Data variability and security constraints introduce instability to the training process. **DACC** eliminates their mutual adverse effects by separating the state transition process into two stages with bi-critic networks.

4 EXPERIMENTS WITH TWO CASE STUDIES

We show the implementations and experiments on two real-world NP-hard industrial scenarios: the bin packing problem(BPP) in §4.1, and the elevator group scheduling problem (EGSP) in §4.2.

4.1 3D BIN PACKING PROBLEM

Problem Settings. The 3D bin packing problem is the essence of mixed palletizing in logistics and warehousing. For a given set of items and containers of a fixed size, the goal is to find the assembly method with the highest space utilization and the least number of containers required. Here we consider the online scenario. Given a sequence of rectangular boxes I with size $x_i \times y_i \times z_i$ (for each $i \in I$) and a standardized pallet with size $L \times W$ and maximum height limit H. The palletization agent e.g., a robot only sees the upcoming box and selects a position to place the box.

Environment Dynamics. Elhedhli et al. (2019) analyzes actual industrial data and finds that the characteristics of depth/width ratio, height/width ratio, and repetition follow specific distributions. The average box volume of different industries and categories is different.

Realistic Constraints. The placement process is subject to the following constraints as widely used in literature (López-Camacho et al., 2013; Zhao et al., 2022): i) no space overlap between boxes on the pallet; ii) while sliding a new box to the assigned position, no collision with the boxes already placed; iii) all placed boxes are physical of stability.

4.1.1 IMPLEMENTATION COMPONENTS

MDP. Specifically, the *state* consists of three parts: height map of the pallet, size of box, and an action mask depending on security constraints and rules, $s_t = (\mathbf{H}_t, i_t, \mathbf{M}_t)$. An Action mask $\mathbf{M}_t$ filters out feasible positions and guides exploration. The *action* is the position selection for box placing. In line with the common practice in literature, the *reward* r_t for each successful placing step is set proportional to $\frac{x_t \times y_t \times z_t}{L \times W \times H}$, and $r_t = 0$ if placement fails.

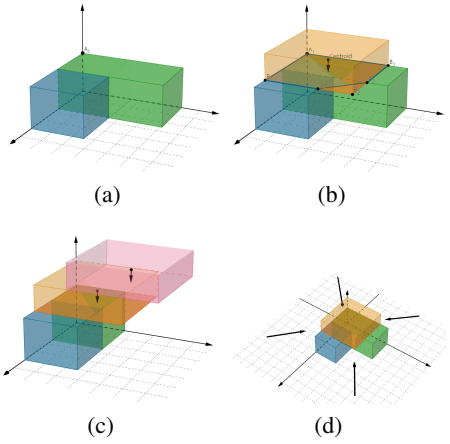

Figure 3: Constraints satisfaction for the BPP task. (a) and (b) shows the convex hull calculation, (c) shows a bad case resulting from considering only the convex hull constraint, and (d) shows four sliding directions we defined.

Network Architecture. We use attention CNN (CBAM) (Woo et al., 2018) as state encoder, long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) as dynamics aware module, and fully-connected networks (FCN) as Critic net and Actor net. Considering that the representation of the pallet is fine-grained and the state dimension is relatively large, we leverage the attention mechanism to increase focus on key regions. The input of the pallet state is a height map $\mathbf{H}_n \in \mathbb{R}^{L \times W}$. And an action mask $\mathbf{M}_n$ represents security constraints and rules for each grid. The size of $\mathbf{H}_n$ and $\mathbf{M}_n$ is doubled while allowing the box to rotate (two rotations: xyz and yxz). Since the action mask calculation takes extended time, we decouple the action space into multiple sub-regions to reduce mask calculation. Details of the hierarchical architecture are given in Appendix A.

Constraints and Rules Module. We combine an invalid action mask to guarantee two constraints (stability and collision avoidance) with a rule guide mask to guide box placement. We use the convex hull and maintain a solid mask to guarantee stability. As Fig.3(a) and 3(b) show, when placing the

yellow box, the centroid of the yellow box must fall within the convex hull of the bottom surface in contact with the lower box. To avoid bad cases like Fig.3(c) (Satisfy that the centroid falls within the convex hull of the contact base but is unstable), we also maintain a solid global mask. Only the intersection of the convex hull and the supporting surface is set to solid, and the box centroid should fall within the solid mask. The placement process is to push the box laterally to a certain position so that even if a collision occurs, it will only cause a slight movement of adjacent boxes. The four selectable sliding directions are shown in Fig.3(d), including left-back to right-front, left-front to right-back, right-front to left-back, right-back to left-front. As for rules, we emphasize the selection of corner points and increase the weight of the corners, which can significantly improve space utilization and reduce the number of solutions (Crainic et al., 2008; Martello et al., 2000; CRA, 2009). To further reduce the computation time of the action mask, we only compute whether the corners on the pallet satisfy the constraints and set other.

4.1.2 Experimental Results

Datasets. We test our model on two datasets generated differently and containing multiple subsets with different distributions. The test results have been verified by Pybullet, a physics engine that supports 3D collision detection. **Mixed-item Dataset (MI Dataset)** is a collection of mixed items with a great variety and few identical items. The data generation scheme follows the realistic 3D-BPP instance generator proposed in (Elhedhli et al., 2019). The training set has 10 million items, with 36,294 species and occurrences varies from 1 to 7,037. The testing set has 10 thousand items, with 4,886 species and occurrences varies from 1 to 15. We set pallet dimensions to the size often used in practice: $L = 120, W = 100$, and $H = 100$ and allow two packing rotation directions **Large-item Dataset (LI Dataset)** is a collection of large items and comes from (Martello et al., 2000), including five classes from (Martello & Vigo, 1998) and three class from (Berkey & Wang, 1987). The pallet dimensions are set to: $L = 100$, $W = 100$, and $H = 100$ and allow two packing rotation directions. The training set has 39,500 items, with 24,889 species in total, and occurrences varies from 1 to 12. The testing set has 500 items, with 401 species in total, and occurrences varies from 1 to 4. We set $\gamma = 1$ in Eq.5 and the learning rate as 1×10^{-5}.

Compared Baselines. We compare our approach with four heuristic methods and two learning-based methods. For heuristic methods, we compare with four classic heuristics, which select free spaces represented by Empty Maximum Space (EMS): Bottom-Left heuristic (BL) (Chazelle, 1983), Deepest-Bottom-Left with Fill heuristic (DBLF) (Kang et al., 2012), Best Match First Packing heuristic (BMF) (Li & Zhang, 2015), Online BPH (Ha et al., 2017). As for learning methods, we compare with an outstanding RL BPP model PCT (Zhao & Xu, 2022) in continuous setting and a model-based Meta-RL model CASRL (Chen et al., 2021). Besides, model A2C and **DACC** (w/o rules) are for ablation study; they represent **DACC** without bi-critic architecture and **DACC** without rule guidance.

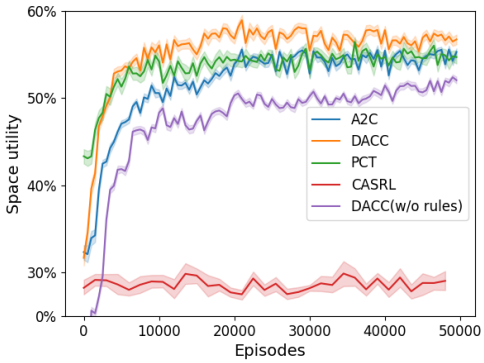

Figure 4: Training rewards of learning-based models on MI Dataset.

Result. Tab.1 shows online bin packing results on two representative datasets. Our model performs well on both mixed-item and large-item datasets, with the flexibility to adapt to different pallet sizes and high-precision placement requirements. Our model achieves higher space utility rates than traditional rule-based algorithms, RL-based models, and the meta-RL model. For learning-based models, the reward curves over training episodes are shown in Fig.4, whereby the confidence interval is 85. As can be seen from the Fig.4, our model converges faster than other learning models.

Ablation study. Compared with the original Actor-Critic (without bi-critic structure) and **DACC** without rule guidance, **DACC** model can converge to stable performance earlier and achieves higher reward, which means the bi-critic structure and the guidance of the rules are beneficial to improve the model's convergence speed and performance.

Table 1: Performance comparison between the baselines and **DACC** in MI Dataset and LI Dataset. The best results are highlighted in **bolded**.

Method	MI Dataset		LI Dataset	
	Space utility ↑	Avg. items ↑	Space utility ↑	Avg. items ↑
BL (Chazelle, 1983)	22.55%	9.32	27.97%	2.04
DBLF (Kang et al., 2012)	26.54%	10.51	27.18%	1.98
BMF (Li & Zhang, 2015)	29.67%	11.47	27.28%	1.99
Online BPH (Ha et al., 2017)	24.56%	10.82	24.77%	2.18
CASRL (Chen et al., 2021)	11.18%	3.75	17.92%	2.22
PCT-continues (Zhao & Xu, 2022)	56.59%	**19.50**	41.18%	3.22
A2C (Mnih et al., 2016)	55.24%	19.15	40.77%	3.11
DACC (w/o rules)	52.56%	18.71	38.71%	3.03
DACC	**57.34%**	19.40	**43.49%**	**3.25**

4.2 Elevator group scheduling problem

Problem Setting. The elevator group scheduling problem (EGSP) mainly studies assigning elevators to respond to a flood of passengers in buildings equipped with multiple elevators. Among various EGSP systems, we consider the most prevalent one: at every time step, stochastic passengers appear and give hall call requests with up/down directions; the system assigns an elevator for every hall call request without knowing passengers' destination floors until they get on the elevator; also, passengers do not know the assigned elevators so that they can be reassigned at any time.

Environment Dynamics. Passenger flow in the EGSP often shows specific traffic patterns within a day, which can be divided into three major periods (Jansson & Uggla Lingvall, 2015): up-peak traffic in morning rush hours, two-way traffic at lunchtime, and down-peak traffic in evening rush hours.

Realistic Constraints. There are some commonly-used constraints for the elevator assignment and movement. 1) The EGSP system is required to dispatch exactly one elevator for every hall call request; 2) The assigned elevator must respond to the hall call request while completing the car calls generated by passengers on it (a car call indicates the passenger's destination floor and is registered at the elevator after the passenger enters it; 3) The elevator can not change its service direction until there is no hall call or car call in the current direction.

4.2.1 Implementation Components

Simulator. To mimic the real-world behavior of EGSP, based on the only available open-source simulator Liftsim (Wang et al., 2020), we further improve it into a more realistic EGSP simulator that satisfies the three realistic constraints mentioned in the last paragraph. Additionally, our simulator can simulate passenger flow following the three traffic patterns. Detailed comparisons among existing simulators are explained in Appendix.

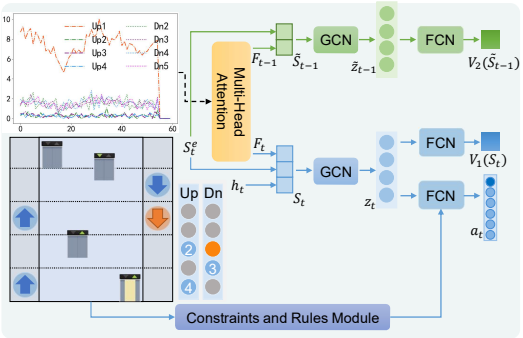

Figure 5: Our framework on the EGSP. We use a multi-head attention network to extract features from historical hall call data. We use graph convolutional networks (GCN) as primary and secondary encoders and adopt fully-connected networks (FCN) as the critics and the actor.

MDP. We formulate the EGSP as a two-stage MDP in our simulator: in every simulation time step, new passengers are generated with a probability matrix corresponding to the specific pattern; the agent receives the state information from the simulator and makes a decision to assign elevators to exist unprocessed hall calls. As illustrated in Fig.5, the state includes the elevator state S_t^e and the hall call stateh_t. The elevator state includes the position, speed, service direction, door status, and registered car call of all elevators. And the hall call state includes the unprocessed requests (marked orange in Fig.5), the existing hall call requests, and their corresponding assigned elevators. The action is defined as assigning an elevator for each unprocessed hall call, and the objective is to reduce the average waiting time (denoted as T_{aw}) and the average transmitting time (denoted as T_{at}) of all passengers. At the same time save energy consumption. Here the waiting time of a passenger is the time taken from the moment he presses the hall call button on a floor to the instant he is picked up. The transmitting time is the time taken from when one is picked up to

Table 2: Performance comparison between the baselines and **DACC** in three traffic patterns in terms of two criteria: T_{as} (the sum of T_{aw} and T_{aw}) and TE (a criteria that integrates T_{as} and energy consumption E). The best results are highlighted in **bolded**.

Method	Two-way		Up-peak		Down-peak	
	$T_{as}\downarrow$	$TE\downarrow$	$T_{as}\downarrow$	$TE\downarrow$	$T_{as}\downarrow$	$TE\downarrow$
ETA (Rong et al., 2003)	92.50	110.98	119.83	142.47	**65.30**	70.50
SFM (Ramalingam et al., 2017)	90.99	110.67	119.27	141.98	70.63	75.37
Robert (Crites & Barto, 1998)	159.31	261.70	126.70	247.87	177.42	209.21
DRL-EGC (Wei et al., 2020)	112.93	134.89	123.60	151.39	82.21	86.40
Context-QL (Padakandla et al., 2020)	91.14	111.18	114.89	137.03	66.34	71.82
DACC	**89.75**	**108.14**	**110.21**	**132.15**	65.83	**70.35**

the instant one reaches the destination floor. Thus the reward function is defined as (0.001 is the discount coefficient)

$$r_t = -(\beta|N_q| + \xi|N_c| + E), \tag{8}$$

Where N_q, N_c is the number of passengers waiting for elevators in the hall, and those in the elevators, respectively, as weighted by hyperparameter α, β. While E is the energy consumption measured by electricity cost.ely as weighted by hyperparameter α, β. While E is the energy consumption measured by electricity cost.

Network Architecture. As shown in Fig.5, we use a multi-head attention network (Vaswani et al., 2017) as the dynamics aware module to capture the trend of passenger flow and learn the hidden features from historical hall call data. We record the number of generated hall calls on each floor every minute during the whole simulation process. For the current time step t, we forward the collected hall call data in the last 30 minutes to the attention network and finally use a two-layer FC network to output the latent feature vector distribution F_t. We use two four-layer graph convolutional networks (GCN) (Kipf & Welling, 2016) as the primary encoder and the secondary encoder, respectively. And we use two-layer fully-connected networks (FCN) as the two critic networks and the actor-network, respectively. For the constraints and rules module, we incorporate the estimated time of arrival algorithm (ETA) (Rong et al., 2003) to score each action at every time step and generate the action mask to guide the exploration of the RL agent. With the action mask, we can avoid illegal actions such as assigning two elevators to the same hall call and reduce the occurrence of inefficient actions such as assigning an elevator that is running oppositely to a hall call.

4.2.2 Experimental Results

Dataset and Hyperparameters. We first collect passenger flow data in 30 days respectively for two-way traffic pattern, up-peak pattern, and down-peak pattern from a real-world 16-floor and 4-elevator building, with nearly 50 persons residing on each floor except the first floor. For each pattern, we calculate a probability matrix $\mathbf{P}_t$ that represents the generating probability of a passenger who appears at floor i and targets floor j at time t and use $\mathbf{P}_t$ to generate passengers randomly. We set $\alpha = 0.01$, $\gamma = 0.99$ in Eq.5, $\beta = 1$, $\xi = 0.6$ in the reward Eq.8, and the learning rate as 2×10^{-5}.

Compared Baselines. We compare our model with traditional elevator group scheduling algorithms, combinatorial optimization-based and learning-based algorithms on the three passenger flow patterns. The popular conventional algorithms used for the EGSP include Round-Robin, estimated time of arrival (ETA) (Rong et al., 2003), and genetic algorithm (GA), among which ETA achieve the best performances in most scenarios. The SFM (Ramalingam et al., 2017) algorithm for EGSP is a combinatorial optimization-based algorithm adopted by commercial elevator companies. Learning-based algorithms for EGSP include Robert (Crites & Barto, 1998) (using Q-Learning) and DRL-EGC (Wei et al., 2020) (using the A3C (Mnih et al., 2016) algorithm). Additionally, we train a context detection method for non-stationary environments Context Q-learning (Padakandla et al., 2020) in our environment as a baseline.

Results. The comparison results are shown in Tab.2. We can see that our framework outperforms all existing methods for EGSP in all of the three traffic patterns in terms of the TE criteria, which takes the average waiting time T_{at}, the average transmitting time T_{at} and energy consumption E into consideration. For the T_{as} criteria, which only considers the average waiting time and transmitting time, our framework outperforms all existing methods in the two-way traffic and up-peak patterns and achieves comparable performance with the best algorithm in the dn-peak pattern.

REFERENCES

Ts2pack: A two-level tabu search for the three-dimensional bin packing problem. *European Journal of Operational Research*, 195(3):744–760, 2009. ISSN 0377-2217.

A modified squirrel search algorithm based on improved best fit heuristic and operator strategy for bin packing problem. *Applied Soft Computing*, 82:105565, 2019. ISSN 1568-4946.

Hyper-heuristic local search for combinatorial optimisation problems. *Knowledge-Based Systems*, 205:106264, 2020. ISSN 0950-7051.

Eitan Altman. *Constrained Markov decision processes: stochastic modeling*. Routledge, 1999.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR, 06–11 Aug 2017.

Judith O Berkey and Pearl Y Wang. Two-dimensional finite bin-packing algorithms. *Journal of the operational research society*, 38(5):423–429, 1987.

Yash Chandak, Scott Jordan, Georgios Theocharous, Martha White, and Philip S Thomas. Towards safe policy improvement for non-stationary mdps. *Advances in Neural Information Processing Systems*, 33:9156–9168, 2020.

Bernard Chazelle. The bottomn-left bin-packing heuristic: An efficient implementation. *IEEE Transactions on Computers*, C-32:697–707, 1983.

Baiming Chen, Zuxin Liu, Jiacheng Zhu, Mengdi Xu, Wenhao Ding, Liang Li, and Ding Zhao. Context-aware safe reinforcement learning for non-stationary environments. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10689–10695, 2021. doi: 10.1109/ICRA48506.2021.9561593.

Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *NeurIPS*, 2019.

Teodor Gabriel Crainic, Guido Perboli, and Roberto Tadei. Extreme point-based heuristics for three-dimensional bin packing. *INFORMS Journal on Computing*, 20:368–384, 08 2008.

Robert H Crites and Andrew G Barto. Elevator group control using multiple reinforcement learning agents. *Machine learning*, 33(2):235–262, 1998.

Bruno C. da Silva, Eduardo W. Basso, Ana L. C. Bazzan, and Paulo M. Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 217–224, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832.

Denise de Oliveira, Ana LC Bazzan, Bruno Castro da Silva, Eduardo W Basso, Luis Nunes, Rosaldo Rossetti, Eugénio de Oliveira, Roberto da Silva, and Luis Lamb. Reinforcement learning based control of traffic lights in non-stationary environments: A case study in a microscopic simulator. In *EUMAS*, 2006.

Jiahui Duan, Xialiang Tong, Fei Ni, Zhenan He, Lei Chen, and Mingxuan Yuan. A data-driven column generation algorithm for bin packing problem in manufacturing industry, 2022.

Samir Elhedhli, Fatma Gzara, and Burak Yildiz. Three-dimensional bin packing and mixed-case palletization. *INFORMS Journal on Optimization*, 1(4):323–352, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 06–11 Aug 2017.

Chi Trung Ha, Trung Thanh Nguyen, Lam Thu Bui, and Ran Wang. An online packing heuristic for the three-dimensional container loading problem in dynamic environments and the physical internet. In Giovanni Squillero and Kevin Sim (eds.), *Applications of Evolutionary Computation*, pp. 140–155, Cham, 2017. Springer International Publishing.

Emmanuel Hadoux, Aurélie Beynier, and Paul Weng. Sequential Decision-Making under Non-stationary Environments via Sequential Change-point Detection. In *Learning over Multiple Contexts (LMCE)*, Nancy, France, September 2014.

Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Anton Jansson and Kristoffer Uggla Lingvall. Elevator control using reinforcement learning to select strategy, 2015.

Kyungdaw Kang, Ilkyeong Moon, and Hongfeng Wang. A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem. *Applied Mathematics and Computation*, 219(3):1287–1299, 2012.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Xueping Li and Kaike Zhang. A hybrid differential evolution algorithm for multiple container loading problem with heterogeneous containers. *Computers & Industrial Engineering*, 90:305–313, 2015.

Eunice López-Camacho, Gabriela Ochoa, Hugo Terashima-Marín, and Edmund K Burke. An effective heuristic for the two-dimensional irregular bin packing problem. *Annals of Operations Research*, 206(1):241–264, 2013.

Hongzi Mao, Shaileshh Bojja Venkatakrishnan, Malte Schwarzkopf, and Mohammad Alizadeh. Variance reduction for reinforcement learning in input-driven environments. *CoRR*, abs/1807.02264, 2018.

Silvano Martello and Daniele Vigo. Exact solution of the two-dimensional finite bin packing problem. *Management science*, 44(3):388–399, 1998.

Silvano Martello, David Pisinger, and Daniele Vigo. The three-dimensional bin packing problem. *Operations research*, 48(2):256–267, 2000.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning, 2018.

Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. In *NeurIPS*, 2018.

Sindhu Padakandla, Prabuchandran KJ, and Shalabh Bhatnagar. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50(11):3590–3606, 2020.

Peter. Elevate 8: a commercial-level elevator simulator. https://www.peters-research.com/index.php/8-elevate/58-elevate-8.

Srikumar Ramalingam, Arvind U Raghunathan, and Daniel Nikovski. Submodular function maximization for group elevator scheduling. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*, 2017.

Aiying Rong, Henri Hakonen, and Risto Lahdelma. *Estimated time of arrival (ETA) based elevator group control algorithm with more accurate estimation*. Turku Centre for Computer Science, 2003.

Reazul Hasan Russel, Mouhacine Benosman, and Jeroen van Baar. Robust constrained-mdps: Soft-constrained robust policy optimization under model uncertainty. *CoRR*, abs/2010.04870, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

George R Strakosch. *Vertical transportation: elevators and escalators*. Wiley, 1983.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Fan Wang, Bo Zhou, Yunxiang Li, and Kejiao Li. Liftsim: a configurable lightweight simulator of elevator systems. https://github.com/PaddlePaddle/MetaGym/tree/master/metagym/liftsim, 2020.

Chen-Yu Wei and Haipeng Luo. Non-stationary reinforcement learning without prior knowledge: An optimal black-box approach. In *Conference on Learning Theory*, pp. 4300–4354. PMLR, 2021.

Qinglai Wei, Lingxiao Wang, Yu Liu, and Marios M. Polycarpou. Optimal elevator group control via deep asynchronous actor–critic learning. *IEEE Transactions on Neural Networks and Learning Systems*, 31(12):5245–5256, 2020. doi: 10.1109/TNNLS.2020.2965208.

Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *ECCV*, September 2018.

Mengdi Xu, Wenhao Ding, Jiacheng Zhu, ZUXIN LIU, Baiming Chen, and Ding Zhao. Task-agnostic online reinforcement learning with an infinite mixture of gaussian processes. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6429–6440. Curran Associates, Inc., 2020.

Hang Zhao and Kai Xu. Learning efficient online 3d bin packing on packing configuration trees. In *International Conference on Learning Representations*, 2022.

Hang Zhao, Qijin She, Chenyang Zhu, Yin Yang, and Kai Xu. Online 3d bin packing with constrained deep reinforcement learning. *arXiv preprint arXiv:2006.14978*, 2020.

Hang Zhao, Chenyang Zhu, Xin Xu, Hui Huang, and Kai Xu. Learning practically feasible policies for online 3d bin packing. *Science China Information Sciences*, 65(1):1–17, 2022.

Liu zheng, Shu Guang, and Dong Hui. Research of elevator group scheduling system based on reinforcement learning algorithm. In *Proceedings of 2013 2nd International Conference on Measurement, Information and Control*, volume 01, pp. 606–610, 2013.

Qianwen Zhu, Xihan Li, Zihan Zhang, Zhixing Luo, Xialiang Tong, Mingxuan Yuan, and Jia Zeng. Learning to pack: A data-driven tree search algorithm for large-scale 3d bin packing problem. In *Proceedings of the 30th ACM International Conference on Information amp; Knowledge Management*, CIKM '21, pp. 4393–4402, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384469.

A BPP HIERARCHICAL ARCHITECTURE

The framework of BPP placement planner is a two-level hierarchical Actor-Critic with an attention mechanism. Facing the Exploding action space, we decompose the action space (whole pallet) into a certain number of sub-regions. The high-level policy selects a sub-region with an attention mechanism for the low level to achieve, and the low level selects a specific packing position in the sub-region. As the action space decreases, the corresponding mask also decreases, and the amount of calculation becomes smaller.

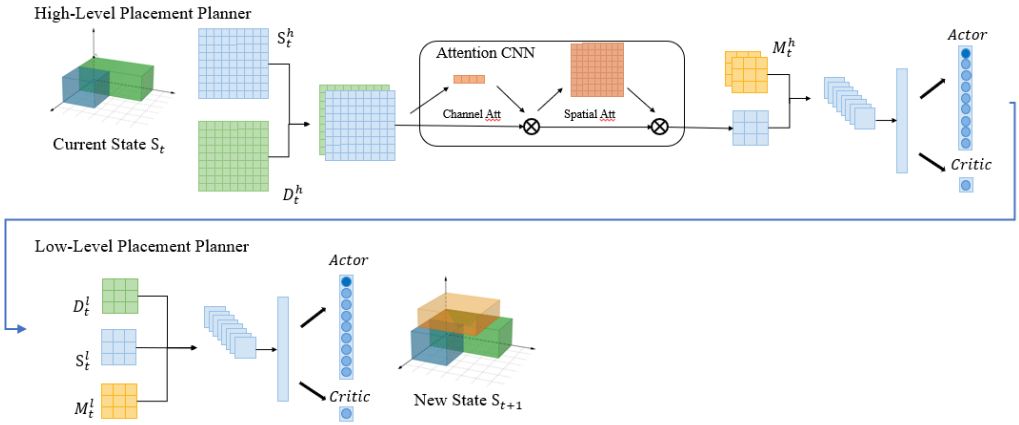

Figure 6: Overview of hierarchical architecture. Since invalid action mask calculation takes extended time and is essential for decision security, we need to decouple the action space to reduce masks calculation. With hierarchical architecture, the entire space of the pallet (9*9) is divided into small regions (3*3), the high-level agent selects a feasible subspace for low-level, and the low-level agent selects a specific position to pack. Decoupling of action space reduces the size of invalid action masks (colored in yellow) thus reducing computation time.

B OUR ENHANCED SIMULATOR OF THE EGSP

The existing EGSP simulator Liftsim (Wang et al., 2020) fails to meet realistic constraints mentioned in §4.1 and the commercial EGSP software Elevate8 (Peter) is not open-source for training RL agents. To better reflect the mechanism of a commercial elevator scheduler, we first develop our simulator based on Liftsim, satisfying all the realistic constraints of the real-world EGSP, and we implement both the up-down mode and destination mode for further study. As for dynamic passenger data, instead of randomly generating passenger data with fix probability factors, we collect real-world passenger data from buildings that conform to different patterns and enhance the data with randomness. Tab. 3 shows the differences between LiftSim, Elevate 8 and our simulator.

Table 3: Major differences among three simulators: LiftSim, Elevate 8, and our self-developed engine which will be open-sourced.

Environment	LiftSim	Elevate 8	Ours
Meet realistic constraints	×	√	√
Realistic passenger data	×	√	√
Support up-down mode	√	√	√
Support destination mode	×	√	√
Provide interface for RL	√	×	√
Open source	√	×	√

C ABLATION STUDY ON THE EGSP

We take an ablation study to evaluate the advantages of the dynamics aware module and combining rules. In this experiment, we evaluate four agents on a 120-min' s passenger data flow that follows

the two-way traffic mode: the agent without the dynamics aware module (DAM) and the rules module (RM), the agent without the rules module, the agent without the dynamics aware module and the full version of DACC agent. Assuming that the objective in this scenario is to reduce the average waiting time and the average transmitting time. From the experimental result shown in Tab.4, we can find that both the dynamics aware module and the rules module play important roles. By combining the advantages of soft rules and dynamics aware, our framework, DACC, can inherit the controllability and reliability of rules and further optimize the customized objectives for different scenarios.

Table 4: Comparison among different agents in two-way traffic pattern in terms of T_{aw} and T_{as}. We set the agent's goal as optimizing the mean waiting time and the average transmitting time.

Agent	$T_{aw} \downarrow$	$T_{as} \downarrow$
DACC w/o DAM and RM	111.73	173.55
DACC w/o RM	65.44	128.02
DACC w/o DAM	35.13	91.08
DACC	32.79	89.36