# Deep Reinforcement Learning for Scalable Offline Three-Dimensional Packing

**Hao Yin**[1], **Hongjie He**[1*], **Fan Chen**[1]

[1]Southwest Jiaotong University, China
haoyin@my.swjtu.edu.cn, hjhe@swjtu.edu.cn, fchen@swjtu.edu.cn

## Abstract

With the increasing number of items requiring handling simultaneously in complex logistics, offline three-dimensional packing methods need to plan larger numbers of items. Existing deep reinforcement learning (DRL)-based packing methods cannot plan for large numbers of items while keeping high-quality solutions due to limited exploration space and high computational complexity. To address this issue, this paper proposes a scalable DRL-based packing method. An attention-based pack-Q-network (PQNet) is constructed to learn the optimal packing policy by integrating unpacked items, available spaces, and packed items. To expand the valid exploration space, a bidding-based multi-policy (BBMP) framework composed of multiple PQNets is designed to efficiently explore more latent valid solutions, thus enhancing solution quality. To reduce computational complexity, a training-free dynamic candidate selection (DCS) framework is proposed to incorporate comprehensive item information during execution with minimal computation overhead, which helps in effectively planning large numbers of items. Experimental results show that across item numbers of 20∼1000, our method consistently outperforms the best-performing baseline at each tested scale by 3.2%∼13.1% in space utilization.

**Code** — https://github.com/Ashenone511/BBMP-DCS

## Introduction

The offline three-dimensional packing problem (3D-PP) is a practical combinatorial optimization problem widely studied in logistics, warehousing, and manufacturing (Lin et al. 2024; Yao et al. 2025). As real-world applications grow more complex, the number of items that need to be planned for 3D-PP has significantly increased. There is an urgent demand for a scalable packing method to solve offline 3D-PP involving large numbers of items.

The offline 3D-PP is known as an NP-hard problem (Bortfeldt and Wäscher 2013), focusing on packing a set of cuboid-shaped items into cuboid-shaped containers, aiming to maximize the space utilization. The NP-hard nature of the problem prevents it from being solved exactly, as exact mathematical methods (Martello, Pisinger, and Vigo 2000; Tsai, Wang, and Lin 2015) often suffer from heavy time consumption and suboptimal probabilities of finding feasible solutions. Hence, using heuristics to obtain approximate solutions within reduced time has been a predominant approach (Ali et al. 2022; Yang et al. 2024; Heßler, Hintsch, and Wienkamp 2025). However, heuristics usually rely on handcrafted rules, which limits their performance and makes them poorly adaptable to diverse packing scenarios (Pan, Chen, and Lin 2023).

Recently, the powerful decision-making capability of deep reinforcement learning (DRL) has attracted studies to explore its application in offline 3D-PP (Hu et al. 2017; Duan et al. 2019). As a key advantage over conventional methods, DRL enables policy exploration during training, thereby obtaining high-quality solutions with minimal overhead during execution (Liu et al. 2023). Moreover, guided by appropriate reward signals, the agent can learn underlying packing patterns to adapt to various packing scenarios with improved performance. However, developing effective and scalable DRL-based packing methods remains challenging. On the one hand, the agent's large action space for item placement causes difficulties in policy exploration, leading to degraded solution quality (Zhao et al. 2021). The dominant approach for this issue is to integrate heuristics to mask out unreasonable placement (Yang et al. 2023; Que, Yang, and Zhang 2023; Xu et al. 2023; Yin, Chen, and He 2024; Xiong et al. 2024; Wang et al. 2025), thus compressing the action space and enhancing exploration efficiency. However, such a heuristic-driven DRL paradigm explores within the narrow space limited by heuristic rules, which results in the neglect of many latent valid solutions and makes the agent prone to getting stuck in local optima. On the other hand, large numbers of items increase memory usage and computational complexity, as existing well-performing DRL-based methods primarily use attention mechanisms (Vaswani et al. 2017) to encode items, whose computational overhead is susceptible to the sequence length.

To address the above issues, this paper proposes an effective and scalable DRL-based method for solving the offline 3D-PP. Firstly, we develop an attention-based pack-Q-network (PQNet) to capture the dependencies among information regarding unpacked items, available spaces, and packed items to derive action values. Secondly, we propose a bidding-based multi-policy (BBMP) framework that consid-

*Corresponding author.

ers multiple packing directions to enrich the packing policy and explore more valid solutions while keeping high exploration efficiency. Finally, we divide all unpacked items into a processing set and a candidate set. A training-free dynamic candidate selection (DCS) framework is proposed to move the optimal candidate item into the processing set using the pre-trained BBMP model parameters. DCS framework incorporates comprehensive item information during execution with minimal computation overhead, thereby enhancing performance for large numbers of items. Extensive experimental results show that the proposed method achieves state-of-the-art performance on 3D-PP with 20$\sim$1000 items. The method also exhibits generalization capabilities across varying container sizes and item numbers.

## Related Works

Throughout the evolution of 3D-PP, numerous methods have emerged, which can be broadly categorized into five classes.

1) *Exact algorithms* (Chen, Lee, and Shen 1995; Martello, Pisinger, and Vigo 2000) and 2) *approximation algorithms* (Chung, Garey, and Johnson 1982; Gálvez et al. 2021) focus on theoretically ensuring optimal and near-optimal solutions, but they are limited by high computational complexity and poor scalability for large-scale instances.

3) *Constructive heuristics* aim to find feasible solutions based on handcrafted rules, which usually place items into spaces based on loading patterns (George and Robinson 1980; Toffolo et al. 2017; Saraiva, Nepomuceno, and Pinheiro 2015; Yang et al. 2024), with these spaces being managed using specialized techniques (Bortfeldt, Gehring, and Mack 2003; Crainic, Perboli, and Tadei 2008; Parreño et al. 2008). However, the reliance on handcrafted rules makes them struggle to adapt to scenario changes.

4) *Search algorithms* attempt to explore search spaces while adhering to constructive heuristic rules. Metaheuristic methods (Liu et al. 2011; Gonçalves and Resende 2013; Liu et al. 2017; Shao and Xiao 2025) are commonly used to improve the quality of complete solutions iteratively. Tree search methods (Ren, Tian, and Sawaragi 2011) create solution branches based on the process of packing and selecting the best one among them. These methods are effective in finding high-quality solutions but come at the cost of significant time inefficiency.

5) *Deep reinforcement learning (DRL)-based methods* are a class of methods that emerged recently as the flourishes of DRL. (Hu et al. 2017; Duan et al. 2019) first used the pointer network (Vinyals, Fortunato, and Jaitly 2015) as the policy network to solve the offline 3D-PP for minimizing the surface area of the bounding cuboid. Conditional query learning (CQL) (Li et al. 2020) solves the offline 3D-PP to minimize the variable container height (named 3D strip packing problem (3D-SPP)), by embedding previous actions as a conditional query to the attention model. Building upon CQL, recurrent CQL (RCQL) (Li et al. 2022) further manages a candidate queue with the head-of-queue items being moved to the processing set at each time step, thus scaling the number of planable items to thousands. (Jiang, Cao, and Zhang 2023) improved CQL by introducing the sparse attention mechanism to reduce computational complexity. The action representation learning was employed to handle the large action space for placement. (Que, Yang, and Zhang 2023) learned a Transformer-based policy network to guide packing. A plane feature was introduced to represent the bin state and compressed the action space by heuristically downsampling these features. (Yin, Chen, and He 2024) constructed a two-stage policy network and heuristically enforced item packing along two orthogonal directions to compress the action space and enhance exploration efficiency. Most DRL-based methods cannot plan for large numbers of items while keeping high-quality solutions due to limited exploration space and high computational complexity. In this paper, we aim to propose an effective and scalable DRL-based method to address this issue.

## Preliminaries

### Problem Formulation

We focus on the 3D-SPP, a variant of the offline 3D-PP aimed at minimizing container height. This section provides a detailed description of this problem.

Given a cuboid-shaped container with a fixed length $L$, width $W$, and variable height, and a set $\mathcal{I} = \{1, 2, \dots, N\}$ of cuboid-shaped items along with their length $l_i$, width $w_i$, and height $h_i$ of item $i$, the goal is to pack all items from $\mathcal{I}$ into the container within $N$ time steps. At each time step, one item is selected from the set $\mathcal{U} \subseteq \mathcal{I}$ to pack into the container (to move to the set $\mathcal{P} \subseteq \mathcal{I}$), where $\mathcal{U}$ records all remaining unpacked items, and $\mathcal{P}$ records all packed items.

The packing space can be described in a Cartesian coordinate system with the origin located at the behind-left-bottom corner of the container. The vector $(x_i, y_i, z_i)$ represents the coordinate of the behind-left-bottom corner of the packed item $i$, while $(l'_i, w'_i, h'_i)$ represents its dimensions after be rotated and packed. The objective of 3D-SPP is to minimize container height (i.e., maximize the space utilization) after packing all the items in $\mathcal{I}$:

$$\text{minimize} \quad \max_{i \in \mathcal{I}} (z_i + h'_i) \,. \tag{1}$$

Each packing operation must satisfy constraints such as non-overlapping and boundary conditions, and the formal definition of 3D-SPP is provided in Appendix A.

### Markov Decision Process

To learn an effective packing policy via DRL, we first formulate 3D-SPP as a Markov decision process (MDP) characterized by a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S}$ is a set of packing states; $\mathcal{A}$ is a set of packing actions; $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the deterministic transition function; $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function; $\gamma$ is the discount factor.

In our formulation, the state $s_t \in \mathcal{S}$ at time step $t$ consists of unpacked state $s_t^u$, packed state $s_t^p$, and space state $s_t^s$:

$$\begin{aligned} s_t^u &= \{(l_i, w_i, h_i) | i \in \mathcal{U}\}, \\ s_t^p &= \{(x_i, y_i, z_i, x'_i, y'_i, z'_i, l'_i, w'_i, h'_i) | i \in \mathcal{P}\}, \\ s_t^s &= \{(x_j^s, y_j^s, z_j^s, x_j'^s, y_j'^s, z_j'^s, l_j^s, w_j^s, h_j^s) | j \in \mathcal{E}\}, \end{aligned} \tag{2}$$

where $(x'_i, y'_i, z'_i)$ is the coordinate of the front-right-top corner of item $i$ in $\mathcal{P}$, $(x_j^s, y_j^s, z_j^s)$, $(x_j'^s, y_j'^s, z_j'^s)$, and

$(l_j^s, w_j^s, h_j^s)$ stand for the coordinate of the behind-left-bottom corner, the front-right-top corner, and the dimensions of space $j$, and the set $\mathcal{E}$ records all available spaces managed by EMS (Parreño et al. 2008). The action $a_t \in \mathcal{A}$ taken at time step $t$ represents selecting an item from $\mathcal{U}$ and determining its rotation and placement. By embracing EMS, the decision for placement can be replaced by selecting space from $\mathcal{E}$ as the item can be placed at one corner of the selected space. As each item has 6 possible rotations, the action space size $|\mathcal{A}| = 6|\mathcal{U}||\mathcal{E}|$. The reward function $R$ is defined as the same as in (Li et al. 2020):

$$\begin{cases} r_t = g_{t-1} - g_t, \\ g_t = LW\widetilde{H}_t - \sum_{i \in \mathcal{P}} (l_i w_i h_i), \end{cases} \tag{3}$$

where $\widetilde{H}_t = \max_{i \in \mathcal{P}} (z_i + h_i')$. The goal of DRL is to optimize the policy $\pi : \mathcal{S} \to \mathcal{A}$ to maximize the expected return with the discount factor $\gamma$:

$$J(\pi) = \mathbb{E}_\pi \left[ \sum_{t=1}^{N} \gamma^{t-1} r_t \right]. \tag{4}$$

# Method

## Pack-Q-Network

To optimize the policy $\pi$, we use an iteratively updated action value function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ to guide the agent's decision-making process. The $Q$-function estimates the expected return starting from a given state $s_t$ and taking an action $a_t$ under the current policy:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1:N}, a_{t+1:N}} \left[ \sum_{i=0}^{N-t} \gamma^i r_{t+i} \middle| s_t, a_t \right]. \tag{5}$$

To find the optimal $Q$-function $Q^*(s, a)$, we represent the function with a pack-Q-network (PQNet) parameterized by $\theta$, and learn $\theta$ to minimize the squared TD error (Mnih et al. 2015). Fig. 1 (a) illustrates PQNet, whose structure is similar to the policy network in (Xu et al. 2023). The main difference is that PQNet introduces an encoding of the packed items in the container to further represent the occupied area, and removes all position encodings because the unpacked items do not have a sequential property, while the packed and space states already contain the positional information. Specifically, several linear layers are used to encode the rotated unpacked state

$$\begin{aligned} s_t^{ru} =& \text{Rotate}(s_t^u) \\ =& \{(l_i, w_i, h_i), (l_i, h_i, w_i), (w_i, l_i, h_i), (w_i, h_i, l_i), \\ & (h_i, l_i, w_i), (h_i, w_i, l_i) | i \in \mathcal{U}\}. \end{aligned} \tag{6}$$

The obtained sequence $h_t^{ru,(0)} \in \mathbb{R}^{6|\mathcal{U}| \times d_m}$ is then processed by a Transformer encoder to capture dependencies among items under various rotations:

$$h_t^{ru,(l)} = \text{FFB}(\text{MAB}(h_t^{ru,(l-1)}, h_t^{ru,(l-1)})), \tag{7}$$

where $h_t^{ru,(l)}$ represents the output of the encoding layer $l \in \{1, \ldots, N_e\}$. The multi-head attention block (MAB) and feed-forward block (FFB) are defined as:

$$\begin{cases} \text{MAB}(X, Y) = \text{LN}(X + \text{MHA}(X, Y, Y)), \\ \text{FFB}(H) = \text{LN}(H + \text{FF}(H), \end{cases} \tag{8}$$

where LN, MHA, and FF refer to layer normalization, multi-head attention layer, and feed-forward layer, respectively.

Moreover, a Transformer encoder is used to encode $s_t^p$ into $h_t^{p,(N_e)} \in \mathbb{R}^{|\mathcal{P}| \times d_m}$, and several linear layers are used to encode $s_t^s$ into $h_t^{s,(0)} \in \mathbb{R}^{|\mathcal{E}| \times d_m}$. $h_t^{p,(N_e)}$ and $h_t^{s,(0)}$ are then fused by the Transformer decoder using a cross-attention mechanism to integrate packed items into space features:

$$\begin{cases} H = \text{MAB}(h_t^{s,(l-1)}, h_t^{s,(l-1)}), \\ h_t^{s,(l)} = \text{FFB}(\text{MAB}(H, h_t^{p,(N_e)})), \end{cases} \tag{9}$$

where $h_t^{s,(l)}$ represents the output of the decoding layer $l \in \{1, \ldots, N_d\}$.

Finally, the dot product between vectors in $h_t^{ru,(N_e)}$ and $h_t^{s,(N_d)}$ is calculated to obtain a matrix $\mathbf{M} \in \mathbb{R}^{6|\mathcal{U}| \times |\mathcal{E}|}$. $\mathbf{M}$ can be used to determine a valid packing index (i.e., selecting which item to pack), rotation, and placement, as it allows the model to learn which features of the rotated unpacked items and spaces match most effectively. $\mathbf{M}$ is then flattened and multiplied by a feasibility mask (used to mask out infeasible packings), deriving action values $Q(s_t, \cdot) \in \mathbb{R}^{6|\mathcal{U}||\mathcal{E}|}$.

## Bidding-Based Multi-Policy Framework

The policy learned by a single PQNet has limitations because it dictates that each item must be placed at one corner of the space, resulting in the neglect of many valid solutions. In fact, the four corners of the bottom of the spaces should all be able to be placed. To explore more latent valid solutions, we propose a bidding-based multi-policy (BBMP) framework to allocate four policies to four PQNets, where each network is responsible for estimating the action values for packing items into one of the corners of spaces: the behind-left-bottom corner, the front-left-bottom corner, the front-right-bottom corner, and the behind-right-bottom corner. To avoid the non-stationary problem and conflicts among four policies, two special schemes are implemented: *parameter sharing* and *action bidding*.

1) *Parameter sharing* is a commonly used scheme in multi-agent reinforcement learning, which effectively coordinates the behavior of homogeneous agents and enhances training efficiency (Li, Wang, and Xu 2025; Chen et al. 2025). By treating each PQNet as an independent agent, we can homogenize the four PQNets by adjusting their observations individually. Specifically, for a given state $s_t = \langle s_t^u, s_t^p, s_t^s \rangle$, the observations of the four PQNets are adjusted as follows:

$$\begin{cases} o_t^1 = \langle o_t^u, o_t^{p,1}, o_t^{s,1} \rangle \in \mathcal{O}^1, \\ o_t^u = \{(l_i, w_i, h_i) | i \in \mathcal{U}\}, \\ o_t^{p,1} = \{(x_i, y_i, z_i, x_i', y_i', z_i', l_i', w_i', h_i') | i \in \mathcal{P}\}, \\ o_t^{s,1} = \{(x_j^s, y_j^s, z_j^s, x_j'^s, y_j'^s, z_j'^s, l_j^s, w_j^s, h_j^s) | j \in \mathcal{E}\}, \end{cases} \tag{10}$$

$$\begin{cases} o_t^2 = \langle o_t^u, o_t^{p,2}, o_t^{s,2} \rangle \in \mathcal{O}^2, \\ o_t^u = \{(l_i, w_i, h_i) | i \in \mathcal{U}\}, \\ o_t^{p,2} = \{(L - x_i', y_i, z_i, L - x_i, y_i', z_i', l_i', w_i', h_i') | i \in \mathcal{P}\}, \\ o_t^{s,2} = \{(L - x_j'^s, y_j^s, z_j^s, L - x_j^s, y_j'^s, z_j'^s, l_j^s, w_j^s, h_j^s) | j \in \mathcal{E}\}, \end{cases} \tag{11}$$

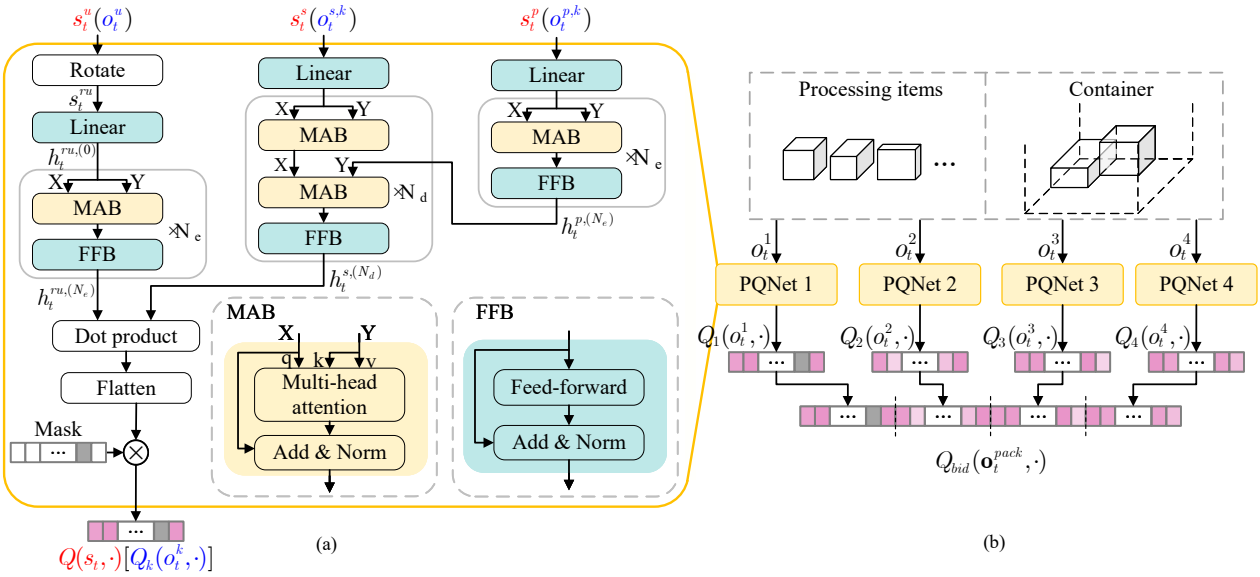


Figure 1: (a) Pack-Q-network (PQNet). Inputs and outputs are marked red for a single PQNet, which are replaced with blue for bidding-based multi-policy framework. (b) Bidding-based multi-policy (BBMP) framework.

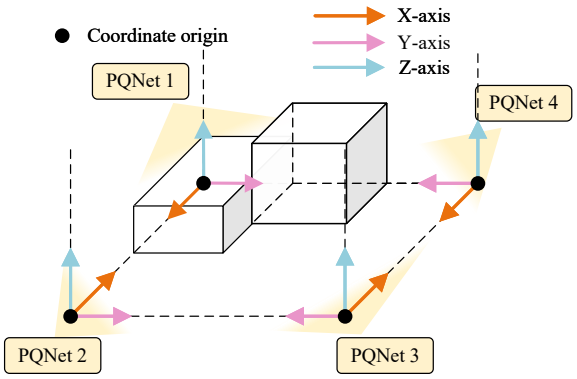


Figure 2: Observations of the four PQNets.

$$\begin{cases} o_t^3 = \langle o_t^u, o_t^{p,3}, o_t^{s,3} \rangle \in \mathcal{O}^3, \\ o_t^u = \{(l_i, w_i, h_i) | i \in \mathcal{U}\}, \\ o_t^{p,3} = \{(L-x_i', W-y_i', z_i, L-x_i, W-y_i, z_i', l_i', w_i', h_i') | i \in \mathcal{P}\}, \\ o_t^{s,3} = \{(L-x_j'^s, W-y_j'^s, z_j^s, L-x_j^s, W-y_j^s, z_j'^s, l_j^s, w_j^s, h_j^s) | j \in \mathcal{E}\}, \end{cases} \tag{12}$$

$$\begin{cases} o_t^4 = \langle o_t^u, o_t^{p,4}, o_t^{s,4} \rangle \in \mathcal{O}^4, \\ o_t^u = \{(l_i, w_i, h_i) | i \in \mathcal{U}\}, \\ o_t^{p,4} = \{(x_i, W-y_i', z_i, x_i', W-y_i, z_i', l_i', w_i', h_i') | i \in \mathcal{P}\}, \\ o_t^{s,4} = \{(x_j^s, W-y_j'^s, z_j^s, x_j'^s, W-y_j^s, z_j'^s, l_j^s, w_j^s, h_j^s) | j \in \mathcal{E}\}. \end{cases} \tag{13}$$

The above adjustment is equivalent to having the four PQNets observe the environment in coordinate systems with origins located at the four corners of the container, as shown in Fig. 2. Each PQNet learns a Q-function for its assigned coordinate system, ensuring they are homogeneous and enabling parameter sharing across them.

2) *Action bidding* is introduced to address the conflicts among multiple polices attempting to select the same item or place items in overlapping area. Specifically, we collect all action values computed by the four PQNets and select one and only one action from them as the final action. Let $Q_k(o^k, a^k)$ represents the $Q$-function of the PQNet $k \in \{1, 2, 3, 4\}$ from a given observation $o^k \in \mathcal{O}^k$ and taking an action $a^k \in \mathcal{A}^k$, where $a^k$ represents packing an item into spaces' corner closest to the origin corresponding to PQNet $k$, we have the bidding $Q$-function:

$$Q_{bid}(\mathbf{o}^{pack}, a^{pack}) = Q_k(o^k, a^k), \quad \text{if } a^{pack} \in \mathcal{A}^k, \tag{14}$$

where $\mathbf{o}^{pack} = \langle o^1, o^2, o^3, o^4 \rangle = \langle o^u, \langle o^{p,k}, o^{s,k} \rangle_{k=1}^4 \rangle$ represents the joint observation, and $a^{pack} \in \bigcup_{k=1}^4 \mathcal{A}^k$ represents the possible final action that can be selected.

The overall setup of the BBMP framework is shown in Fig. 1 (b). Each PQNet $k$ receives $o_t^k$ as input and derives its action values $Q_k(o_t^k, \cdot)$ at time step $t$. We concatenate $Q_1(o_t^1, \cdot) \sim Q_4(o_t^4, \cdot)$ to form the bidding action values $Q_{bid}(\mathbf{o}_t^{pack}, \cdot)$. The action $a_t^{pack} \in \bigcup_{k=1}^4 \mathcal{A}^k$ guided by $Q_{bid}$ integrates considerations of four policies while ensuring that decisions are coordinated and conflict-free.

## Dynamic Candidate Selection Framework

The computational complexity of attention mechanisms increase dramatically with the number of items. (Li et al. 2022) proposed storing most items in a candidate queue, with the head-of-queue items being moved to the processing set at each time step. Although this approach increases the number of processable items to thousands, it introduces performance degradation by neglecting items outside the processing set during decision-making. We believe fully considering all the information in the candidate queue contributes

to improving performance, and to this end, propose a dynamic candidate selection (DCS) framework.

Before the packing task begins (i.e., $\mathcal{U} = \mathcal{I}$ and $\mathcal{P} = \emptyset$), we divide the set of unpacked items $\mathcal{U}$ into two parts:

$$\mathcal{U}_{proc} = \begin{cases} \{1, 2, \ldots, N_{max}\}, & \text{if } N_{max} < N, \\ \{1, 2, \ldots, N\}, & \text{otherwise,} \end{cases} \tag{15}$$

$$\mathcal{U}_{cand} = \begin{cases} \{N_{max} + 1, \ldots, N\}, & \text{if } N_{max} < N, \\ \emptyset, & \text{otherwise,} \end{cases} \tag{16}$$

where $\mathcal{U}_{proc}$ and $\mathcal{U}_{cand}$ represent the processing set and candidate set, respectively, and $N_{max}$ is the maximum number of items that can be processed by the BBMP framework, limited by the memory usage during training. At each time step, one item is selected from $\mathcal{U}_{proc}$ to pack, and one candidate item is selected from $\mathcal{U}_{cand}$ (if any) to move to $\mathcal{U}_{proc}$.

We aim to learn an optimal function $Q^*_{sel}$ to guide the selection of candidate items for moving to $\mathcal{U}_{proc}$. At each time step $t$ after packing one item $i_p$, $Q^*_{sel}$ receives the observation $\mathbf{o}^{sel}_t = \langle o^{cand}_t, o^{proc}_{t'}, \langle o^{p,k}_{t+1}, o^{s,k}_{t+1} \rangle^4_{k=1} \rangle$ and outputs action values $Q^*_{sel}(\mathbf{o}^{sel}_t, \cdot)$, where the candidate observation $o^{cand}_t = \{(l_i, w_i, h_i) | i \in \mathcal{U}_{cand}\}$, the processing observation $o^{proc}_{t'} = \{(l_i, w_i, h_i) | i \in \mathcal{U}_{proc} \setminus \{i_p\}\}$, and $t'$ represents an intermediate step between $t$ and $t+1$. Each action $a^{sel}_t \in \mathcal{A}^{sel}$ ($|\mathcal{A}^{sel}| = |\mathcal{U}_{cand}|$) represents selecting an item from $\mathcal{U}_{cand}$ to move to $\mathcal{U}_{proc}$. Unfortunately, learning $Q^*_{sel}$ is challenging as $|\mathcal{U}_{cand}|$ can be extremely large, which leads to memory exhaustion during training. To overcome this issue, we devise a training-free DCS framework as shown in Fig. 3, where the parameters of linear layers, MAB, and FFB are derived from the pre-trained BBMP model.

Firstly, the rotation operations followed by several linear layers are employed to encode $o^{cand}_t$ and $o^{proc}_{t'}$ into $h^{cand,(0)}_t \in \mathbb{R}^{6|\mathcal{U}_{cand}| \times d_m}$ and $h^{proc,(0)}_{t'} \in \mathbb{R}^{6(|\mathcal{U}_{proc}|-1) \times d_m}$, respectively. For each layer $l \in \{1, \ldots, N_e\}$, perform:

$$h^{cand,(l)}_t = \text{FFB}(\text{MAB}(h^{cand,(l-1)}_t, h^{proc,(l-1)}_{t'})), \tag{17}$$

$$h^{proc,(l)}_{t'} = \text{FFB}(\text{MAB}(h^{proc,(l-1)}_{t'}, h^{proc,(l-1)}_{t'})). \tag{18}$$

The output $h^{cand,(N_e)}_t \in \mathbb{R}^{6|\mathcal{U}_{cand}| \times d_m}$ of the final layer encodes the features of all items in $\mathcal{U}_{cand}$ as if they were individually moved to $\mathcal{U}_{proc}$. This is achieved because all information of items in $\mathcal{U}_{cand}$ serves as the query in the attention layers, while the information of items in $\mathcal{U}_{proc} \setminus \{i_p\}$ is used as key-value pairs for relevance computation. This operation mirrors the actual process that would occur at $t+1$ step after individually moving these candidate items to $\mathcal{U}_{proc}$.

Subsequently, the vectors in $h^{cand,(N_e)}_t$ are respectively calculated the dot product with the vectors in $h^{s,1,(N_d)}_{t+1}, h^{s,2,(N_d)}_{t+1}, h^{s,3,(N_d)}_{t+1}, h^{s,4,(N_d)}_{t+1} \in \mathbb{R}^{|\mathcal{E}'| \times d_m}$, where $h^{s,k,(N_d)}_{t+1}$ represents the intermediate variable $h^{s,(N_d)}_{t+1}$ of the PQNet $k$ by taking $\langle o^{p,k}_{t+1}, o^{s,k}_{t+1} \rangle$ as input, and $\mathcal{E}'$ represents the space set at time step $t+1$. Each resulting matrix $\mathbf{M}^k \in \mathbb{R}^{6|\mathcal{U}_{cand}| \times |\mathcal{E}'|}$ is then reshaped to dimensions $|\mathcal{U}_{cand}| \times 6|\mathcal{E}'|$. The maximum value is taken along the second dimension of

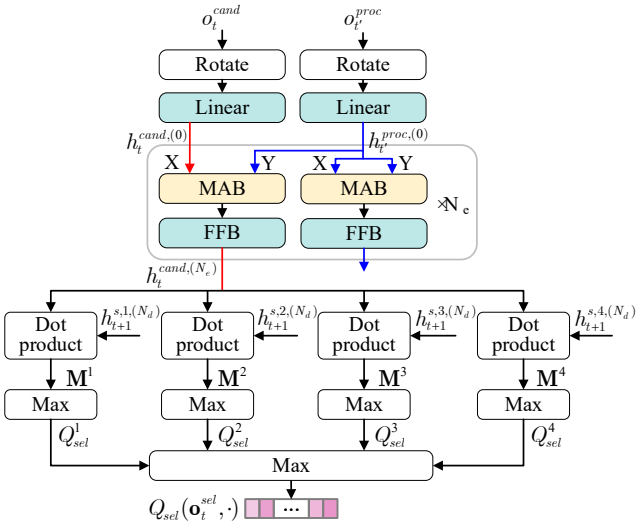


Figure 3: Dynamic candidate selection (DCS) framework.

$\mathbf{M}^k$ to derive action values $Q^k_{sel}(o^{sel,k}_t, \cdot) \in \mathbb{R}^{|\mathcal{U}_{cand}|}$, where $o^{sel,k}_t = \langle o^{cand}_t, o^{proc}_{t'}, o^{p,k}_{t+1}, o^{s,k}_{t+1} \rangle$. $Q^k_{sel}$ records the maximum action value that each item in $\mathcal{U}_{cand}$ can derive when moved to $\mathcal{U}_{proc}$ at $t+1$ time step evaluated by the PQNet $k$.

Finally, we take the maximum action value for each candidate item across the four PQNets, deriving

$$Q_{sel}(\mathbf{o}^{sel}_t, \cdot) = \{ \max_{k \in \{1,2,3,4\}} Q^k_{sel}(o^{sel,k}_t, a) | a \in \mathcal{A}^{sel} \}. \tag{19}$$

$Q_{sel}$ can approximate $Q^*_{sel}$ for selecting candidate items, as it directly reflects the maximum potential value that would result from moving the candidate item to $\mathcal{U}_{proc}$.

The overall structure of our method, which combines BBMP and DCS frameworks, is shown in Fig. 4. Each time step the packing process is divided into two steps: 1) using BBMP framework to guide the packing of an item in $\mathcal{U}_{proc}$, and 2) using DCS framework to guide the selection of an item in $\mathcal{U}_{cand}$ to move to $\mathcal{U}_{proc}$. The detailed training and execution procedures are provided in Appendix B.

# Experiments

## Experimental Settings

Following the common data generation rules (Jiang, Cao, and Zhang 2023; Que, Yang, and Zhang 2023; Yin, Chen, and He 2024), we randomly sample the length, width, and height of each item within ranges $[L/10, L/2]$, $[W/10, W/2]$, and $[\min(L/10, W/10), \max(L/2, W/2)]$, respectively. The container size $L \times W$ is set to $100 \times 100$, $200 \times 200$, and $400 \times 200$. The item number $N$ is set to $\{20, 30, 40, 50, 100, 120\}$ and further extended to $\{200, 500, 1000\}$ to evaluate scalability for large numbers of items. The detailed hyperparameter configurations and experimental environments are provided in Appendix C.

Several packing methods are used as baselines for comparison with our method: 1) Extreme points (EP) (Crainic, Perboli, and Tadei 2008); 2) Largest area first fit (LAFF)

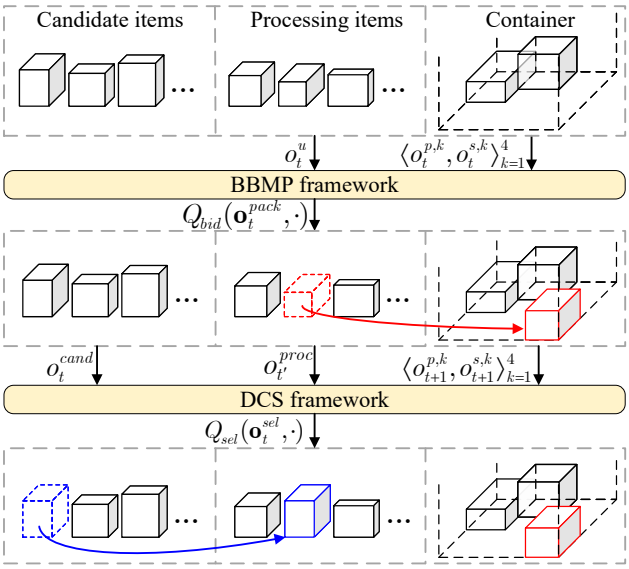


Figure 4: Overall structure of our method.

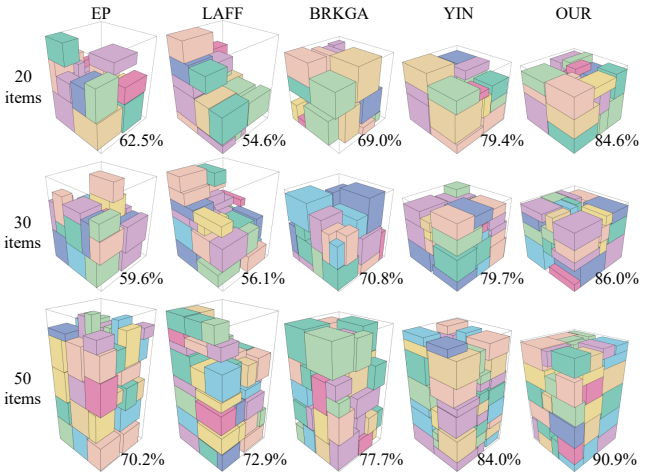


Figure 5: Visualization for packing results.

(Gurbuz et al. 2009); 3) Biased random key genetic algorithm (BRKGA) (Gonçalves and Resende 2013); 4) Multi-task selected learning (MTSL) (Duan et al. 2019); 5) Conditional query learning (CQL) (Li et al. 2020); 6) Recurrent conditional query learning (RCQL) (Li et al. 2022); 7) JIANG (Jiang, Cao, and Zhang 2023); 8) QUE (Que, Yang, and Zhang 2023); 9) YIN (Yin, Chen, and He 2024). In the above methods, EP and LAFF are constructive heuristics, BRKGA is a search algorithm, while MTSL, CQL, RCQL, JIANG, QUE, and YIN are DRL-based methods.

## Experimental Results

**Comparison** Table 1 presents the average space utilization and computation time (noted in parentheses) results of various methods on discrete instances with $20 \sim 1000$ items and the $100 \times 100$ container. The results of each method are averaged over 1024 uniformly sampled instances. The "-" symbol in the table indicates that the method requires excessive computation time or cannot be solved under general hardware conditions. Note that RCQL focuses more on instances with large numbers of items; therefore, its performance on instances with $N \leq 50$ is not reported. To ensure comparative fairness and validity, our method is evaluated through 10 independent trials with distinct random datasets sampled from the same distribution. The results (provided in Appendix D) exhibit low variance ($< 10^{-6}$) across all trials, and the mean value is adopted as the final outcome.

Table 1 illustrates that our method requires shorter computational time compared to most DRL-based methods. This primarily stems from our deterministic policy that always selects the optimal action for the current state, as opposed to policy-based methods, which sample multiple solutions and select the best outcome. Existing DRL-based methods and heuristic methods demonstrate space utilization advantages for $N \leq 120$ and $N > 120$, respectively. However, no single method delivers high-quality solutions across all item numbers. Heuristic methods fail to efficiently arrange packing space for small numbers of items (as visualized in Fig. 5), while most DRL-based methods cannot plan large numbers of items due to high computational complexity. Our method, in contrast, delivers high-quality solutions for arbitrary $N$, achieving 3.2%$\sim$13.1% higher space utilization compared to the best-performing method for each $N$. This is primarily attributed to the capabilities of PQNet and the BBMP framework to efficiently explore more valid solutions, as well as the capability of the DCS framework to effectively plan large numbers of items with minimal computation overhead. Fig. 5 visualizes the packing results of several methods for the same instance with 20, 30, and 50 items. It can be observed that our method fully uses the container space and plans a relatively flat top surface, thus reducing space wastage.

We further evaluate the space utilization of methods on instances with 50 items and larger-size containers. As recorded in Table 2, our method achieves state-of-the-art performance even for 200×200 and 400×200 containers, demonstrating its capability to handle tasks requiring more precise item sizes and more diverse item types.

**Generality** By employing the attention mechanism (Vaswani et al. 2017) for encoding state and the EMS heuristic (Parreño et al. 2008) for managing spaces, our method can handle arbitrary numbers of items and container sizes without modifying the network configurations. Fig. 6 illustrates the generalization test results of the model trained with 50 items and $100 \times 100$ container (denoted as $\text{OUR}^{50}_{100\times100}$) on instances with different item numbers and container sizes. When the number of items changes modestly ($N = 20$, 30, 100, or 120), $\text{OUR}^{50}_{100\times100}$ exhibits a minor decline in space utilization compared to models trained on corresponding instances. When $N \geq 500$, the space utilization of $\text{OUR}^{50}_{100\times100}$ declines markedly. This is because, for small numbers of items, the model tends to prioritize forming flat top surfaces at lower heights, while for large numbers, it may sacrifice early-stage regularity to en-

| $N$ | EP | LAFF | BRKGA | MTSL | CQL | RCQL | JIANG | QUE | YIN | OUR |
|---|---|---|---|---|---|---|---|---|---|---|
| 20 | 66.5(<1) | 54.4(<1) | 66.6(7.5) | 62.4(4.8) | 67.0(1.0) | - | 71.8(1.2) | 76.5(1.4) | 79.2(3.5) | **82.4**(0.6) |
| 30 | 70.3(<1) | 60.0(<1) | 70.5(15.5) | 60.1(10.2) | 69.3(1.2) | - | 75.5(1.5) | 79.3(2.1) | 81.5(5.9) | **85.3**(0.9) |
| 50 | 73.6(<1) | 66.4(<1) | 73.9(36.2) | 55.3(23.0) | 73.6(3.3) | - | 81.3(3.8) | 82.4(3.5) | 84.1(9.9) | **89.1**(1.5) |
| 100 | 76.3(<1) | 74.4(<1) | 77.1(106.3) | - | - | 71.0(1.6) | 84.4(12.4) | - | - | **92.8**(5.4) |
| 120 | 76.6(<1) | 76.3(<1) | 77.9(141.1) | - | - | 71.4(1.8) | 84.8(15.2) | - | - | **93.2**(6.7) |
| 200 | 77.8(2.5) | 81.0(<1) | - | - | - | 72.4(2.8) | - | - | - | **94.1**(11.6) |
| 500 | 79.0(38.0) | 86.9(<1) | - | - | - | 73.2(6.9) | - | - | - | **95.4**(31.9) |
| 1000 | 79.4(312.7) | 90.0(1.2) | - | - | - | 74.0(13.6) | - | - | - | **96.0**(66.3) |

Table 1: Average space utilization (%) and computation time (second) results across varying item numbers.

| Method | $100 \times 100$ | $200 \times 200$ | $400 \times 200$ |
|---|---|---|---|
| EP | 73.6 | 72.6 | 70.6 |
| LAFF | 66.4 | 64.7 | 67.1 |
| BRKGA | 73.9 | 72.9 | 72.1 |
| MTSL | 55.3 | 50.8 | 46.9 |
| CQL | 73.6 | 58.7 | 47.5 |
| JIANG | 81.3 | 75.2 | 70.5 |
| QUE | 82.4 | 80.5 | 76.7 |
| YIN | 84.1 | 80.8 | 77.1 |
| OUR | **89.1** | **88.8** | **87.0** |

Table 2: Average space utilization (%) results across varying container sizes.

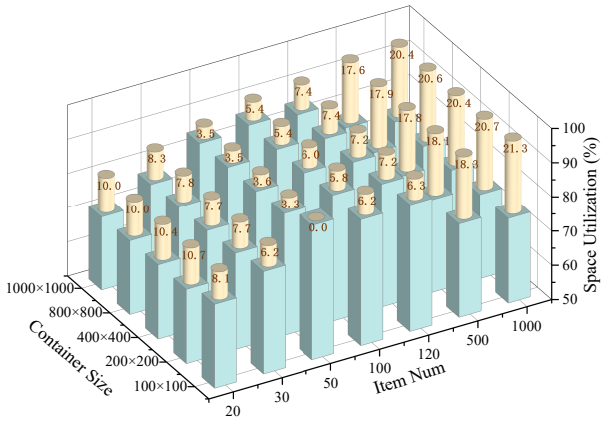


Figure 6: Generalization results. The square bars correspond to $\text{OUR}^{50}_{100\times100}$, while the cylindrical bars correspond to models trained on their corresponding instances. The numerical values are the difference between the two.

hance global packing efficiency. On the other hand, changes in container size have minimal impact on $\text{OUR}^{50}_{100\times100}$. This robustness stems from the model's capability to abstract spatial relationships proportionally rather than relying on absolute coordinates. Additional generalization experiments on item distributions are provided in Appendix D.

**Ablation** To validate the effectiveness of components in our method, we conduct ablation studies on instances with the $100 \times 100$ container, by removing or modifying each component while keeping other settings unchanged. As illustrated in Table 3, the absence of PQNet (-PQNet, i.e., using the same network structure as (Xu et al. 2023)) and BBMP framework (-BBMP, i.e., using only a single PQNet) both lead to declines in space utilization. This is because encoding packed items enables the network to understand occupied area within the container, offering essential insights for more precise value estimates. The BBMP framework, on the other hand, provides more valid placements compared to a single PQNet, enriching packing policy and exploring more valid solutions. Table 4 illustrates that the absence of DCS framework (-DCS, i.e., using candidate queue) also leads to a decline in space utilization, with the decline becoming more pronounced as $N$ increases. This trend can be explained as larger numbers of candidate items providing more selection possibilities for DCS framework, enabling it to adjust a more favorable $\mathcal{U}_{proc}$.

| Method | $N = 20$ | $N = 30$ | $N = 50$ |
|---|---|---|---|
| OUR | **82.4** | **85.3** | **89.1** |
| -PQNet | 81.4 | 84.8 | 88.4 |
| -BBMP | 81.0 | 84.7 | 88.4 |

Table 3: Ablation results on PQNet and BBMP.

| Method | $N = 100$ | $N = 200$ | $N = 500$ | $N = 1000$ |
|---|---|---|---|---|
| OUR | **92.8** | **94.1** | **95.4** | **96.0** |
| -DCS | 92.3 | 93.5 | 94.1 | 94.5 |

Table 4: Ablation results on DCS.

## Conclusion

This paper proposes a scalable DRL-based method to solve the offline 3D-PP of diverse scales. The proposed method demonstrates superior space utilization performance across various scales of 3D-PP tasks compared to existing methods, attributed to its three core components: PQNet, BBMP framework, and DCS framework. Furthermore, the solution exhibits robust generalization to varying container sizes and item numbers. Challenges remain in handling real-world dynamic online packing scenarios. Extending the framework to incorporate realistic constraints and adaptive reward mechanisms can further improve its applicability.

## Acknowledgments

This work was partly supported by the National Natural Science Foundation of China (NSFC) under Grant U1936113 and Grant 61872303; and in part by Doctoral Innovation Fund Program of Southwest Jiaotong University under Grant CX2025YB10.

## References

Ali, S.; Ramos, A. G.; Carravilla, M. A.; and Oliveira, J. F. 2022. On-line three-dimensional packing problems: A review of off-line and on-line solution approaches. *Computers & Industrial Engineering*, 168: 108122.

Bortfeldt, A.; Gehring, H.; and Mack, D. 2003. A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing*, 29(5): 641–662.

Bortfeldt, A.; and Wäscher, G. 2013. Constraints in container loading - A state-of-the-art review. *European Journal of Operational Research*, 229(1): 1–20.

Chen, C.; Lee, S.; and Shen, Q. 1995. An analytical model for the container loading problem. *European Journal of Operational Research*, 80(1): 68–76.

Chen, Y.; Yang, K.; Tao, J.; and Lyu, J. 2025. Novelty-Guided Data Reuse for Efficient and Diversified Multi-Agent Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 15930–15938. Philadelphia, USA.

Chung, F. R.; Garey, M. R.; and Johnson, D. S. 1982. On packing two-dimensional bins. *SIAM Journal of Algebraic Discrete Methods*, 3(1): 66–76.

Crainic, T. G.; Perboli, G.; and Tadei, R. 2008. Extreme Point-Based Heuristics for Three-Dimensional Bin Packing. *INFORMS Journal on Computing*, 20(3): 368–384.

Duan, L.; Hu, H.; Qian, Y.; Gong, Y.; Zhang, X.; Xu, Y.; and Wei, J. 2019. A Multi-task Selected Learning Approach for Solving 3D Flexible Bin Packing Problem. arXiv:1804.06896.

George, J. A.; and Robinson, D. F. 1980. A heuristic for packing boxes into a container. *Computers & Operations Research*, 7(3): 147–156.

Gonçalves, J. F.; and Resende, M. G. 2013. A biased random key genetic algorithm for 2D and 3D bin packing problems. *International Journal of Production Economics*, 145(2): 500–510.

Gurbuz, M. Z.; Akyokus, S.; Emiroglu, I.; and Guran, A. 2009. An Efficient Algorithm for 3D Rectangular Box Packing. In *Applied Automatic Systems: Proceedings of Selected AAS 2009 Paper*, 131–134.

Gálvez, W.; Grandoni, F.; Ingala, S.; Heydrich, S.; Khan, A.; and Wiese, A. 2021. Approximating Geometric Knapsack via L-packings. *ACM Transactions on Algorithms*, 17(4): 1–67.

Heßler, K.; Hintsch, T.; and Wienkamp, L. 2025. A fast optimization approach for a complex real-life 3D multiple bin Size Bin Packing Problem. *European Journal of Operational Research*.

Hu, H.; Zhang, X.; Yan, X.; Wang, L.; and Xu, Y. 2017. Solving a new 3D bin packing problem with deep reinforcement learning method. arXiv:1708.05930.

Jiang, Y.; Cao, Z.; and Zhang, J. 2023. Learning to Solve 3-D Bin Packing Problem via Deep Reinforcement Learning and Constraint Programming. *IEEE Transactions on Cybernetics*, 53(5): 2864–2875.

Li, D.; Gu, Z.; Wang, Y.; Ren, C.; and Lau, F. C. 2022. One model packs thousands of items with Recurrent Conditional Query Learning. *Knowledge-Based Systems*, 235: 107683.

Li, D.; Ren, C.; Gu, Z.; Wang, Y.; and Lau, F. C. 2020. Solving Packing Problems by Conditional Query Learning. https://openreview.net/forum?id=BkgTwRNtPB. Accessed: 2019-09-26.

Li, M.; Wang, Q.; and Xu, Y. 2025. Gtde: Grouped training with decentralized execution for multi-agent actor-critic. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 18368–18376. Philadelphia, USA.

Lin, B.; Li, J.; Cui, T.; Jin, H.; Bai, R.; Qu, R.; and Garibaldi, J. 2024. A pattern-based algorithm with fuzzy logic bin selector for online bin packing problem. *Expert Systems with Applications*, 249: 123515.

Liu, J.; Yue, Y.; Dong, Z.; Maple, C.; and Keech, M. 2011. A novel hybrid tabu search approach to container loading. *Computers & Operations Research*, 38(4): 797–807.

Liu, K.; Zhang, H.; Zhang, Y.; and Sun, C. 2023. False Data-Injection Attack Detection in Cyber–Physical Systems With Unknown Parameters: A Deep Reinforcement Learning Approach. *IEEE Transactions on Cybernetics*, 53(11): 7115–7125.

Liu, S.; Shang, X.; Cheng, C.; Zhao, H.; Shen, D.; and Wang, F. 2017. Heuristic algorithm for the container loading problem with multiple constraints. *Computers & Industrial Engineering*, 108: 149–164.

Martello, S.; Pisinger, D.; and Vigo, D. 2000. The three-dimensional bin packing problem. *Operations Research*, 48(2): 256–267.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Pan, Y.; Chen, Y.; and Lin, F. 2023. Adjustable robust reinforcement learning for online 3d bin packing. *Advances in Neural Information Processing Systems*, 36: 51926–51954.

Parreño, F.; Alvarez-Valdés, R.; Tamarit, J. M.; and Oliveira, J. F. 2008. A maximal-space algorithm for the container loading problem. *INFORMS Journal on Computing*, 20(3): 412–422.

Que, Q.; Yang, F.; and Zhang, D. 2023. Solving 3D packing problem using Transformer network and reinforcement learning. *Expert Systems with Applications*, 214: 119153.

Ren, J.; Tian, Y.; and Sawaragi, T. 2011. A tree search method for the container loading problem with shipment priority. *Computers & Industrial Engineering*, 214(3): 526–535.

Saraiva, R. D.; Nepomuceno, N.; and Pinheiro, P. R. 2015. A layer-building algorithm for the three-dimensional multiple bin packing problem: a case study in an automotive company. *IFAC-PapersOnLine*, 48(3): 490–495.

Shao, X.; and Xiao, L. 2025. Research on Packing Problem Based on Improved Ant Colony Algorithm. In *International Conference on Sensors and Information Technology*, 361–364. Nanjing, China.

Toffolo, T. A.; Esprit, E.; Wauters, T.; and Berghe, G. V. 2017. A two-dimensional heuristic decomposition approach to a three-dimensional multiple container loading problem. *European Journal of Operational Research*, 257(2): 526–538.

Tsai, J.; Wang, P.; and Lin, M. 2015. A global optimization approach for solving three-dimensional open dimension rectangular packing problems. *Journal of Optimization*, 64(12): 2601–2618.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. *Advances in Neural Information Processing Systems*, 30.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer Networks. *Advances in Neural Information Processing Systems*, 28: 2692–2700.

Wang, B.; Lin, Z.; Kong, W.; and Dong, H. 2025. Bin packing optimization via deep reinforcement learning. *IEEE Robotics and Automation Letters*, 10(3): 2542–2549.

Xiong, H.; Guo, C.; Peng, J.; Ding, K.; Chen, W.; Qiu, X.; Bai, L.; and Xu, J. 2024. GOPT: Generalizable online 3D bin packing via transformer-based deep reinforcement learning. *IEEE Robotics and Automation Letters*, 9(11): 10335–10342.

Xu, J.; Gong, M.; Zhang, H.; Huang, H.; and Hu, R. 2023. Neural Packing: from Visual Sensing to Reinforcement Learning. *ACM Transactions on Graphics*, 42(6).

Yang, S.; Song, S.; Chu, S.; Song, R.; Cheng, J.; Li, Y.; and Zhang, W. 2023. Heuristics Integrated Deep Reinforcement Learning for Online 3D Bin Packing. *IEEE Transactions on Automation Science and Engineering*, 1(1): 1–12.

Yang, Y.; Wu, Z.; Hao, X.; Liu, H.; and Qi, M. 2024. Two-layer heuristic for the three-dimensional bin design and packing problem. *Engineering Optimization*, 56(10): 1601–1638.

Yao, S.; Liu, F.; Lin, X.; Lu, Z.; Wang, Z.; and Zhang, Q. 2025. Multi-objective evolution of heuristic using large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 27144–27152. Philadelphia, USA.

Yin, H.; Chen, F.; and He, H. 2024. Solving Offline 3D Bin Packing Problem with Large-sized Bin via Two-stage Deep Reinforcement Learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, 2576–2578. Auckland, New Zealand.

Zhao, H.; She, Q.; Zhu, C.; Yang, Y.; and Xu, K. 2021. Online 3D Bin Packing with Constrained Deep Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 741–749. Virtual, Online.