Knowledge-Based Systems 299 (2024) 111990

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A Dynamic Multi-modal deep Reinforcement Learning framework for 3D Bin Packing Problem

Anhao Zhao *, Tianrui Li, Liangcai Lin

School of Computing and Artificial Intelligence, Southwest jiaotong university, No. 999 , Xi'an Road, Pidu District, Chengdu, 610031, Sichuan, China

ARTICLE INFO

Keywords:
Deep reinforcement learning
Combinatorial optimization
3D bin packing problem

ABSTRACT

The 3D bin packing problem, a notorious NP-hard combinatorial optimization challenge with wide-ranging practical implications, focuses on optimizing spatial allocation within a bin through the arrangement of boxes. Previous studies have shown promise in employing Neural Combinatorial Optimization to tackle such intractable problems. However, due to the inherent diversity in observations and the sparse rewards, current learning-based methods for addressing the 3D bin packing problem have yielded less-than-ideal outcomes. In response to this shortfall, we propose a novel Dynamic Multi-modal deep Reinforcement Learning framework tailored specifically for the 3D Bin Packing Problem, coined as DMRL-BPP. This framework stands apart from existing learning-based bin-packing approaches in two pivotal respects. Firstly, in order to capture the range of observations, we introduce an innovative dynamic multi-modal encoder, comprising dynamic boxes state and gradient height map sub-encoders, effectively modeling multi-modal information. Secondly, to mitigate the challenge of sparse rewards, we put forth a novel reward function, offering an efficient and adaptable approach for addressing the 3D bin packing problem. Extensive experimental results validate the superiority of our method over all baseline approaches, across instances of varying scales.

1. Introduction

Numerous practical challenges, such as urban freight transportation [1] and the design of intermodal networks [2], have the potential to yield substantial economic benefits through the optimization of cargo placement in three-dimensional space. In this study, we systematically categorize these real-world scenarios as instances of the 3D Bin Packing Problem (3D-BPP). When presented with a collection of cuboid-shaped items, the goal of 3D-BPP is to either densely pack a single bin or use the fewest possible bins to accommodate all items. In this paper, we focus on the former objective.

3D-BPP stands as a fundamental conundrum in combinatorial optimization, acknowledged for its NP-hard nature [3]. Over the past few decades, it has garnered substantial attention and dedicated research endeavors. Broadly, traditional approaches fall into two distinct categories: exact algorithms and heuristic algorithms. While exact algorithms can ascertain the optimal solution for 3D-BPP, they may struggle to do so within reasonable time constraints. Conversely, heuristic algorithms rely on intuition or experiential knowledge to furnish a feasible solution with an acceptable cost. Nevertheless, both exact and heuristic methodologies necessitate the formulation of extensive explicit or hand-crafted rules tailored to each specific problem configuration. In recent years, there has been a surge of interest in employing Deep Reinforcement Learning (DRL) to address classical combinatorial optimization challenges, including prominent instances like the Traveling Salesman Problem (TSP) [4,5] and the Vehicle Routing Problem (VRP) [6]. These developments underscore the substantial potential of leveraging DRL in resolving combinatorial optimization dilemmas. Notably, pioneering efforts have been made in applying DRL to tackle the 3D-BPP. For instance, CQL [7] introduced the inaugural end-to-end learning algorithm tailored to the packing problem, while Attend2Pack [8] employs DRL to address bin packing and incorporates an innovative prioritized oversampling strategy to accelerate on-policy learning. However, they did not simultaneously consider the challenges of effective modeling of multi-modal information and sparse rewards, which limited their ability to produce sufficiently compact packing results.

This paper introduces a novel approach, the Dynamic Multi-modal deep Reinforcement Learning framework, tailored for solving the 3D Bin Packing Problem. This framework, referred to as DMRL-BPP, aims to learn a policy that yields compact and efficient packing outcomes. When tackling the 3D-BPP, the process of fitting a box into a bin unfolds in three distinct phases: selection, rotation, and placement. To grapple with the expansive combinatorial action space inherent in this task, we employ DRL for the initial two steps, while employing a heuristically

* Corresponding author.
E-mail addresses: zhaoanh@my.swjtu.edu.cn (A. Zhao), trli@swjtu.edu.cn (T. Li), alim@home.swjtu.edu.cn (L. Lin).

https://doi.org/10.1016/j.knosys.2024.111990
Received 26 December 2023; Received in revised form 23 April 2024; Accepted 22 May 2024
Available online 24 May 2024

devised strategy for the placement step. Our methodology begins with the adoption of a dynamic multi-modal encoder, which embeds the dynamic state of the boxes and the gradient height map. Subsequently, we utilize the selection decoder and rotation decoder to generate the selection policy and the rotation policy, respectively. Furthermore, we incorporate a heuristically crafted strategy to determine the placement policy. A key departure from prior methods lies in our curated multi-modal encoder, which adeptly leverages both spatial information and the states of all boxes. To tackle the challenge of sparse rewards, we introduce a novel reward function that offers an efficient and adaptable approach for addressing 3D-BPP. For updating model parameters, we employ the Soft Actor–Critic algorithm [9] due to its superior exploratory performance. This combination of innovative techniques constitutes a promising solution for 3D-BPP.

To sum up, the contributions of this paper are as follows:

- We propose a Dynamic Multi-modal deep Reinforcement Learning framework for the 3D-BPP (DMRL-BPP). Compared to existing learning-based bin-packing approaches, our method results in lower gap ratios across various instances.
- We propose a novel dynamic multi-modal encoder, which comprises a dynamic boxes state sub-encoder and a gradient height map sub-encoder, to model multi-modal information more effectively.
- We introduce a novel reward function designed to address the issue of sparse rewards, providing an efficient and adaptable solution to the 3D bin packing problem.
- We achieve improvements of 7.8%, 2.2%, 1.4% in the gap ratios over the state-of-the-art 3D-BPP approach based on DRL in 16 boxes, 20 boxes, and 30 boxes, respectively. Numerical results show that our method outperforms both heuristic algorithms and the state-of-the-art DRL-based method.

2. Related works

Deep Reinforcement Learning for Combinatorial Optimization. Combinatorial optimization is a long-studied optimization problem that aims to find an optimal object from a finite set of objects. Several recent works have begun to incorporate DRL into the combinatorial optimization problem where the strategy can be well parameterized with deep neural networks. [5] proposed to learn the conditional probability of an output sequence based on the positional elements from the input sequence to solve the Traveling Salesman Problem (TSP). [4] developed a general deep reinforcement learning framework to tackle combinatorial optimization problems, such as TSP and Knapsack problems. [10] proposed an attentive model over the Pointer Network, which makes significant improvement against recent heuristics for TSP. In this work, as a typical combinatorial optimization problem, the studied 3D-BPP also enjoys the merits of deep reinforcement learning.

3D Bin Packing Problem. The 3D-BPP, an intricate combinatorial optimization challenge, has been a focal point of considerable research attention in recent times. Building upon the foundation of the 2D bin packing, the 3D-BPP, classified as an NP-hard problem [3], has been subjected to a multitude of investigative efforts aimed at devising effective resolution strategies. These strategies predominantly bifurcate into traditional methods and learning-based methods. The traditional methods can be divided into exact and heuristic algorithms. [11] considered the problem of loading containers with non-uniform-sized cartons and developed a 0–1 mixed integer programming model to obtain the optimal solution. [12] tried to obtain a nearly optimal solution within polynomial time. With the advance of deep learning, several recent works have turned to deep reinforcement learning to address the 3D-BPP by formulating it as a decision-making problem. [13] applied the pointer net framework to optimize the sequence of items to be packed. [7] proposed the conditional query learning (CQL) model to solve packing problems. [8] proposed a new end-to-end learning model for this task of interest and achieved state-of-the-art performance in a range of experimental settings.

3. Formulating packing problem as Markov decision process

3.1. Problem definition

The 3D-BPP problem is a real-world-driven combinatorial optimization problem that aims to solve the optimization problem of the spatial arrangement of boxes within a bin to maximize space utilization. Given a set of cuboid-shaped items, the objective of 3D-BPP is to either pack a single bin as compactly as possible or pack all items using as few bins as possible. In this paper, we use the first objective. The width and length of the bin are fixed and limited, and the height is infinite.

The problem involves a set of N boxes $\{(l_i, w_i, h_i)\}_{i=1}^{N}$ and a bin whose width, length, and height are W, L, and $H = \sum_{i=1}^{N} max(l_i, w_i, h_i)$, where l_i, w_i, h_i represent the length, width, and height of the ith box respectively. When placing a box into the bin, We select a box from all available boxes, so the maximum size of the select action space is N. When the box is selected, the box can be rotated 90° along the x, y, and z-axis, which results in a rotation action space of size O = 6. When the direction of rotation is selected, the box has a maximum of $L \times W \times H$ optional positions in the bin, so the maximum size of the placement action space is $L \times W \times H$ (we set the coordinate of the bottom-left-front corner of the bin to (0, 0, 0), the position of the box is represented by the coordinate (x_i, y_i, z_i) of its bottom-left-front corner relative to the bottom-left-front corner of the bin). Then for each box packing step, the total action space size is $N \times O \times L \times W \times H$. A solution to this problem involves a sequence of boxes, all strategically placed within the bin. This placement must satisfy geometric constraints, ensuring that there is no overlap between items. The goal is to find a solution that maximizes the space utilization of the bin. Space utilization is defined as $\frac{\sum_{i=1}^{N} w_i l_i h_i}{W L H_N}$, where H_N denotes the height of the smallest bounding box containing all N packed boxes within the bin.

3.2. Markov decision process

Since the 3D-BPP is NP-hard, it is not feasible to obtain an exact solution within an acceptable time limit, so a reasonable opinion is to use reinforcement learning to make the agent learn from experience. MDP is a mathematically idealized form of reinforcement learning problem. MDP is a classical formalization of sequential decision-making, where actions influence not just immediate rewards but also subsequent situations or states. Thus, MDP involves delayed rewards and the need to trade off immediate and delayed rewards. In MDP, the state must have the Markov property. That is, it includes all aspects of the interaction between the agent and the environment in the past, and this information will have a certain impact on the future. When packing a box into the bin, the process can be divided into three steps: 1. Selecting a box from all boxes not yet packed into the bin; 2. Selecting the rotation for the selected box; 3. Selecting the coordinate in the bin for the rotated box to place it.

These three steps are executed strictly in order. When executing the selecting step, the agent needs to know which boxes have been packed into the bin, the arrangement of the packed boxes, and which boxes have not been packed into the bin. When executing the rotating step, the agent needs to know which box is selected. When executing the placement step, the agent needs to know which box is selected and the rotation of the box.

4. Methods

Here we start with introducing the dynamic boxes state sub-encoder and gradient height map sub-encoder in Section 4.1. The Selection Decoder and Rotation Decoder are then discussed in Sections 4.2 and 4.3. Once the box index and box rotation are determined, the box can be placed in the bin using different packing placement methods, which are explained in Section 4.4. How to train the model is presented in Section 4.5.

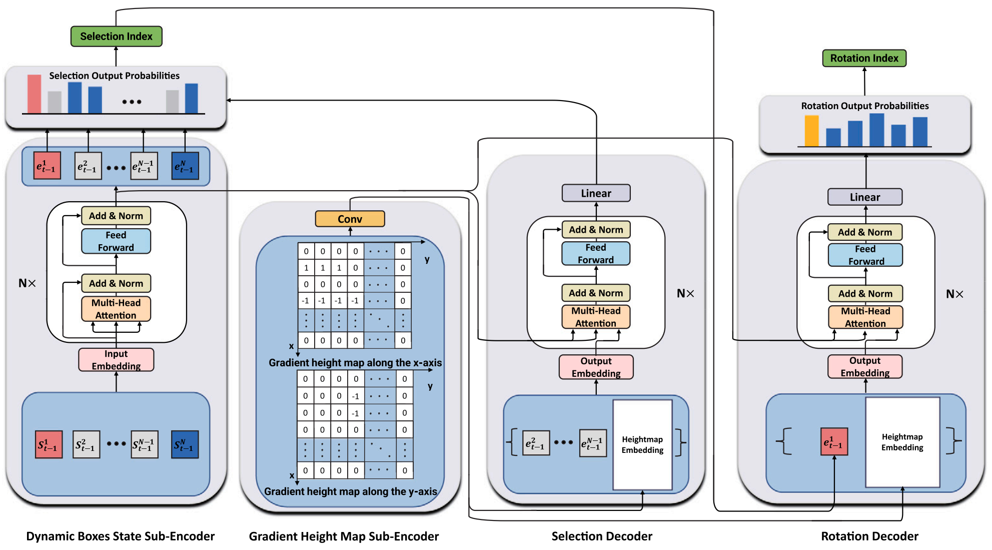

Selection Index
Selection Output Probabilities
Add & Norm
Feed Forward
Multi-Head Attention
Input Embedding
N×
Conv
Gradient height map along the x-axis
Gradient height map along the y-axis
Linear
Output Embedding
Heightmap Embedding
Rotation Index
Rotation Output Probabilities
Dynamic Boxes State Sub-Encoder
Gradient Height Map Sub-Encoder
Selection Decoder
Rotation Decoder

Fig. 1. The architecture of DMRL-BPP consists of four modules, including two sub-encoders, a selection decoder for choosing a box that has not yet been placed into the bin, and a rotation decoder to determine the rotational orientation of the selected box.

4.1. Dynamic multi-modal encoder

To adeptly leverage both the spatial information and dynamic states of all boxes, we have developed a novel dynamic multi-modal encoder. This encoder integrates a sub-encoder for dynamically tracking the boxes' states and another sub-encoder for the gradient height map, ensuring both inputs are responsive to changes.

4.1.1. Dynamic boxes state sub-encoder

The input for the dynamic boxes state sub-encoder describes the dynamic basic information of all boxes. Inspired from [7], at time step t, the input is a set $B_t = \{s_t^1, s_t^2, \dots, s_t^N\}$, where $s_t^i = (p_t^i, l_t^i, w_t^i, h_t^i, x_t^i, y_t^i, z_t^i)$, p_t^i is a boolean variable indicating whether the ith box is packed into the bin at time step t, (l_t^i, w_t^i, h_t^i) is the length, width and height of the ith box after rotation at time step t, (x_t^i, y_t^i, z_t^i) is the left-front-bottom coordinate of the ith box in the bin at time step t. If the box is not packed, we mask the box position information in the input.

As shown in Fig. 1, inspired by the strong contextual modeling capabilities, we employ Transformer layers, as introduced by Vaswani et al. [14], to encode the input. Unlike the approach by Vaswani et al. [14], the order of boxes in our input does not impact the problem. Therefore, we omit the positional encoding described in their work. At each packing step, the input set B_t can be stacked into an input matrix $\boldsymbol{X}_t$, $\boldsymbol{X}_t$ is first embedded through a linear layer

$$\tilde{\boldsymbol{X}}_t = Linear(\boldsymbol{X}_t). \tag{1}$$

Input embedding is then passed through several multi-head self-attention layers [14]. each layer performs the following operations

$$\hat{\boldsymbol{X}}_t = LayerNorm(\tilde{\boldsymbol{X}}_t + MultiHeadAttention(\tilde{\boldsymbol{X}}_t)), \tag{2}$$

$$\bar{\boldsymbol{X}}_t = LayerNorm(\hat{\boldsymbol{X}}_t + FeedForward(\hat{\boldsymbol{X}}_t)). \tag{3}$$

Where $\tilde{\boldsymbol{X}}_t$ and $\hat{\boldsymbol{X}}_t$ denote the input of Multi-Head Attention layer [14] and Feed Forward layer respectively. $\tilde{\boldsymbol{X}}_t + MultiHeadAttention(\tilde{\boldsymbol{X}}_t)$ and $\hat{\boldsymbol{X}}_t + FeedForward(\hat{\boldsymbol{X}}_t)$ denote residual connections [15]. $LayerNorm$ denotes Layer Normalization [16]. In this way, the embedding matrix $\bar{\boldsymbol{X}}_t$ is obtained, and the set of box embeddings $E_t = \{\bar{\boldsymbol{X}}_t^1, \bar{\boldsymbol{X}}_t^2, \dots, \bar{\boldsymbol{X}}_t^N\}$ is obtained by decomposing it, the dimensionality of each box embedding in the set is h. Unlike machine translation problem that has fixed input features during inference, in the 3D-BPP, box embeddings need to be re-encoded at each time step during an episode.

4.1.2. Gradient height map sub-encoder

The gradient height map sub-encoder input describes the situation inside the bin. Different from [8,17], which directly use the top-down view of the current packing layout as input, inspired from [18], we extend the original 2D gradient height map to 3D gradient height map. Our problem is set in a 3D scene, however, for brevity, we will not repeatedly specify '3D' throughout our discussion. A gradient height map uses the height differences between adjacent columns. Since the bottom of the bin is 2D, the height differences between adjacent columns can be along the x-axis or the y-axis. We calculate the height differences between adjacent columns separately in both directions, so the gradient height map sub-encoder input is a 2D image with two channels. As shown in Fig. 1, for a bin of width $W = 10$ and length $L = 10$, its gradient height map is of size $2 \times 10 \times 10$. The value of grid cell (x, y) in the gradient height map along the x-axis refers to the total height of stacked boxes in grid cell (x, y) minus the total height of stacked boxes in grid cell $(x-1, y)$. The value of grid cell (x, y) in the gradient height map along the y-axis refers to the total height of stacked boxes in grid cell (x, y) minus the total height of stacked boxes in grid cell $(x, y-1)$. The example in Fig. 1 is a gradient height map formed by placing a box of size $2 \times 3 \times 1$ at $x = 1, y = 0$. At each time step t, a gradient height map H_t is passed through several convolutional layers to obtain the gradient height map embedding $\boldsymbol{h}_t$, the dimensionality of $\boldsymbol{h}_t$ is h. The gradient height map sub-encoder is designed to embed the gradient height map for more informative auxiliary representations. Intuitively, it is easier to observe the possible Corners of Empty-Maximal-Space [19] using a gradient height map than directly using the top view of the current package layout.

4.2. Selection decoder

In the box selection step, at time step t, we first feed the dynamic input to the encoder and get a set of box embeddings $E_t = \{\bar{\boldsymbol{X}}_t^1, \bar{\boldsymbol{X}}_t^2, \dots, \bar{\boldsymbol{X}}_t^N\}$ and a gradient height map embedding $\boldsymbol{h}_t$ as described before. As shown in Fig. 1, in order to be able to handle variable length input, We build on the pointer network [5] to model the selection policy π_t^s. The selection decoder input should be able to describe the situation in the bin as much as possible. Gradient height map directly gives the stacking situation of the boxes in the bin, which may not be well captured by the boxes state alone, so the selection decoder input should contain the embeddings of the selected boxes until time step t

Table 1
The correspondence between the output index of the rotation decoder and the rotation of the box.

Rotation	0	1	2	3	4	5
l', w', h'	l,w,h	l,h,w	w,l,h	w,h,l	h,l,w	h,w,l

and a gradient height map embedding $\boldsymbol{h}_t$ at time step t. We define the selection decoder input as

$$\boldsymbol{q}_t^s = Mean(Mean(E_t^{selected}), \boldsymbol{h}_t). \tag{4}$$

Where $E_t^{selected}$ denotes the set of selected box embeddings. $Mean$ denotes an operation that takes in a set of vectors of the same dimension and returns their mean vector, which has the same dimension as each input vector.

Then we feed $\boldsymbol{q}_t^s$ to the selection decoder. Inspired from [4], we use the (M-head) attention mechanism to compute glimpse query vector $\tilde{\boldsymbol{q}}_t^s$, then we use it to calculate selection policy probabilities. For zth head, the keys and values come from the set of box embeddings $E_t = \{\bar{\boldsymbol{X}}_t^1, \bar{\boldsymbol{X}}_t^2, \dots, \bar{\boldsymbol{X}}_t^N\}$ at time step t, but we only compute a single query $\boldsymbol{q}_t^z$ from $\boldsymbol{q}_t^s$:

$$\boldsymbol{q}_t^z = \boldsymbol{W}^{q,z}\boldsymbol{q}_t^s, \quad \boldsymbol{k}_{t,i}^z = \boldsymbol{W}^{k,z}\bar{\boldsymbol{X}}_t^i, \quad \boldsymbol{v}_{t,i}^z = \boldsymbol{W}^{v,z}\bar{\boldsymbol{X}}_t^i. \tag{5}$$

Where $\boldsymbol{W}^{q,z}, \boldsymbol{W}^{k,z}, \boldsymbol{W}^{v,z} \in \mathbb{R}^{h\times d_k}$ are trainable parameters, here $d_k = \frac{h}{M}$ is the query/key dimensionality. Then We compute the compatibility of the query with all box embeddings at time step t:

$$u_{t,j}^z = \frac{\boldsymbol{q}_t^{z^T}\boldsymbol{k}_{t,j}^z}{\sqrt{d_k}}. \tag{6}$$

From the compatibilities $u_{t,j}^z$, we use a softmax to compute the attention weights $p_{t,j}^z \in [0, 1]$:

$$p_{t,j}^z = \frac{e^{u_{t,j}^z}}{\sum_{j'} e^{u_{t,j'}^z}}. \tag{7}$$

Finally, we perform a weighted sum of the glimpse value vectors to obtain the glimpse query vector for the zth head:

$$\boldsymbol{q}_t^{\tilde{s},z} = \sum_j p_{t,j}^z \boldsymbol{v}_{t,j}^z \tag{8}$$

Concatenating the glimpse query vectors from all M = 8 heads and we obtain the final glimpse query vector $\tilde{\boldsymbol{q}}_t^s$. As shown in Fig. 1, in order to be able to calculate the selection strategy probabilities, We add a final decoder layer that has only one attention head (M = 1). We compute the updated compatibility, and mask (set $u_{\tilde{t},j} = -\infty$) the boxes that have been packed into the bin($p_t^j = True$):

$$u_{\tilde{t},j} = \begin{cases} C \cdot tanh(\frac{(\boldsymbol{W}^Q\tilde{\boldsymbol{q}}_t^s)^T(\boldsymbol{W}^K\bar{\boldsymbol{X}}_t^j)}{\sqrt{h}}) & if \quad p_t^j = Flase, \\ -\infty & otherwise. \end{cases} \tag{9}$$

Where $\boldsymbol{W}^Q, \boldsymbol{W}^K \in \mathbb{R}^{h\times h}$ are trainable parameters. The result is clipped within $[-C, C]$ $(C = 10)$ using tanh following [4]. Finally, we calculate the selection strategy probabilities using a softmax:

$$p_{\tilde{t},j} = p_{\theta^s}(\pi_t^s = j|\boldsymbol{q}_t^s) = \frac{e^{u_{\tilde{t},j}}}{\sum_{j'} e^{u_{\tilde{t},j'}}}. \tag{10}$$

During training, we sample from the distribution $\pi_t^s \sim p_{\theta^s}(\cdot|\boldsymbol{q}_t^s)$ to select box index π_t^s at time step t. While during evaluation, we select the box index π_t^s with the maximum probability $\pi_t^s = argmax_{\pi_t^s} p_{\theta^s}(\cdot|\boldsymbol{q}_t^s)$ at time step t.

4.3. Rotation decoder

When the box index π_t^s is selected by the selection decoder, the corresponding box can undergo a 90° rotation about the x-axis, y-axis, and z-axis independently, resulting in six possible orientations as represented by the rotation decoder's output of size 6. The correspondence between the output index and the rotation of the box is shown in Table 1. The rotation decoder input should be able to describe which box is selected by the selection decoder and the situation in the bin as much as possible, so the rotation decoder input should contain the embedding of the box selected by the selection decoder at time step t and a gradient height map embedding $\boldsymbol{h}_t$ at time step t. We define the rotation decoder input as

$$\boldsymbol{q}_t^r = Mean(\bar{\boldsymbol{X}}_t^{\pi_t^s}, \boldsymbol{h}_t). \tag{11}$$

Then we feed $\boldsymbol{q}_t^r$ to the rotation decoder. We use the same (M-head) attention mechanism as the selection decoder to compute glimpse query vector $\tilde{\boldsymbol{q}}_t^r$. Then we compute the logit vector $\boldsymbol{L}_t^r$:

$$\boldsymbol{L}_t^r = C \cdot tanh(\boldsymbol{W}^L\tilde{\boldsymbol{q}}_t^r) \tag{12}$$

Where $\boldsymbol{W}^L \in \mathbb{R}^{h\times h}$ is trainable parameter. Finally, we calculate the rotation strategy probabilities using a softmax:

$$p_{\theta^r}(\cdot|\boldsymbol{q}_t^r) = softmax(\boldsymbol{L}_t^r) \tag{13}$$

As in the box selection step, we sample from the distribution $\pi_t^r \sim p_{\theta^r}(\cdot|\boldsymbol{q}_t^r)$ to select rotation π_t^r at time step t. While during evaluation, we select rotation π_t^r greedily.

4.4. Packing placement methods

As described in Section 3.1, for each box packing step, the total action space size is $N\times O\times L\times W\times H$. For 20 boxes case, that is, $N = 20$, if we follow the same setting as [17], in which $W \times H = 100 \times 100$ for the bin, the sides of boxes are sampled from integers in the range of [20, 80], so $H = \sum_{i=1}^{N} max(l_i, w_i, h_i) = \sum_1^{20} 80 = 80 \times 20 = 1600$, the total action space size is $N \times O \times L \times W \times H = 20 \times 6 \times 100 \times 100 \times 1600 = 1,920,000,000$. Although there have been works on large discrete action spaces [20,21], it is unrealistic to handle billion-scale discrete action spaces, so it is not quite suitable to use learning-based methods for the placement step. In order to reduce the size of the total action space, we employ a heuristically designed strategy for the placement step following [22]; [13]; [18]. We build on the strategy denoted as LB [18] to model the placement policy π_t^p. LB is an Empty-Maximal-Spaces-based (EMS-based) packing placement strategy [19]. It tries the left-front-bottom corner of each EMS and chooses the one that minimizes the height of the smallest bounding box containing all packed boxes.

4.5. Training

We utilize the Soft Actor-Critic (SAC) algorithm, as proposed by Haarnoja et al. [9], to train our model because of its outstanding exploratory performance. The previously introduced model functions as the actor model within this framework.

4.5.1. Critic network and advantage estimation

The input state of the critic model should have the Markov property. Similar to the selection decoder and rotation decoder, we adopt the (M-head) attention mechanism to model the critic network. The input of the critic is a gradient height map embedding $\boldsymbol{h}_t$ at time step t. The keys and values come from the set of box embeddings $E_t = \{\bar{\boldsymbol{X}}_t^1, \bar{\boldsymbol{X}}_t^2, \dots, \bar{\boldsymbol{X}}_t^N\}$ at time step t. The query comes from $\boldsymbol{h}_t$. After obtaining the final glimpse query vector, we pass it through several linear layers. Finally, we get a scalar output that denotes the value of the input state at time step t. We use the Generalized Advantage Estimation (GAE) [23], which balances the variance and the bias to accurately estimate the advantage:

$$\hat{A}_t^{\gamma,\lambda} = \delta_t + (\gamma\lambda)\delta_{t+1} + \dots + (\gamma\lambda)^{T-t-1}\delta_{T-1}. \tag{14}$$

Where $\delta_t = r_t + \gamma V_{\theta_c}(s_{t+1}) - V_{\theta_c}(s_t)$, t denotes the time index in [0, T], T denotes the size of trajectory segment. $V_{\theta_c}(s_t)$ and $V_{\theta_c}(s_{t+1})$ denote the value functions at time step t and time step t+1 respectively. r_t denotes the reward at time step t. γ and λ are hyper-parameters that contribute to the bias–variance tradeoff.

4.5.2. *Reward function*

How to design the reward function is an extremely important part of the application of deep reinforcement learning. The quality of the reward function determines whether the agent can learn a good policy and directly affects the convergence speed and final performance of the algorithm. Inspired from [7], we propose a novel reward function:

$$r_t = \frac{WLH_{t-1} - \sum_{i=1}^{t-1} w_i l_i h_i}{WLH_{t-1} + \xi} - \frac{WLH_t - \sum_{i=1}^{t} w_i l_i h_i}{WLH_t + \xi}. \tag{15}$$

Where H_t denotes the height of the smallest bounding box containing t boxes within the bin. W and L denote the width and length of the bin. The design of the reward function is based on the change in the gap ratio in the bin. ξ is a very small positive real number to avoid the first item of the reward function will become not a number when t = 1. The proposed reward function distinguishes itself from the existing reward function in the 3D-BPP in two aspects. First, the agent does not need to wait until the end of an episode to get a reward but can get a reward at each step, which alleviates the sparse reward problem. Second, the numerical values of the proposed reward function are independent of the scales of the problem instances, and the denominator of the reward function effectively serves as a normalization process. So this is an efficient general scheme in the 3D-BPP. The agent gets a meaningful reward at each step, and the accumulated reward is the negative number of the final gap ratio in the bin:

$$\begin{aligned} G = \sum_{t=1}^{N} r_t &= \frac{WLH_0}{WLH_0 + \xi} - \frac{WLH_N - \sum_{i=1}^{N} w_i l_i h_i}{WLH_N + \xi} \\ &\approx -\frac{WLH_N - \sum_{i=1}^{N} w_i l_i h_i}{WLH_N}. \end{aligned} \tag{16}$$

Therefore agent learns the policy that can reduce the final gap ratio in the bin during the training process.

4.5.3. *Training process*

We employ on-policy reinforcement learning for the training process.

$$\begin{aligned} &L(\theta, \alpha | s_t) = L(\theta_a | s_t) + L(\theta_c | s_t) + L(\alpha | s_t) \\ &L(\theta_a | s_t) = E_{a_t \sim p_{\theta_a}} \left[-\hat{A}_t^{\gamma,\lambda}(s_t, a_t) * \log p_{\theta_a}(a_t | s_t) \right] \\ &L(\theta_c | s_t) = E_{a_t \sim p_\theta} \left[(V_{\theta_c}(s_t) - (\hat{A}_t^{\gamma,\lambda}(s_t, a_t) + V_{\theta_c}(s_t)))^2 \right] \\ &L(\alpha | s_t) = E_{a_t \sim p_\theta} \left[\bar{H} - \log p_\theta(a_t | s_t) \right]. \end{aligned} \tag{17}$$

Where $\log p_{\theta_a}(a_t|s_t) = \log p_{\theta_s}(\pi_t^s|\boldsymbol{q}_t^s) + \log p_{\theta_r}(\pi_t^r|\boldsymbol{q}_t^r)$, it contains the selection policy and the rotation policy. $\bar{H}$ denotes target entropy, which has the effect of keeping the policy entropy close to it. $L(\theta_a|s_t)$ denotes actor loss function based on policy gradient. $L(\theta_c|s_t)$ denotes critic loss function. $L(\alpha|s_t)$ denotes the entropy regularization temperature loss function.

As shown in Alg. 1, the algorithm gets T-step experiences for GAE. After each trajectory segment, we perform a one-step parameter update for actor and critic networks and temperature α.

5. Experiments

5.1. *Experimental setup*

In all our experiments, we adopt mini-batches of fixed size 128. The dynamic boxes state sub-encoder has two multi-head self-attention layers. The gradient height map sub-encoder has three convolutional layers. Both the box selection decoder and the rotation decoder have

Algorithm 1 Dynamic Multi-Modal Deep Reinforcement Learning Framework.

Input: Training set X of size Z, instance size N, batch size B, trajectory segment size T.

Parameter: Initialize actor and critic network parameters θ_a, θ_c.

```
1:  for t=1 to Z/B do
2:    Randomly sample a batch from X.
3:    for i=1 to N/T do
4:      for j=1 to T do
5:        Calculate $V_{\theta_c}(s_{i*T+j})$.
6:        Calculate the set of box embeddings $E_{i*T+j}$.
7:        Calculate the gradient height map embedding $\boldsymbol{h}_{i*T+j}$.
8:        Calculate the selection decoder input$\boldsymbol{q}^s_{i*T+j}$.
9:        Sample box index $\pi^s_{i*T+j}$ from $p_{\theta_s}(\cdot|\boldsymbol{q}^s_{i*T+j})$.
10:       Calculate the rotation decoder input $\boldsymbol{q}^r_{i*T+j}$.
11:       Sample box rotation $\pi^r_{i*T+j}$ from $p_{\theta_r}(\cdot|\boldsymbol{q}^r_{i*T+j})$.
12:       Use LB to place the box.
13:       Update state, and calculate reward by Eq. (15)
14:     end for
15:     Estimate advantages by Eq. (14).
16:     Calculate loss function by Eq. (17).
17:     Calculate gradient $g_{\theta,\alpha} = \triangle L_{\theta,\alpha}$.
18:     Update $\theta, \alpha = ADAM(\theta, \alpha, g_{\theta,\alpha})$.
19:   end for
20: end for
```

Output: θ_a, θ_c.

one multi-head attention layer. The critic network has two multi-head attention layers. All attention layers have eight attention heads and apply batch normalization. We set the attention hidden size as $h = 128$, except for the feed-forward layers, which have 512 units. We train our model with Adam optimizer [24], the initial learning rates of the actor model and critic model are $1e^{-5}$ and $1e^{-4}$ respectively, and decay by a factor of 0.96 every 2000 steps. We set the target entropy $\bar{H}$ as 0.6. For better stability, we use gradient clipping at 5.0. We set the hyper-parameters γ, λ, and C as 0.99, 0.98, and 10, respectively. Our model has 1.38M parameters. We adopt 100,000 steps to train the model. It takes two days of training for 20 boxes on a single GeForce RTX 3090 GPU.

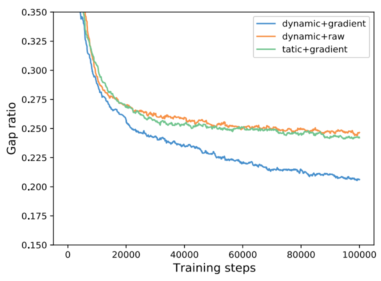
5.2. *Dataset*

The dataset used for training the DMRL-BPP framework consists of randomly generated boxes within a bin of fixed dimensions $W \times H = 100 \times 100$. The sides of the boxes are sampled from integers in the range of [20, 80]. This setup simulates a broad variety of packing scenarios to prevent the model from overfitting to specific box sizes.

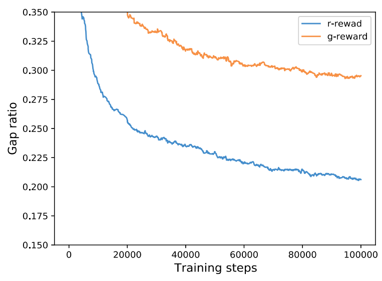
For handling bins and boxes that differ from these training dimensions, our framework employs a scaling technique. Bins and boxes that are proportionally larger or smaller are adjusted to match the $W \times H = 100 \times 100$ dimensions. After scaling, the placement strategy is determined using the process described earlier. Once the placement strategy within the scaled bin is obtained, the dimensions and placement coordinates are resized back to their original scale. This approach ensures that the DMRL-BPP framework can be effectively applied to a wide range of bin and box sizes, thereby enhancing its practical utility and generalization across different operational contexts.

5.3. *Ablation study*

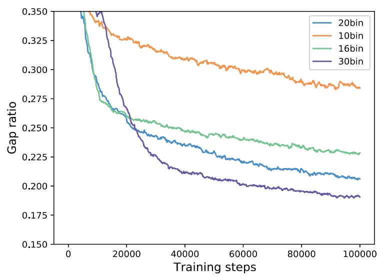
To validate the effectiveness of our proposed framework, randomization was incorporated in both the selection and rotation steps. During the box selection phase, a box was chosen randomly from the available pool. Following this selection, a rotation direction was also



(a) The learning curves of encoder input ablation study.

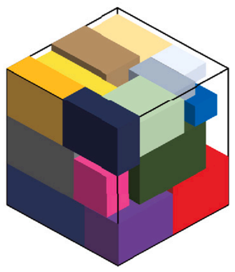


(b) The learning curves of reward function ablation study.

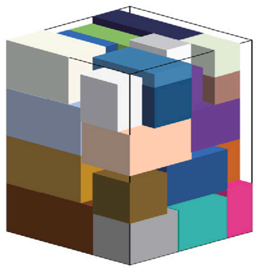


(c) The learning curves of 10, 16, 20, and 30 boxes.

Fig. 2. Learning curve for ablation studies and overall performance.



(a) 20 boxes



(b) 30 boxes

Fig. 3. The visualizations of the packing results.

determined at random. The same heuristic packing placement method was then applied (as detailed in Section 4.4). The results presented in Table 2 highlight the pivotal role of our dynamic multi-modal encoder, selection decoder, and rotation decoder components in achieving effective outcomes.

Multi-modal Input of Encoder. We first conduct an ablation study to evaluate the effectiveness of the encoder input. Here we perform ablations on the 20 boxes dataset. In this paper, the multi-modal input of encoder contains the dynamic boxes state, which describes the dynamic basic information of all boxes, and the gradient height map, which provides more auxiliary information (dynamic+gradient). Three settings are tested, one with static boxes state and gradient height map (static+gradient), one with dynamic boxes state and raw height map (dynamic+raw), and the other with our proposed method (dynamic+gradient). Bin gap ratio $r = \frac{WLH_N - \sum_{i=1}^{N} w_i l_i h_i}{WLH_N}$ is used to measure the packing performance. The results of this ablation are plotted in Fig. 2(a). Dynamically updating boxes state does boost learning, which indicates the necessity of using dynamic boxes state to provide basic information about all boxes. Adopting the gradient height map also helps to improve packing performance, which demonstrates the effectiveness of the gradient height map in offering auxiliary information.

Reward Function. The sparse reward problem is a critical challenge in DRL. In the 3D-BPP domain based on deep reinforcement learning, [7] designed a reward function based on the change of the current volume gap of the bin for solving the sparse reward problem. We conduct another ablation study to evaluate the effectiveness of our reward function. Two settings are tested, one with the reward function as in [7] (g-reward) and the other with our proposed reward function (r-reward). The results of this ablation are plotted in Fig. 2(b), which demonstrate that the reward function we proposed is not only independent of the scales of the problem instance but also outperforms g-reward.

5.4. Overall performance

As introduced in Section 5.2, we adopt the dataset which randomly generates boxes, in which $W \times H = 100 \times 100$ for the bin, the sides of boxes are sampled from integers in the range of $[20, 80]$. We evaluate our proposed method on 10 boxes, 16 boxes, 20 boxes, and 30 boxes. We then compare against six different methods, including: (1) Randomization Setting (RS) (as detailed in Section 5.3); (2) Genetic Algorithm (GA) [25]; (3) Multi-Task Selected Learning (MTSL) [22]; (4)

Table 2
The gap ratios of DMRL-BPP and baselines.

	RS	GA	MTSL	CQL	MM	ATTEND2PACK	DMRL-BPP
10	48.0%	27.2%	50.2%	27.4%	–	–	27.0% ± 0.0094
16	44.1%	31.5%	45.3%	30.6%	–	–	22.8% ± 0.0052
20	43.0%	33.9%	42.6%	32.4%	28.2%	23.3%	21.1% ± 0.0021
30	41.6%	37.4%	40.6%	30.2%	24.5%	20.3%	18.9% ± 0.0018

Conditional Query Learning (CQL) [7];(5) Multimodal Deep Reinforcement Learning (MM) [17]; (6) attend2pack [8], the state-of-the-art DRL method for solving the 3D-BPP. As in [22], all DRL-based methods for solving the 3D-BPP, including ours, sample 128 solutions and choose the best one as the final output. Table 2 shows the bin gap ratios and their variances of our proposed method on 1e4 test samples after 100,000 training steps. We achieve 7.8%, 2.2%, and 1.4% improvement than the state-of-the-art 3D-BPP approach based on DRL in 16 boxes, 20 boxes, and 30 boxes, respectively. DMRL-BPP is superior than other methods in most cases but is slightly less competitive than CQL in 10 boxes. Fig. 2(c) shows the learning curve of the training process. Our method shows superior stability, corresponding to low variance in Table 2. In all DRL-based methods for solving the 3D-BPP except ours, the learning curves of the training process frequently encounter either oscillating or catastrophic forgetting. This confirms that it is inappropriate to learn policy directly in such a large action space. Adopting a heuristically designed strategy for the placement step is equivalent to defining a policy for the placement step. This can not only avoid oscillation but also focus on learning better policy in the more important selection and rotation steps. The stability of the learning curve also benefits from the dynamic multi-modal encoder, which delivers sufficient information at each time step. We show the visualizations of the packing results in Fig. 3.

6. Practical applications of the DMRL-BPP framework

Our proposed framework has significant practical implications in industries where space optimization is crucial. Specifically, in logistics and warehousing, our model can dynamically optimize the packing of diverse item types into containers or storage areas, thus maximizing space utilization and reducing operational costs. For instance, integration into automated logistics systems allows for more efficient route and load planning, decreasing the need for multiple trips and reducing environmental impact through lower emissions. By implementing DMRL-BPP, companies can achieve more efficient use of space, leading to economic savings and environmental benefits by minimizing the resources required for transport and storage.

7. Conclusions and future work

In this paper, we introduce a dynamic multi-modal deep reinforcement learning framework designed to address the challenges of the 3D Bin Packing Problem (3D-BPP). Our framework features a dynamic multi-modal encoder that effectively processes multi-modal information, coupled with a reward function that alleviates the issue of sparse rewards—a common obstacle in 3D-BPP. Our numerical results demonstrate that DMRL-BPP achieves superior performance and stability compared to other methods.

As we continue to develop the DMRL-BPP framework, our research will focus on several key areas to enhance its utility and performance:

- **Exploring Additional Constraints:** Future versions of our model will incorporate additional practical constraints such as weight distribution, item fragility, and stability within the bin. Addressing these factors is crucial for applications in industries like pharmaceuticals and electronics, where handling sensitive items is common.
- **Extending to Multi-Bin Scenarios:** We aim to adapt the DMRL-BPP framework to complex scenarios involving multiple bins. This will allow for optimization across larger logistic networks, improving the framework's applicability to industrial-scale operations.

These targeted research initiatives will enable us to refine the DMRL-BPP framework further, broadening its applicability and improving its effectiveness in real-world applications.

CRediT authorship contribution statement

Anhao Zhao: Writing – review & editing, Writing – original draft, Methodology, Investigation. **Tianrui Li:** Writing – review & editing, Supervision. **Liangcai Lin:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] I. Sánchez-Díaz, J. Holguín-Veras, X.J. Ban, A time-dependent freight tour synthesis model, Transp. Res. B 78 (2015) 144–168.
[2] Q. Meng, X. Wang, Intermodal hub-and-spoke network design: incorporating multiple stakeholders and multi-type containers, Transp. Res. B 45 (4) (2011) 724–742.
[3] E.G. Coffman Jr., M.R. Garey, D.S. Johnson, R.E. Tarjan, Performance bounds for level-oriented two-dimensional packing algorithms, SIAM J. Comput. 9 (4) (1980) 808–826.
[4] I. Bello, H. Pham, Q.V. Le, M. Norouzi, S. Bengio, Neural combinatorial optimization with reinforcement learning, 2016, arXiv preprint arXiv:1611.09940.
[5] O. Vinyals, M. Fortunato, N. Jaitly, Pointer networks, Adv. Neural Inf. Process. Syst. 28 (2015).
[6] M. Nazari, A. Oroojlooy, L.V. Snyder, M. Takác, Deep reinforcement learning for solving the vehicle routing problem, 2018, arXiv preprint arXiv:1802.04240.
[7] D. Li, C. Ren, Z. Gu, Y. Wang, F. Lau, Solving packing problems by conditional query learning, 2019.
[8] J. Zhang, B. Zi, X. Ge, Attend2Pack: Bin packing through deep reinforcement learning with attention, 2021, arXiv preprint arXiv:2107.04333.
[9] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.
[10] W. Kool, H. Van Hoof, M. Welling, Attention, learn to solve routing problems! 2018, arXiv preprint arXiv:1803.08475.
[11] C. Chen, S.-M. Lee, Q. Shen, An analytical model for the container loading problem, European J. Oper. Res. 80 (1) (1995) 68–76.
[12] E. Baltacioglu, The Distributer's Three-Dimensional Pallet-Packing Problem: A Human Intelligence-Based Heuristic Approach, Tech. Rep., AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING AND . . . , 2001.
[13] H. Hu, X. Zhang, X. Yan, L. Wang, Y. Xu, Solving a new 3d bin packing problem with deep reinforcement learning method, 2017, arXiv preprint arXiv:1708.05930.
[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[16] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer normalization, 2016, arXiv preprint arXiv:1607.06450.
[17] Y. Jiang, Z. Cao, J. Zhang, Solving 3D bin packing problem via multimodal deep reinforcement learning, in: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, 2021, pp. 1548–1550.
[18] R. Hu, J. Xu, B. Chen, M. Gong, H. Zhang, H. Huang, TAP-Net: transport-and-pack using reinforcement learning, ACM Trans. Graph. 39 (6) (2020) 1–15.
[19] A.G. Ramos, J.F. Oliveira, J.F. Gonçalves, M.P. Lopes, A container loading algorithm with static mechanical equilibrium stability constraints, Transp. Res. B 91 (2016) 565–581.
[20] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, B. Coppin, Deep reinforcement learning in large discrete action spaces, 2015, arXiv preprint arXiv:1512.07679.
[21] J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, M. Ostendorf, Deep reinforcement learning with an unbounded action space 5, 2015, arXiv preprint arXiv:1511.04636.
[22] L. Duan, H. Hu, Y. Qian, Y. Gong, X. Zhang, Y. Xu, J. Wei, A multi-task selected learning approach for solving 3D flexible bin packing problem, 2018, arXiv preprint arXiv:1804.06896.
[23] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, 2015, arXiv preprint arXiv:1506.02438.
[24] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[25] Y. Wu, W. Li, M. Goh, R. De Souza, Three-dimensional bin packing problem with variable bin height, European J. Oper. Res. 202 (2) (2010) 347–355.