Online 3D Bin Packing Algorithm Based on Heuristic Deep Reinforcement Learning

Baoxin Zhang[1] , Hui Zhao [1*]

[1]*School of Electrical Engineering and Automation, Tianjin University of Technology, Tianjin, 300384, China*

baoxin128@sina.com

*Corresponding author: Zhaohui3379@126.com

2025 IEEE International Conference on Mechatronics and Automation (ICMA) | 979-8-3315-1426-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICMA65362.2025.11120685

***Abstract* - In the field of logistics and transportation, the three-dimensional bin packing problem(3D-BPP) is one of the core challenges in optimizing transportation resource utilization. Given the superior learning and optimization capabilities of deep reinforcement learning (DRL) in complex decision-making, applying deep reinforcement learning to solve the online three-dimensional bin packing problem has become a key research direction in intelligent logistics. This study proposes a deep reinforcement learning algorithm integrating an Event Point trigger and a dynamically updated Empty Maximal Space heuristic rule, along with a multi-objective combined reward function. Experimental results demonstrate that the improved model significantly outperforms traditional deep reinforcement learning algorithms in online three-dimensional bin packing, achieving a 9% higher average space utilization and a 27.3% faster convergence speed.**

Index Terms* - *Online 3D Bin Packing Problem . Combinatorial Optimization Problem. Deep Reinforcement Learning. Heuristic.

I. INTRODUCTION

Traditional methods for solving 3D bin packing problems mainly rely on exact algorithms and heuristic rules. However, these methods often face exponentially increasing computational complexity with growing cargo size diversity or larger loading scales. In particular, due to the fixed-rule nature of heuristic algorithms, they exhibit limited adaptability to dynamic environmental changes and complex constraints; making it difficult to guarantee the global optimization of packing schemes. Therefore, developing intelligent optimization algorithms with online decision-making and real-time response capabilities is crucial. By simulating the interaction mechanism between the agent and the environment, deep reinforcement learning can independently optimize decision-making strategies in the process of trial and error learning without relying on preset heuristic rules, so it can effectively adapt to the size change of the container object and the requirements of complex scenes.

Martello et al. proposed an exact algorithm based on branch-and-bound and dynamic programming, which can quickly find optimal solutions[1] . However, when dealing with large-scale problems, it requires a long computational time and is not suitable for practical applications. The Extreme Point-based Heuristics proposed by Crainic et al. improve space utilization by placing items at the poles[3] . Although the above algorithm has made some progress in the application scenario of offline packing problem, it has great limitations in the case of not predicting the size information of all objects. Falkenauer et al. designed a hybrid algorithm combining grouping genetic algorithm and domination criterion, in which grouping genetic algorithm can adapt to the structure of grouping problems through special encoding and genetic operations, and their combination can effectively improve solution efficiency and quality[4] . Yang et al. proposed a DRL-based 3D-BPP optimization method that significantly improves packing efficiency by introducing physical heuristics and unpacking heuristics[5] . This method not only inherits the advantages of the heuristic algorithm, but also further optimizes the packing strategy through the learning ability of DRL. Zhao et al. proposed a method based on constrained DRL to guide the DRL training process by predicting the feasibility mask of placement actions, avoiding the exploration of invalid actions[6] . Xiong et al. proposed a DRL method based on Transformer, which improves the performance of DRL in complex scenarios by introducing Packing Configuration Tree to represent container space[7] . Wong et al. proposed the Hybrid Heuristic Proximal Policy Optimization method, which combines the heuristic algorithm for bin packing with the Proximal Policy Optimization algorithm of deep reinforcement learning, providing a new solution for the online 3D-BP problem[7] .

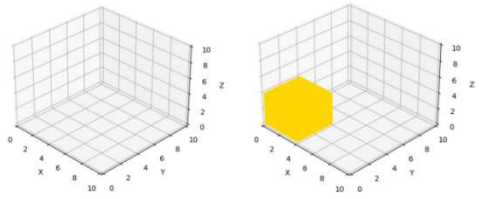
We aim to design a heuristic deep reinforcement learning algorithm by constructing heuristic initialization strategies that integrate domain prior knowledge, thereby providing high-quality initial solution distributions for DRL agents. This approach effectively reduces inefficient exploration and constrains the search space. Secondly, aiming at the problem of space fragmentation in online packing scenarios, the framework innovatively designs a multi-scale reward function based on geometric compactness. By accurately quantifying key indicators such as the three-dimensional space utilization rate of the container, it guides the agent to learn the optimal

loading strategy for maximizing space utilization under the condition of satisfying physical constraints.

II. Related work

A. Problem Description

In order to simplify the three-dimensional bin packing problem, the online packing problem is defined as: objects with random size, uniform distribution of mass and regular shape arrive in turn, and the agent needs to put the current goods into a container of fixed size, and cannot adjust the goods that have been placed once the placement position and posture are determined. In order to make the encoding process more realistic, the following constraints are set: The supporting constraint condition is that the bottom of the goods must be effectively supported in the process of stacking, and the supporting area should reach more than 80%, and the object is not allowed to be suspended; The boundary constraint is that the goods cannot go beyond the edge of the container; The non-coincidence constraint is that goods cannot be overlapped; Parallel constraint means that the side length of the goods and the corresponding side length of the container are parallel to each other to ensure the stability and regularity of the goods stack. The packing constraint is that each incoming cargo should be placed and stopped when the maximum packing rate is reached or the cargo is not placed in the container. In the process of modeling the three-dimensional packing problem, the corner point of the container is taken as the origin of the coordinates, and the object is placed as shown in Figure 1.



(a)Initial state (b) Place 1 item

Fig. 1 container state

B. Markov Decision Process

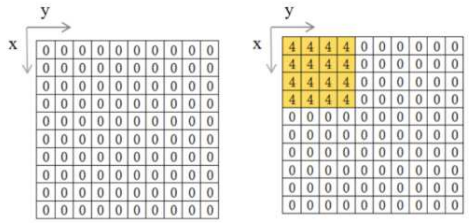
The three-dimensional packing problem can be formalized as A Markov Decision Process, which is mathematically represented as a (S,A,P,R,γ) quintuple. State space S represents the three-dimensional space occupancy of the current container, including the geometric distribution of loaded items and the remaining available space. Action space A is defined as the set of currently feasible operations, covering the placement coordinates and spatial posture (such as rotation Angle) of items. The state transition probability P describes the transition probability distribution of the system state after performing a given action in a particular state. The reward function R is a quantitative environmental feedback signal, which is used to evaluate the pros and cons of action choice. Discount factor γ is used to adjust the influence weight of future rewards on the current decision, and balance the short-term returns and long-term returns in the process of strategy optimization.

C. State space and action space

The state space in this study can be formalized in the following form: $S=(B_t, M_t, N_t)$.

Where B_t represents the remaining space in the space, the two-dimensional array represents the space occupation of the horizontal plane of the container, and the height occupied in the vertical direction of the corresponding position of each element value in the array. In this way, the agent can intuitively understand the remaining space of different positions in the container, so as to choose the appropriate placement position. M_t describes the position and shape information of the items placed in the space, including the location and size information of the items, and stores the vertex coordinate information and height information of each already placed object for collision detection and space occupancy calculation, so as to avoid overlapping between the subsequent placement of items and the already placed items. N_t is used to describe the size of the item that is about to arrive.

As can be seen in Figure 2 below, the initial state of the container is empty, so the height values of the two-dimensional array are all 0. After placing the first square object with a side length of 4, the state in the container has changed, and the shadow area on the bottom of the container is 4x4, corresponding to the table value of 4.



(a)Height map in the initial state (b) Place a heightmap of 1 item

Fig. 2 Container stacking heightmap

The action space A is defined as the complete set of all feasible operations. The action $a_t \in A$ performed at time step t is composed of two key decision dimensions: spatial positioning decision and attitude choice decision. The spatial positioning decision determines the three-dimensional coordinates (x,y,z) of the item to be loaded in the container coordinate system, while the attitude selection decision specifies the attitude of the item to be placed. The multi-objective reward function guides the agent to optimize the packing strategy by quantifying the space utilization index. The reward function is composed of two subfunctions with definite physical meaning: space utilization ratio reward and space compactness reward. The space utilization ratio calculates the reward according to the proportion of the volume of the item to be placed to the total volume of the container, and this part of the reward encourages the agent to place as many items as possible to improve the space utilization.The formula is as follows:

$$R_1 = \alpha \frac{l_i \cdot w_i \cdot h_i}{L \cdot W \cdot H} \tag{1}$$

Where α is the scale factor, l_i w_i h_i is the size of the goods,

L, W, H is the size of the container.

The space compactness reward is calculated by calculating the volume of the remaining free space in the container. This part of the reward encourages the agent to place the item in a compact location, reducing the dispersion of free space and improving the effective utilization of space. The formula is as follows:

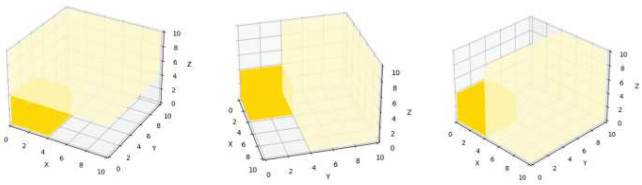
$$R_2 = \beta \frac{1}{V_S + 1} \tag{2}$$

Where β is the scale coefficient and Vs is the volume of remaining space in the container.

The composite reward function realizes multi-dimensional evaluation of packing quality, R_1 focuses on macro loading efficiency and R_2 focuses on micro spatial structure. This reward mechanism can effectively guide the strategy network to balance immediate benefits and long-term optimization goals in the process of exploration.

D. Heuristic rule design

Empty Maximal Space is a method used to represent and manage the remaining space in three-dimensional packing bin problem. Its core idea is to divide the unoccupied space in the container into a series of largest and non-overlapping rectangular spaces. When a new item is placed, the original maximum subspace is removed and a new subspace is generated.The Empty Maximal Space focuses on the upper, right, and rear spaces of the cargo, as shown in Figure 3.



(a) upper space (b) rear space (c) right space

Fig. 3 Schematic diagram of Empty Maximal Space

The heuristic method proposed in this paper was based on the Event Point triggering mechanism and the Empty Maximal Space dynamic maintenance strategy to carry out 3D boxing optimization. The method uses an event-driven status update mechanism to trigger Event point after the item is successfully placed, update the container space occupancy status synchronously and recalculate EMS segmentation. During initialization, the bottom surface of the whole container is regarded as the only EMS. After placing items, the original EMS is decomposed into the upper, right and last three subspaces by axial-aligned Split method, and the maximum continuous region is retained. Based on greedy strategy, the largest EMS is preferentially selected for placing attempt when the conditions are met. The R-tree index is used to accelerate the spatial query. Ensure the feasibility of subsequent placement by updating the EMS list and height matrix in real time. We have conducted a detailed computational complexity analysis of the EMS dynamic maintenance mechanism. The key steps involved in the analysis include the decomposition of the EMS after item placement and the spatial query using the R-tree index. We have determined that the computational complexity of these steps is O(n) and O(log n), respectively, where n is the number of items. The method innovatively combines event triggering with space segmentation mechanism, which significantly improves space utilization while ensuring computational efficiency. Experimental results show that the framework not only effectively avoids local optimal trap, but also effectively avoids local optimal trap. Its dynamic spatial maintenance mechanism, event-driven status update strategy and fast query algorithm based on spatial index together constitute an efficient 3D bin packaging optimization solution, which shows excellent performance in dealing with large-scale packaging problems.

E: Network Structure

Actor-Critic algorithm, as a typical reinforcement learning framework, realizes the efficient solution of Markov decision process through the cooperative optimization mechanism of strategy network and value network. The main task of Actor (policy network) is to generate action decisions in a given state, select actions according to the current policy, and represent the policy function π(a|s) through the neural network, which gives the corresponding action probability distribution according to the current state s. The main function of a Critic (Value network) is to evaluate the value of the actions chosen by the Actor, measuring the quality of the strategy by calculating time-difference errors and advantages. The network structure is as follows：

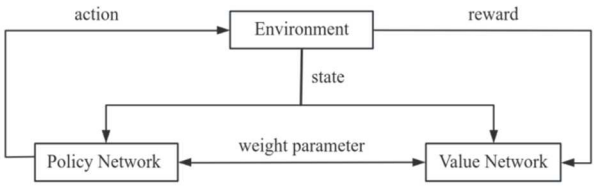


Fig. 4 structure diagram of the Actor-Critic algorithm

The update of Actor adopts the principle of policy gradient, and the expression of policy gradient is as follows:

$$g = E\left[\sum_{t=0}^{T} \varphi_t \nabla_\theta \log \pi_\theta(a_t \mid S_t)\right] \tag{3}$$

Where a_t represents the action taken by the agent at time step t; $S_t$stands for agent The state of the environment at time step t; φ_t is the advantage function, which measures the advantage degree of taking action in the state compared with the average case; $\nabla_\theta$is the gradient operator of policy parameters, which is used to indicate the direction in which policy parameters should be updated to improve performance. $\pi_\theta(a_t \mid S_t)$ is the policy function, which represents the probability that the agent takes an action a_t in state S_t under the policy with parameter Θ.The Critic network is updated by time difference residual, and the expression of its value function loss function is as follows:

$$L(\omega)=\frac{1}{2}\left(R+\gamma V_{\omega}S_{t+1}-V_{\omega}S_{t}\right)^{2} \quad (4)$$

$V_{\omega}S_{t+1}$ is the estimate of the value function in the state S_{t+1}; $V_{\omega}S_{t}$ is the estimate of the value function in its state S_t.

The gradient of the final value function is calculated as follows:

$$\nabla_{\omega}L(\omega)=-\left(R+\gamma V_{\omega}(S_{t+1})-V_{\omega}(S_{t})\right)\nabla_{\omega}V_{\omega}(S_{t}) \quad (5)$$

The deep reinforcement learning framework constructed in this study uses convolutional neural network to extract spatial features, and its core function is to analyze the geometric spatial relationships in the packing layout. The network realizes hierarchical extraction of local spatial features through multi-layer convolution operation, which can effectively identify the occupied state of adjacent areas to avoid overlapping items, and detect highly discontinuous areas to ensure placement stability. The network architecture consists of a flat layer and a fully connected layer, which integrates local spatial features into global state representations.

The model adopts two-flow network design, which includes two core components: policy network and feature extractor. Based on the features extracted from the basic network, the strategy network realizes the state-action mapping function, and the output includes action selection, action probability distribution and state value evaluation. The network quantifies decision quality by calculating strategy entropy and action probability distribution, and provides gradient signal for strategy optimization.

The feature extractor is specially designed to branch out three functional heads after sharing the convolutional layer: a spatial mask prediction head to identify viable placement areas, a strategy network head to generate action distribution, and a value function head to assess state value. This architecture design can automatically learn the optimal space allocation strategy to effectively avoid invalid placement. Through co-optimization with multi-objective reward function, the model can significantly improve the stability of packing scheme while ensuring space utilization.

III. EXPERIMENTAL RESULTS

The experimental platform is a computing workstation equipped with Intel(R) Core(TM) i7-4700HX processor, 16GB system memory and NVIDIA GeForce RTX 4070 graphics card (8GB video memory). The operating system is ubuntu, and the experimental framework is built based on Python programming language. In order to verify the generalization ability and practical application value of the algorithm, two sets of data sets with different characteristics are designed in this study: the size parameters of the items to be loaded are randomly generated within the preset value range, and diversified test scenarios are constructed through systematic parameter combinations.

This experimental design method can not only comprehensively evaluate the performance of the algorithm in cases of different scale problems, but also effectively simulate the randomness characteristics of the size of items in the actual logistics scene, so as to ensure that the research conclusions are fully representative and practical guiding significance. The length, width and height of the container in the first set of data are set as L=10, W=10, H=10; The dimensional information of the item meets the following conditions: $1\le l_i\le L/2$, $1\le w_i\le W/2$, $1\le h_i\le H/2$; In the second group of data, the length, width and height of the container are set as L=30, W=20, H=10; The dimensional information of the item meets the following conditions: $10\le l_i\le L/2$, $3\le w_i\le W/2$, $1\le h_i\le H/2$.

The three-dimensional bin packing based on heuristic deep reinforcement learning adopts Actor-Critic framework, the shared layer consists of two 256-dimensional fully connected layers, followed by ReLU activation function, the optimizer uses AdamW, the initial learning rate is 1e-4, the weight attenuation is 1e-5, and the StepLR scheduler is coordinated.The algorithm flow is shown in Figure 5：

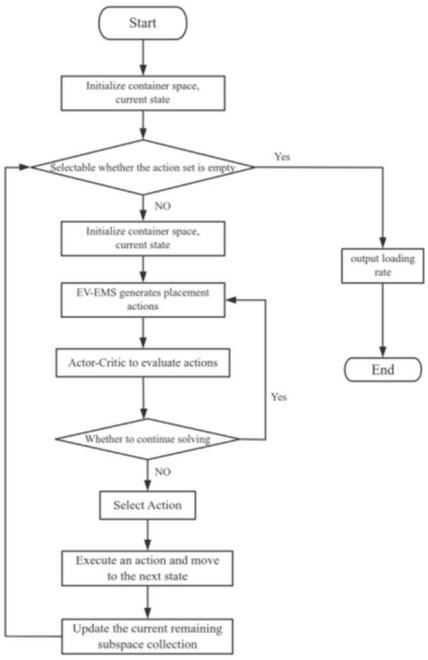


Fig. 5 Algorithm flow chart

Figure 6 shows the average space utilization curve based on the heuristic DRL model and the DRL model. The upper curve is the space utilization curve of deep reinforcement learning with EV-Ems heuristic algorithm added, and the lower curve is the space utilization curve of the deep reinforcement learning model. It can be observed that the agent can quickly learn the correct packing method by interacting with the environment and exploring strategies in the early stage of training, so the average utilization rate increases rapidly. In the middle period of training, the agent tried some wrong packing methods in the process of exploration, which led to fluctuations in space utilization, but it was constantly optimized and adjusted. In the later stage of training, the average space utilization rate of the improved model is stable at about 0.72, while that of the unimproved

model is maintained at about 0.63, which cannot be improved even after a lot of training. Finally, the agent converges to a relatively stable state, close to the performance upper limit of the model. During training, the improved model shows higher space utilization and smaller late-stage fluctuations than the unimproved one.

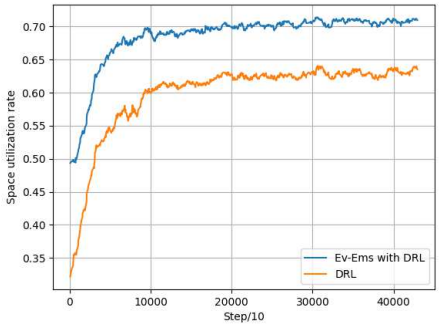


Fig. 6 Average space utilization

Figure 7 shows the curve of the reward function changing with the number of training steps. It is used to measure the quality of the strategy of the algorithm at each training step. The better the value, the better the strategy. Ev - Ems with DRL has a faster learning speed, can reach a higher reward value more quickly and converge to a better solution; the DRL method has a slower learning speed and the finally converged reward value is also lower. Therefore, the Ev - Ems with DRL method outperforms the pure DRL method in the 3D bin packing program based on deep reinforcement learning.

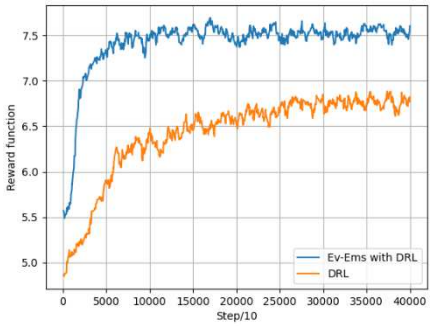


Fig.7 Reward function value curve

In the comparative experiment, we strictly controlled the experimental conditions to ensure the comparability of the results: we used a unified data generation strategy to create random data sets, maintain the consistency of hyperparameters such as container size, learning rate and batch size, and realized the parameter initialization of each model by fixing random seeds, thus eliminating the impact of differences in initial conditions on the experimental results.The experimental evaluation indexes include three key performance indexes: average space utilization rate, average load quantity and average calculation time. In order to comprehensively examine the adaptability of the algorithm, two containers of different specifications and their corresponding cargo size data sets were set up for cross-verification.This multi-dimensional, multi-scenario comparative analysis framework can not only objectively evaluate the comprehensive performance of the algorithm, but also deeply investigate its generalization ability on different scale problems. Four representative methods were selected as the benchmark comparison, and the reliability of the result statistics was ensured by strict parameter control and repeated experiments.The experimental results are shown in Table 1and Table 2.

As can be seen from Table 1, our algorithm has the highest average space utilization rate, which can reach 72.4%, which can make the space proportion of items in the container the largest, and the optimal space efficiency, while the average number of loaded items is also the most, the running time is longer than zhao and xiong's method, and the running speed is not dominant. In general, the algorithm designed in this paper has excellent performance in space utilization and average load quantity.

TABLE I
Packing Yield of Different Algorithms in Dataset 1

Algorithm	Sp	N	T
Based on heuristic algorithm	48.10%	18.4	0.12
hybrid genetic algorithm	62.10%	20.6	0.15
Zhao et al	70.20%	28.1	0.07
xiong et al	71.20%	28.7	0.09
ours	72.40%	30.3	0.14

According to the comparison between Table 2 and Table 1, it can be seen that the volume of the container and the size of the item have an impact on the three-dimensional packing problem. The above methods all reduce the average packing rate and the number of containers to different degrees due to changes in data.

TABLE II
Packing Yield of Different Algorithms in Dataset 2

Algorithm	Sp	N	T
Based on heuristic algorithm	45.10%	16.8	0.12
hybrid genetic algorithm	58.10%	18.5	0.15
Zhao et al	64.10%	26.5	0.07
xiong et al	63.20%	26.2	0.09
ours	63.20%	27.2	0.14

In order to verify the generalization effect of the model, different sizes of containers were used to test the placement effect of objects. Figure 8 shows the placement renderings of the cube container with a side length of 10 In order to verify the generalization effect of the model, different sizes of containers were used to test the placement effect of objects. Figure 8 shows the placement renderings of the cube container with a side length of 10 and the corresponding cargo size. Figure 9 shows the placement renderings of the cuboid container and the corresponding cargo size.

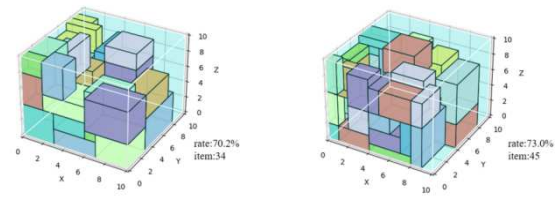


Fig. 8 The placement effect in the cube container0

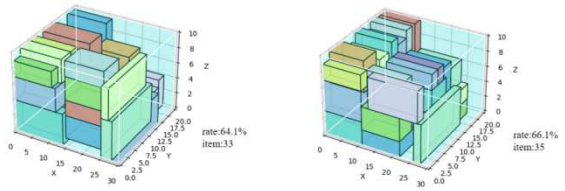


Fig. 9 The placement effect in a cuboid container

When placing items in a cuboid container, the size of the items is much smaller than the size of the container, and the dimensions of the items vary evenly in each dimension, and there are many combinations of placement methods, which are easy to fill the space by adjusting the Angle and position, so the space utilization rate is high. In contrast, in cuboid containers, the dimensions of items vary greatly. When learning how to place items in a space with obvious differences in length, width and height, the model needs to consider the relationship between object size and container boundaries, which increases the difficulty of learning placement strategies and makes it difficult to achieve a high space utilization rate. Finally, the suitability of items and container sizes may have a greater impact on space utilization.

IV. Conclusion

Compared with the common deep reinforcement learning algorithm, integrating heuristic rules can reduce invalid exploration in the initial training of the model, make the model obtain more cumulative reward value, shorten the time required for model convergence, and improve the space utilization. However, compared with the current highest online packing algorithm, there is still a certain gap, but the combination of heuristic method and deep reinforcement learning has a good performance in solving three-dimensional packing problems, and provides a favorable support for solving online three-dimensional packing problems in the future.

References

[1] P. C. Gilmore and R. E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem—Part II," *Operations Research*, vol. 11, no. 6, pp. 863 – 888, Dec. 1963.

[2] S. Martello, D. Pisinger, and D. Vigo, "The Three-Dimensional Bin Packing Problem," *Operations Research*, vol. 48, no. 2, pp. 256–267, Apr. 2000.

[3] T. G. Crainic, G. Perboli, and R. Tadei,"TS2PACK: A two-level tabu search for the three-dimensional bin packing problem," *European Journal of Operational Research*, vol. 195, no. 3, pp. 744–760, Jun. 2009.

[4] Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of Heuristics*, vol. 2, no. 1, pp. 5–30, 1996.

[5] S. Yang et al., "Heuristics Integrated Deep Reinforcement Learning for Online 3D Bin Packing," in *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 1, pp. 939-950, Jan. 2024,

[6] H. Zhao, Q. She, C. Zhu, Y. Yang, and K. Xu, "Online 3D Bin Packing with Constrained Deep Reinforcement Learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, pp. 741–749, May 2021.

[7] H. Xiong, K. Ding, W. Ding, J. Peng, and J. Xu, "Towards reliable robot packing system based on deep reinforcement learning," *Advanced engineering informatics*, vol. 57, no. 2023, 57: 102028, pp. 102028 – 102028, Aug. 2023.

[8] C.-C. Wong, T.-T. Tsai, and C.-K. Ou, "Integrating Heuristic Methods with Deep Reinforcement Learning for Online 3D Bin-Packing Optimization," *Sensors*, vol. 24, no. 16, p. 5370, Aug. 2024.

[9] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, Sep. 1983.