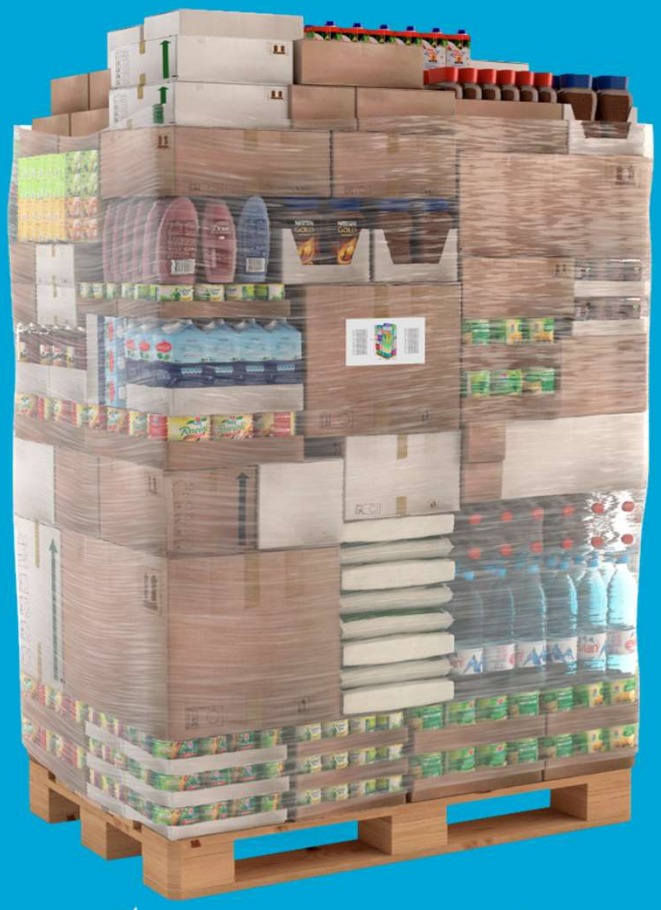
Solving the online 3D Bin Packing Problem with Graph-based Reinforcement Learning

Giovanni Corvi



Supervisors:

Dr. Cosimo Della Santina
Dr. Ronald Poelman

Thesis Committee:

Dr. Cosimo Della Santina
Dr. Ronald Poelman
Prof. Martijn Wisse

Examination date: 25th of April 2024
Awarding Institute: Technische Universiteit Delft (TU Delft)
Faculty: Faculty of Mechanical Engineering (ME)
Department: Cognitive Robotics (CoR)

TU Delft

FIZYR

Solving the Online 3D Bin Packing Problem with Graph-Based Reinforcement Learning

Giovanni Corvi

***Abstract*—The rapidly growing volume of parcel shipments is straining transportation and logistics sectors, highlighting the need for innovative solutions to optimize packing and loading processes. The online bin packing problem (BPP), an NP-hard computational problem, finds practical applications in numerous sectors, including modern packaging and intelligent logistics. This study proposes a novel reinforcement learning (RL) approach to tackle the online 3D-BPP emphasizing applicability and versatility. The key innovation is the representation of the packing scene as a graph, enabling effective encoding of task-specific high-level features. This graph-based structure serves as the foundation for an RL agent designed to learn an optimal packing strategy through dynamic interaction with the environment. The proposed approach uniquely operates within the continuous domain, enhancing generalization across diverse packing tasks. Experimental evaluations in both simulated environments and a real-world setting demonstrate that the solution achieves state-of-the-art performance across multiple complex three-dimensional packing scenarios.**

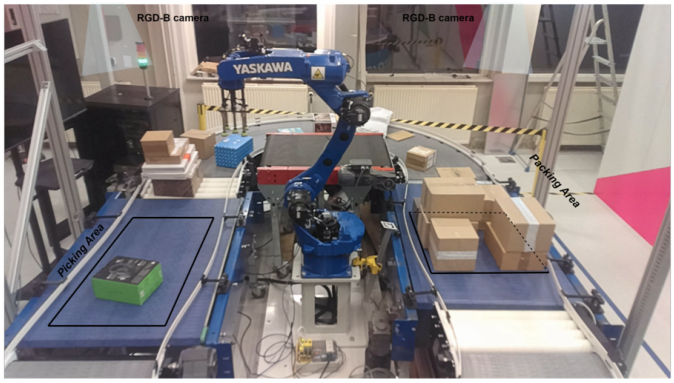


Fig. 1: Illustration of the real-world testing scenario. The system consists of a robotic arm for picking and packing items, a conveyor belt to feed incoming items, and RGB-D cameras mounted above to capture the packing scene and incoming item data.

I. Introduction

The logistic industry is witnessing an exponential rise of parcels volumes, with projections indicating the number of shipments will increase by 150% in the next four years [1]. This surge poses a substantial challenge for transportation companies, requiring innovative solutions to efficiently manage the intricate network of product logistics. Optimizing transportation resources, including the efficient packing of items, emerges as a crucial factor. Achieving an accurate and reliable solution in this context holds the potential to significantly improve customer satisfaction and cost-effectiveness in logistics, as well as ensuring personnel safety during loading and unloading procedures.

The bin packing problem is a classical optimization problem within the broader Cutting & Packing (C&P) domain, which has attracted research attention since the 1960s. In industrial applications, 'cutting' problems involve segmenting materials such as wood or paper rolls, while 'packing' problems address the arrangement of items into containers, often for logistical purposes. Despite this distinction, the dynamic interaction between material and space in both scenarios suggests that packing problems can be conceptualized as cutting large spaces into smaller units [2].

The online three-dimensional bin packing problem (3D-BPP) is a variant of the classic NP-hard problem with significant real-world applications in the logistic industry. Its objective is to efficiently load items of varying sizes into a minimum number of containers, also referred to as bins. In contrast with the classical 3D-BPP, the online problem requires real-time decision-making as items arrive sequentially without prior knowledge of the set of incoming items and their dimensions. This scenario distinctly reflects the challenges encountered in real-world logistics operations.

The real-world setting for the online 3D-BPP typically features a conveyor belt delivering items to a robot manipulator, with RGB-D cameras positioned above the picking and packing areas. These sensors provide a 2.5D perspective of both the incoming item and the current configuration of the container. A solution in this scenario processes the visual data to determine a three-dimensional pose for optimal item placement.

Numerous variants of the bin packing problem have been developed, each designed to meet the specific demands of practical constraints in real-world scenarios. These constraints range from safety considerations, such as weight limitations, to logistical factors, like loading priorities. The current study tackles a general version of the problem that is widely relevant and designed to be adaptable to more specialized scenarios. It particularly focuses on addressing the most frequently encountered safety constraints.

Historically, heuristic methods, which are based on empirical rules derived from hands-on packing experiences, have been the dominant approach for solving the online 3D-BPP. However, recent advancements have seen the emergence of deep reinforcement learning as a promising alternative approach. While learning-based approaches have generally demonstrated superior performance compared to traditional methodologies, a notable gap persists when assessing their effectiveness in realistic scenarios. These methods are predominantly designed under the assumption of discrete settings, limiting their applicability to specific cases and rendering them impractical for real-world deployment.

This study aims to address the identified research gap by

presenting a more versatile and practical solution to the online 3D bin packing problem. The proposed learning-based approach inherently operates within a continuous space, resulting in state-of-the-art performance across diverse packing tasks. The problem was formulated as a Markov decision process and tackled with a reinforcement learning approach. A distinctive feature, setting this RL solution apart from its predecessors, lies in the novel representation of the container configuration. The packing scene, combined with the characteristics of the incoming item, is encoded as a graph. This representation enhances data efficiency and provides the RL agent high-level features tailored to the specific task.

The graph-structured data derived from the packing environment is processed using the Graph Attention Network version 2 (GATv2) architecture. The reinforcement learning agent is subsequently trained with the Proximal Policy Optimization (PPO) algorithm. Within the Isaac Gym platform, a custom-build simulation environment was developed to leverage the physics simulator's advanced capabilities. This proved crucial for assessing the full packing configuration's stability, a task that has historically posed significant challenges.

Finally, as illustrated in Fig. 1, the approach is rigorously tested within a real-world scenario. This study was conducted in collaboration with Fizyr, a company focused on developing computer vision software for 'pick-and-place' applications. Their detection software played a crucial role to enable the real-world implementation and evaluation of the proposed solution. These tests confirmed the model's performance and demonstrated an effective simulation-to-reality transition, highlighting the applicability of the proposed solution in operational settings.

II. Related Work

The original bin packing problem has numerous variants, including natural generalizations such as 2D- and 3D-BPP, where higher dimensions introduce significantly increased complexity. Despite the practical relevance of the online 3D-BPP, the overwhelming majority of current literature focuses on the offline version of the problem [3]. Consequently, a wide range of approaches has been proposed to tackle this variant. These strategies include exact algorithms [4], constructive and meta-heuristics [5], tree-search methods [6], machine learning techniques [7], and approximation algorithms [8]. On the other hand, only a handful of works attempted to develop end-to-end solutions for the online 3D-BPP. Nevertheless, it is important to note that the insights gained from valuable offline studies can inform and guide the development of relevant approaches for online problems.

This section begins by examining the most relevant solutions for the online 3D-BPP, analyzing their strengths and limitations. Subsequently, it explores the application of reinforcement learning (RL) algorithms for training a decision-making agent in this setting. Finally, given the graph-structured data collected from the environment, the section examines various graph neural networks (GNNs) to select the most suitable architecture.

A. Solutions to the online 3D-BPP

When addressing the online 3D-BPP, the literature presents two primary methodologies: heuristics and learning-based approaches. In general, heuristics are practical methods employed to address complex problems that prioritize efficiency over finding optimal solutions. In the context of the 3D-BPP, heuristic algorithms can be viewed as a distilled representation of the practical knowledge gained from real-world packing experiences. These methods can be divided into two essential components. The first identifies potential empty spaces, known as placement locations, that are suitable for item packing. Common methods, drawing inspiration from offline studies, include Corner Points (CPs) [9], Extreme Points (EPs) [10], and Empty Maximal Spaces (EMSs) [5]. The second component involves selecting an optimal placement location to fulfill packing requirements. While constructive heuristics for this task are sparse in online bin packing literature, methods such as Deepest-Bottom-Left with Fill (DBLF) [11], First Fit Decreasing (FFD) [12], and Best Fit Decreasing (BFD) [13] have been employed. Although these heuristic methods demonstrated acceptable performance, their generalization capabilities exhibited significant limitations.

In recent years, learning-based techniques, particularly reinforcement learning (RL), have emerged as promising solutions for the online 3D-BPP. Kundu et al. [14] first introduced a deep RL framework to address a vision-based online 2D-BPP, employing a Double Deep Q-Network (DDQN) with a convolutional neural network (CNN) to optimize packing efficiency. A crucial aspect of this framework is the state-action representation, which significantly impacts the complexity of the learning problem and the agent's ability to capture relevant patterns. In the literature, the conventional approach for representing the container involves discretizing its bottom area to construct a heightmap. This results in a two-dimensional (LxW) grid where each cell describes the current height of the stacked items in the corresponding region. Consequently, the action set is directly derived from this heightmap, with each cell considered a potential placement location. Inherent to this representation is a trade-off between a higher resolution map, which exponentially increases the action space thus significantly slowing down policy convergence, and a lower resolution one, which limits the action space while potentially sacrificing detail and policy effectiveness. To address this challenge, two main approaches have been proposed. The first method, employed in [15] and [16], involves invalid action masking (IAM) to filter actions for feasibility, leveraging stability analyses in the 3D-BPP context. The second approach adopts a hybrid strategy, as illustrated in the studies [17] and [18], where novel heuristics are integrated with reinforcement learning for decision-making. However, findings from studies such as Yang et al.'s [16] indicate that hybrid methods for online 3D-BPP might present limitations inherited from heuristic methodologies.

While these approaches offer potential remedies, the conventional heightmap representation itself imposes inherent limitations that are challenging to overcome. These solutions have often been designed and evaluated on tailored datasets,

and their applicability to practical scenarios, where continuous space and dynamic environments are involved, remains questionable.

B. Reinforcement learning

Reinforcement learning (RL) has gained significant attention in artificial intelligence research, with deep reinforcement learning (DRL) emerging as a prominent field. By integrating neural network models into conventional RL algorithms, DRL has proven successful in mastering complex behavioral skills and addressing intricate control tasks [19]. The existing research can be broadly categorized into value-based, policy-based, and maximum entropy methods.

Value-based methods, such as Deep Q-Network (DQN) [20] and Double Deep Q-Network (DDQN) [21], aim to approximate the optimal value function, which represents the expected long-term reward for taking an action in a given state. On the other hand, policy-based methods, including Asynchronous Advantage Actor-Critic (A3C) [22], Trust Region Policy Optimization (TRPO) [23], Proximal Policy Optimization (PPO) [24], and Actor-Critic using Kronecker-Factored Trust Region (ACKTR) [25], directly optimize the agent's policy function, proving effective for scenarios with complex stochastic policies. Finally, maximum entropy methods, such as Soft Actor-Critic (SAC) [26], leverage entropy regularization to encourage exploration and diverse behavior, making them potentially data-efficient but requiring careful tuning for optimal performance. Among these approaches, PPO emerges as a compelling choice due to its balance between sample efficiency and simplicity. It consistently demonstrated strong performance across a wide range of continuous control tasks, often outperforming more complex algorithms such as ACKTR, without imposing significant computational overhead [27]. Additionally, PPO's strategy to handle the exploration-exploitation trade-off renders it desirable for optimizing stochastic policies. Its clipped objective function ensures stable learning dynamics by encouraging incremental updates to the policy, making it a robust and rational choice for tackling the online 3D-BPP with the proposed state representation.

C. Graph neural networks

Graph neural networks (GNNs) have attracted significant attention for their ability to analyze and model intricate relationships within graph-structured data. Inspired by convolutional neural networks, GNNs are specifically designed for non-Euclidean data structures, showcasing remarkable efficacy in tasks such as node classification, link prediction, and knowledge graphs [28]. Employing a 'graph-in graph-out' architecture, GNNs adeptly process input graphs, operating on node, edge, and global embeddings while preserving the inherent connectivity. At their core, GNNs iteratively propagate information through nodes and edges using convolutional-like operators, with two predominant approaches addressing this critical operation: spectral and spatial methods. Spectral approaches in GNNs involve representing graphs in the spectral domain, utilizing principles from graph signal processing [29] and defining convolutional operators through Fourier transforms. The Graph Convolutional Network (GCN) [30] is a widely used example of a spectral GNN architecture.

In contrast, spatial methods directly apply convolutions on the graph by aggregating information from neighboring nodes. Examples of spatial GNN architectures include GraphSAGE [31], particularly suitable for large-scale graphs, and the graph attention network (GAT) [32], which employs the attention mechanism to prioritize the most relevant information during aggregation. More recently, an enhanced version of the graph attention network, GATv2 [33], was introduced to address a limitation in GAT's attention mechanism. By incorporating dynamic attention, GATv2 achieves enhanced expressiveness, leading to superior performance across node-, link-, and graph-prediction tasks. In the context of the online 3D-BPP, which requires learning complex spatial relationships within the graph-structured environment, GATv2's ability to capture local spatial relationships makes it a compelling choice over spectral approaches like GCN. Its dynamic attention mechanisms are capable of learning intricate dependencies within the graph, enabling the model to adapt to varying structural configurations. By selecting GATv2, this study aims to harness these advanced capabilities to enhance the learning and decision-making processes in the online 3D bin packing scenario.

III. Method

A. Problem Formulation

The online 3D bin packing problem addressed in this research is framed as an output maximization problem. Given a container $\mathcal{C}$, of size (W, L, H), the objective is to assign a maximum-value subset $I \subseteq \mathcal{I}$ of items, where each item $i \in \mathcal{I}$ has arbitrary size (w_i, l_i, h_i) and cuboid shape, to the container. To model this problem effectively, it is formulated as a Markov decision process (MDP), a key framework for solving sequential decision-making problems. An MDP is typically expressed as a 5-tuple $(S, \mathcal{A}, \mathcal{P}, \rho, \gamma)$, where S is the set of observable states, $\mathcal{A}$ is the action set, $\mathcal{P}\colon S \times \mathcal{A} \times S \to [0, 1]$ determines the transition probabilities from a state-action pair to the following state, $\rho\colon S \times \mathcal{A} \to \mathbb{R}$ is the reward function and γ is the discount factor.

In the context of the online 3D-BPP, the environment state combines the current configuration of the bin and the characteristics of the incoming item, while the action specifies the placement and orientation of the item within the container. The objective is to devise a stochastic policy $\pi\colon S \to \mathcal{A}$, which maps states to probability distributions over potential actions, with $\pi(a|s)$ denoting the probability of selecting action a under state s. The policy aims to maximize the expected cumulative discounted reward, defined as

$$J_\pi = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\rho(s_t, a_t)\right]$$

where $\tau = (s_0, a_0, s_1, ...)$ is a trajectory sampled based on the policy π. In this formulation, the online 3D-BPP environment satisfies two fundamental MDP assumptions: it is fully observable and the next state solely depends on the current state and action. Additionally, the environment is assumed fully deterministic. Therefore, the transition probability function $\mathcal{P}$

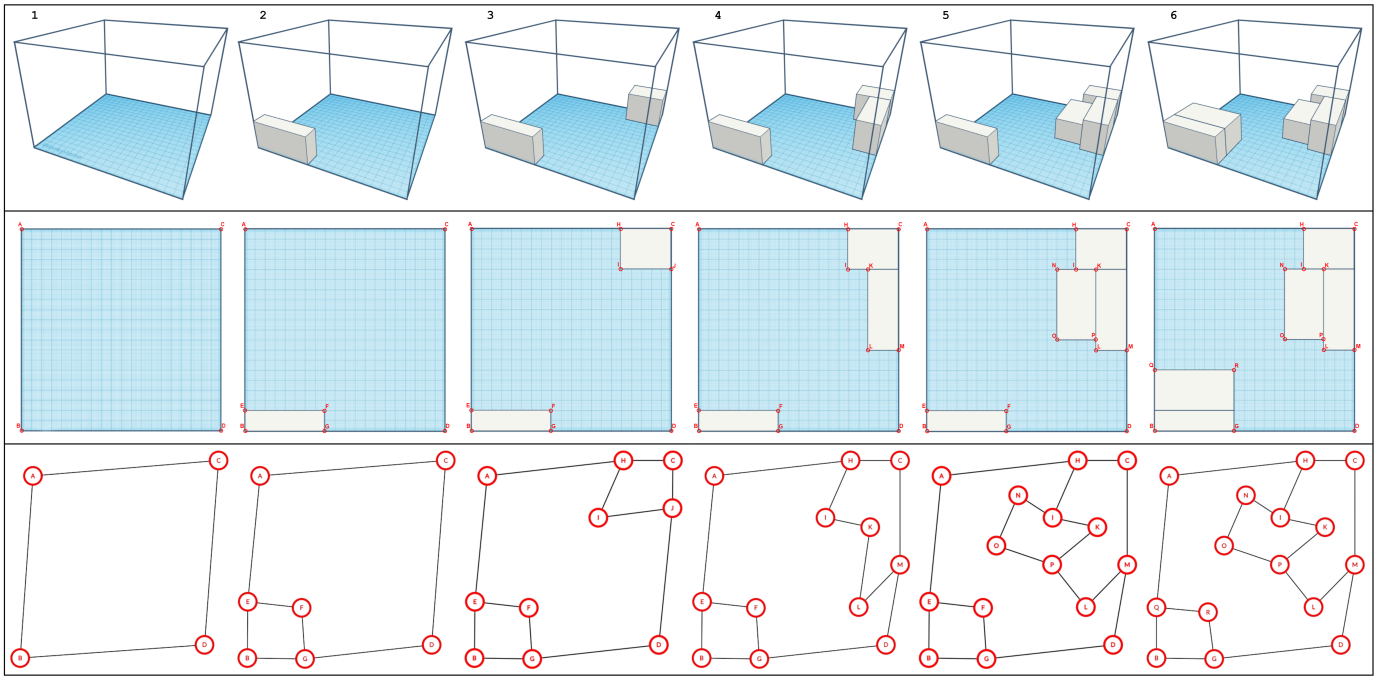


Fig. 2: llustration of the graph structure's evolution across six packing steps. At step 4, node J is removed as it becomes redundant due to the equal height of the adjacent items. Similarly, nodes E and F are discarded in the final step. In contrast, node K is preserved in step 5, as the height difference between items requires its inclusion for accurately describing the scene.

can be replaced with the transition function $\mathcal{T}: S \times \mathcal{A} \rightarrow S$ which maps a state-action pair (s, a) to a deterministic next state s'.

The traditional 3D-BPP imposes three basic geometric constraints: all items must lie entirely within the container, no item overlapping is allowed, and all items must be packed orthogonally to the walls of the container. To address the challenges faced in the real-world scenario, two additional constraints are introduced. The first restricts the possible orientations of items during the packing process. Specifically, each item is limited to a single vertical orientation while no restrictions are imposed on their horizontal direction. This is done to accommodate the use of a suction cup end-effector for the pick-and-place operations of items in the real-world testing scenario. Moreover, this constraint mirrors common practices observed in logistics when handling boxes marked with a "This way up!" sign, requiring that they be placed on a predefined surface. The second constraint ensures the structural integrity of the items and the safety of operators involved in loading/unloading processes by imposing vertical static stability. Implemented as a hard constraint within the simulation environment, it terminates training episodes whenever unstable stacking configurations are detected. In such instances, a penalty in the form of a negative reward is assigned to the agent.

B. State-Action Representation

Designing an appropriate state-action representation is a critical aspect of solving problems via reinforcement learning. It plays a significant role in determining how the agent perceives and interacts with the environment, thereby impacting its decision-making process and shaping the learning outcomes. In the context of the online 3D-BPP, the state environment centers around the current container configuration, complemented by the features of the incoming item.

The convention of representing the state of the container through a heightmap grid, wherein each cell represents a potential location for item placement, has posed challenges for RL-based methods in dealing with large action spaces. This limitation contributes to the observed lack of generalization capabilities in previous approaches, particularly evident in continuous environments where most methods struggle to perform effectively.

In light of these considerations, the goal was to develop an enhanced representation of the environment state. The chosen strategy involves encoding the current container configuration into a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where individual nodes $v \in \mathcal{V}$ identify corners within the packing scene, while edges $e \in \mathcal{E}$ represent the physical edges connecting these corners. In this novel approach, the packing scenario is described not only by the underlying graph structure but also through the feature vectors associated with each node v. Leveraging this graph-based representation enables the reinforcement learning agent to operate in continuous space, which serves as a significant advantage for optimizing packing strategies, particularly in real-world settings.

The action representation directly follows the structure of the environment state. As each node v in the graph structure corresponds to a corner in the scene, the action set $\mathcal{A}$ naturally extends from the node set $\mathcal{V}$. Given the constraint on item orientations during packing, each node in the graph structure offers a finite set of potential configurations for item placement. Specifically, as illustrated in Fig. 3, each node-item pair generates eight possible configurations linking the

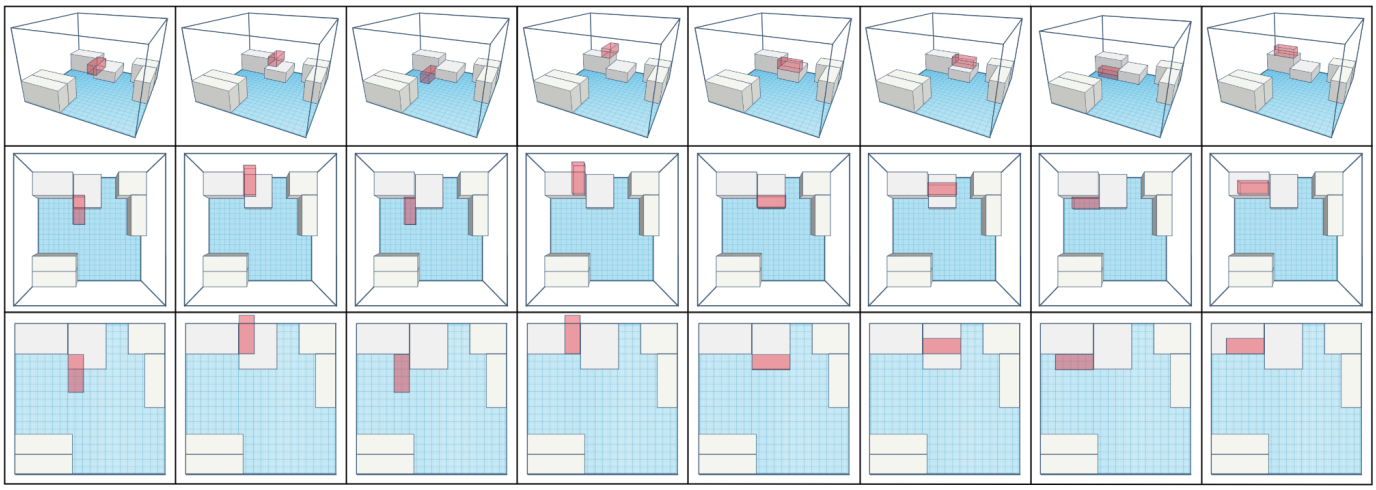


Fig. 3: Illustration displaying the eight configurations generated from a node-item pair, viewed from three different perspectives for a complete understanding of the resulting packing configurations. It clarifies that while not all actions lead to feasible configurations, each one connects the incoming item with a corner.

incoming item with a corner in the scene. It's worth noting that heuristic methods indicate that the corner set inherently includes configurations optimal for item placement.

The graph-based method employed to describe the packing scenario relies on node feature vectors, which offer significant versatility in representing the environment. Selecting suitable features for each node requires careful consideration of task-specific constraints and objectives. The chosen approach integrates a combination of high- and low-level features. Specifically, the feature vector associated with a node v, identifying a corner in the scene, is characterized by the following features:

- x, y representing the coordinates of the node v, with the bottom-left corner of the container serving as the reference point.
- $h_{TL}, h_{TR}, h_{BL}, h_{BR}$ denoting the heights of the top-left, top-right, bottom-left, and bottom-right corners, respectively, relative to node v. Each value is crucial for handling nodes shared by multiple items.
- d_T, d_B, d_R, d_L indicating the maximum distance along the positive and negative vertical, and positive and negative horizontal directions, that can be traveled from the node v without encountering a height change.
- $s_1, s_2, ..., s_8$ representing the percentage of the supported surface that an incoming item would have if paired with node v, considering the eight different configurations permitted by the orientation constraint.
- w_i, l_i, h_i denoting the dimensions of the incoming item i.

By incorporating these features into the node vectors, the representation effectively captures both the current configuration of the container $\mathcal{C}$ and the characteristics of the incoming item i, ensuring a comprehensive representation of the environment state. All spatial features, including coordinates, dimensions, heights, and distances, are normalized with respect to the size of the container $\mathcal{C}$ and its maximum allowed height, ensuring scale-invariance.

C. Network Architecture & Training Method

GATv2. The architecture selected for processing the graph-structured data obtained from the environment is the Graph

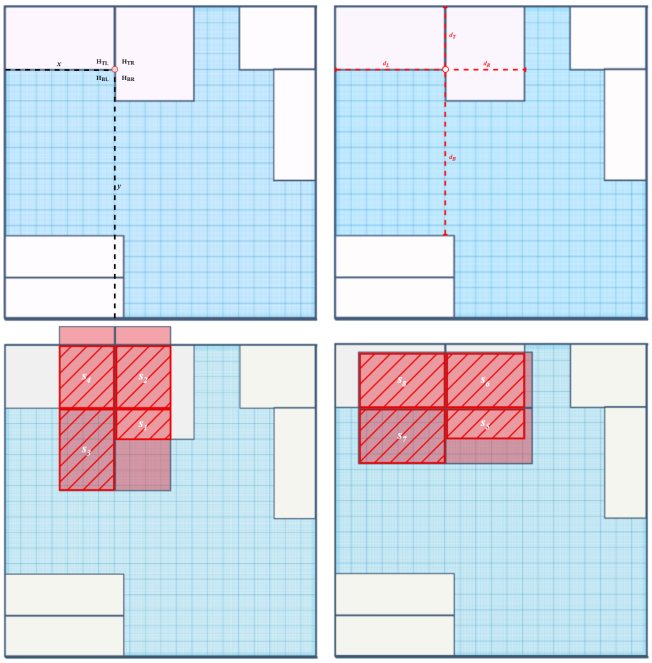


Fig. 4: Visualization of the low- and high-level features from a top view perspective. In the top-left section, the primitive spatial features are shown, while the top-right illustrates the 'distance to height change' metric. The lower section displays the supported areas for an incoming item across the eight configurations.

Attention Network version 2 (GATv2) [33]. Operating as a spatial method, GATv2 updates the node features by performing a convolution-style operation to the graph, aggregating information from neighboring nodes. As the name suggests, it employs the attention mechanism which has become a standard in sequence-based tasks such as machine translation and natural language understanding. This technique involves assigning different weights to input elements based on their relevance to the current context, thereby enabling the model to focus on critical information. As part of the update step, GATv2's attention mechanism computes coefficients (a_{ij}) to represent the relative importance of features from neighboring nodes (j) in the context of the current node (i). These coeffi-

cients commonly undergo a *softmax* function transformation, generating a probability distribution that reflects the influence of each neighbor on the current node's update. This process is iteratively applied across each layer of the network. In the proposed method, the GATv2 model is configured with 5 layers, utilizes 32 hidden features, and employs 4 attention heads.

Building upon the strengths of GATv2, the proposed model incorporates skip connections. These connections establish direct pathways between the input and output of consecutive layers to preserve and reuse valuable insights. This component effectively creates a shortcut within the network architecture, facilitating the efficient flow of information and enhancing the learning process for improved feature representation.

The problem is framed as a node-level task, with the model designed to predict an eight-dimensional output for each node. Each dimension corresponds to one of the possible orientations supported by a corner, thereby generating eight distinct scores per node. These scores indicate the suitability of each available configuration for item placement.

PPO. The Proximal Policy Optimization (PPO) [24] framework, a state-of-the-art on-policy reinforcement learning approach, is employed in this study. Inspired by the Trust Region Policy Optimization (TRPO), PPO introduces a clipping mechanism within the objective function. This mechanism acts on the probability ratio, denoted as $r = \frac{\pi(a|s)}{\pi_{old}(a|s)}$, constraining it within the interval [1 - ϵ, 1 + ϵ], effectively ensuring that the policy updates are conservative. PPO leverages the actor-critic framework, which involves the simultaneous learning of both a policy function (actor) and a value function (critic). The actor's role is to select actions to maximize expected rewards, while the critic estimates the long-term value of a given state-action pair. A common architectural design involves sharing parameters between the two functions to streamline the learning process. In the proposed approach, the actor and critic networks share three of their five layers, with an additional permutation-invariant max layer introduced to the critic. This layer is designed to derive a singular value from the final layer of the network.

During training, the goal is to minimize the following objective function with respect to the policy parameters θ:

$$L^{PPO}(\theta) = \mathbb{E}\left[L^{CLIP}(\theta) - \alpha L^{VF}(\theta) + \beta S[\pi_\theta]\right]$$

where α, β are coefficients, L^{CLIP} is the clipped policy loss, L^{VF} is the value function error term and S denotes the entropy of the policy π_θ. The optimization technique employed by PPO proves particularly effective for controlling stochastic policies, aligning well with the inherent demand for such policies in the online bin packing problem. Throughout the training process, the actor network's outputs, consisting of eight values per node, are transformed into a categorical probability distribution. The action for the subsequent step is then selected by sampling from this distribution, effectively balancing the exploration-exploitation trade-off. In the proposed method, the PPO model employs a clipping parameter $\epsilon = 0.2$, a value loss scaling factor $\alpha = 0.5$, an entropy scaling factor $\beta = 1e^{-2}$, an Adam learning rate $l_r = 2.5e^{-4}$, and a discount factor $\gamma = 0.99$.

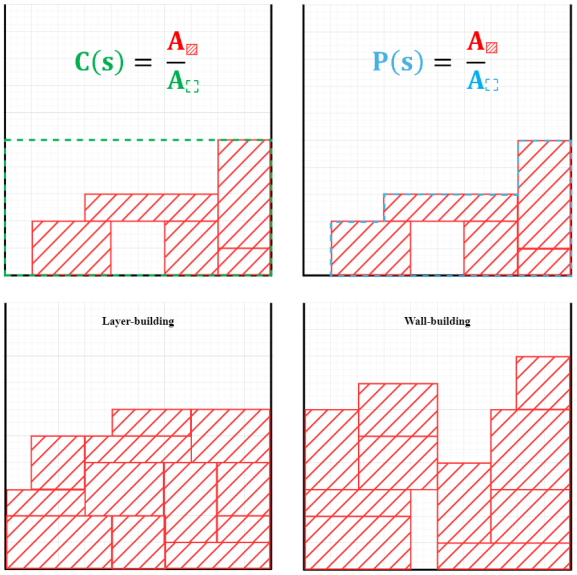


Fig. 5: Illustration of the computations for compactness (top-left) and pyramidality (top-right) through a two-dimensional example. The lower section displays a layer-building strategy on the left and a wall-building strategy on the right. While the first typically offers better volume utilization, it becomes less practical in scenarios with high heterogeneity among items' dimensions.

Reward function. Within the reinforcement learning framework for the 3D-BPP, the reward function plays a critical role in assessing packing quality, incentivizing stable and efficient stacking. Drawing inspiration from Hu et al.'s study [34], the proposed approach integrates two key aspects of a packing scenario. Formally, for a stacking configuration s, the reward function is defined as:

$$\rho(x) = \begin{cases} \alpha_c \cdot C(s) + \alpha_p \cdot P(s), & \text{if } s \text{ is feasible.} \\ -1, & \text{otherwise.} \end{cases}$$

where P and C, as shown in Fig. 5, are the pyramidality and compactness of the cargo, both scaled by their respective coefficient. The pyramidality (P) is computed as the ratio of the total volume of packed items to the volume of the region obtained by projecting all top faces toward the bottom of the container. On the other hand, the compactness (C) is determined by the ratio of the total volume of packed items to the volume of the cuboid area defined by the maximum height of the items and the base of the container. Intuitively, both metrics favour tightly packed items, yet $C = 1$ only when all items entirely occupy the container up to a certain height. As a result, altering these coefficients, particularly their ratio, profoundly influences the strategies adopted by the agent and consequently affects its performance. Specifically, prioritizing compactness is likely to encourage a layer-building policy, whereas emphasizing pyramidality tends to promote a wall-building policy. Finally, the reward is assigned a negative value, acting as a penalty, whenever the agent achieves an unstable stacking configuration or when an item is placed outside the boundaries of the container.

D. Training Environment

The training environment for the 3D-BPP was developed within Isaac Gym [35], a high-performance learning platform

designed for training policies across a broad spectrum of robotics tasks. This setup leverages Isaac Gym's capabilities to execute both physics simulations and neural network policy training on the GPU, enabling efficient data transfer from physics buffers to PyTorch tensors. A Python-based Tensor API allows direct access to the data buffers, wrapping them into PyTorch tensors and minimizing potential CPU bottlenecks. As a result, the agent's policy can interact with the simulated environment by accessing and manipulating physics data through these tensors.

In the custom-designed 3D-BPP environment, multiple scenarios are executed in parallel. As shown in Fig 6, each scenario features a container, modeled as a flat surface, and a set of cuboids, representing the items, that exceed the container's capacity. Once the scene is initialized, a dynamic queue of items is formed for each scenario, advancing iteratively as the agent strategically places each item until the end of the episode is reached. With the aim of enhancing the model's ability to generalize to real-world scenarios without additional training, efforts were directed toward bridging the reality gap. Domain randomization, a proven strategy in narrowing this gap, suggests that introducing substantial variability in simulation facilitates effective generalization of models to the real world. Consequently, every scenario features a unique set of items, with their sequence randomized at the beginning of each training episode, aiming to mimic the unpredictability and diversity of real-world conditions.

Datasets. The approach was evaluated using two distinct datasets, drawing inspiration from existing literature to assess its performance against previous attempts. It's important to highlight that, consistent with prior research, the container $\mathcal{C}$, of size (W, L, H), is assumed to have a square base, implying $L = W$. The items' dimensions in both datasets are constrained as follows:

$$\frac{L}{10} \leq w_i \leq \frac{L}{2}, \quad \frac{L}{10} \leq l_i \leq \frac{L}{2}, \quad \frac{H}{10} \leq h_i \leq \frac{H}{2} \quad \forall\, i \in \mathcal{I}.$$

The first test set, referred to as the discrete dataset, is modeled after the RS dataset introduced by Zhao et al. [15]. In this case, the resolution is fixed at $L = W = 10$, generating 75 pre-defined item sizes ($|\mathcal{I}| = 75$). Specifically, each item's dimensions (w_i, l_i, h_i) are independently and uniformly sampled from the set $\{1, 2, ..., 5\}$. The subsets provided to each scenario are constructed by randomly selecting items from the full set $\mathcal{I}$. It's worth noting that random sampling introduces uncertainty regarding the optimality of a sequence, however this approach mirrors more closely the variability encountered in practical applications. The utilization of this dataset intentionally aligns with established benchmarks, enabling a comprehensive evaluation of how the proposed approach compares to previous methods.

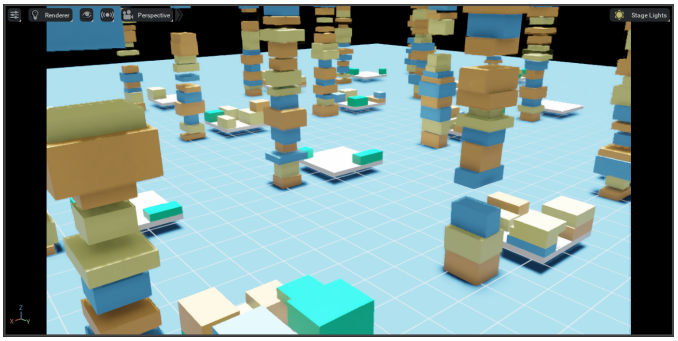


Fig. 6: Snapshot of the simulation environment during training.

The second test set, referred to as the continuous dataset, is generated by sampling the width and length of the items (w_i, l_i) from a continuous uniform distribution within the range $[\frac{L}{10}, \frac{L}{2}]$. However, similarly to the first dataset, the height dimension h_i is uniformly sampled from the discrete set $\{1, 2, ..., 5\}$. This approach aligns with the most commonly employed continuous dataset found in the literature, enabling the formation of layers in the packing configuration. The continuous dataset was selected to highlight the progress and potential of the proposed approach, demonstrating its capability to handle the intricacies of real-world environments with enhanced efficiency and adaptability.

Stability estimation. Determining the feasibility of a given environment state is a crucial step in the reinforcement learning pipeline. In the context of the 3D-BPP, considering the constraint previously discussed, this requires assessing the vertical stability of a packing arrangement. Various approaches have been explored in the literature, typically involving the definition of hard constraints, to ensure vertical stability. For instance, Schuster et al. [36] asserts that an item is vertically stable if at least two opposite edges of its base are supported from below. Alternatively, Nascimento et al. [4] introduces the *support factor approach* for modeling vertical stability. This method imposes that a minimum fraction $\phi \in (0, 1]$ of an item's bottom surface must be supported to achieve stability.

The proposed approach combines Schuster et al.'s stability criteria with the advanced capabilities of the physics simulator to determine the environment state's feasibility. Initially, it examines the stability of each newly placed item based on the *two-edges* criteria. Following this preliminary check, the stability of the entire packing configuration is assessed. This evaluation leverages the physics simulator's ability to aggregate all rigid body states into a unified state tensor. Specifically, a packing arrangement is identified as vertically stable on the condition that, after a predefined number of simulation steps, all items within the container maintain perfect parallelism to the base and are completely stationary, exhibiting no linear or angular velocity.

IV. Results

To evaluate the performance of the proposed approach, a comparative analysis was conducted with seven established methods. These baselines can be categorized into two groups. The first includes heuristic methods, such as the conventional Deepest Bottom Left with Fill (DBLF) [11] and Extreme Point Best Fit Decreasing (EP-BFD) [10], along with the more recent OnlineBPH method [37]. The second group consists of four learning-based methods [7], [15], [18], [38] that leverage the DRL framework to learn a bin packing policy. The experiments are performed using the discrete and continuous datasets outlined in the previous section. Following

the conventions of existing studies, the volume utilization and the number of packed items serve as evaluation criteria for the bin packing policies. The experimental setup featured 32 parallel scenarios, training both discrete and continuous policies over 5,000 iterations, each comprising 128 simulation steps.

A. Performance in the discrete scenario

Method	Vol. Utilization	Num. Items
Online BPH	52.1 ± 0.142	12.98
DBLF	60.5 ± 0.093	14.81
EP-BFD	63.9 ± 0.104	15.64
LSAH	52.6 ± 0.110	13.05
Zhao, 2021	70.4 ± 0.105	17.58
Zhao, 2022	73.1 ± 0.072	17.91
Pack-Heu-RL	73.8 ± 0.103	18.14
Proposed	**73.6 ± 0.126**	**18.09**

TABLE I: Results in the discrete setting

After experimentation, it was determined that setting both α_P and α_C to 0.5 was optimal for training the model in the discrete scenario. This balance was observed to promote stacking strategies focused on layer building from the initial training phases, aligning with the characteristics of the discrete scenario which involves weakly heterogeneous items. Table I presents the experimental results, showcasing the superiority of learning-based methods over heuristic approaches. The proposed method outperforms the majority of baselines in terms of volume utilization and the number of packed items, indicating its state-of-the-art performance in the discrete bin packing scenario. This is especially significant given that the model is designed for continuous environments. Despite a marginally higher variance, the results are comparable to those obtained by Zhao et al. [38] and Yang et al. [18], suggesting that the proposed approach is competitive with established benchmarks.

B. Performance in the continuous scenario

Method	Vol. Utilization	Num. Items
Online BPH	43.9 ± 0.119	10.77
LSAH	48.3 ± 0.110	11.86
Zhao, 2022	65.4 ± 0.057	16.12
Proposed	**69.2 ± 0.083**	**17.23**

TABLE II: Results in the continuous setting

In the continuous dataset evaluation, only a subset of baseline methods is considered, as many solutions were not designed to operate in this context. Specifically, while Zhao et al.'s solution is inherently compatible with this data type, LSAH, OnlineBPH, and DBLF required adaptations to function in this scenario. It is worth noting that the proposed approach stands out as the only method capable of functioning in continuous space without relying on heuristics. In contrast, other learning-based approaches, such as Zhao (2022) [38] and LSAH [7] employ Empty Maximal Spaces (EMS) and Least Surface Area (LSA) heuristics, respectively, to generate placement locations.

For this scenario, the reward coefficients were adjusted to $\alpha_P = 0.7$ and $\alpha_C = 0.3$ to discourage a layer-building strategy due to high item heterogeneity. The detailed results, as shown in Table II, reveal that the proposed method outperforms all competing approaches, with volume utilization exceeding 69%. Such a result indicates that the solution is capable of packing, on average, over one additional item per container than its counterparts, demonstrating its superior performance in the continuous bin packing scenario.

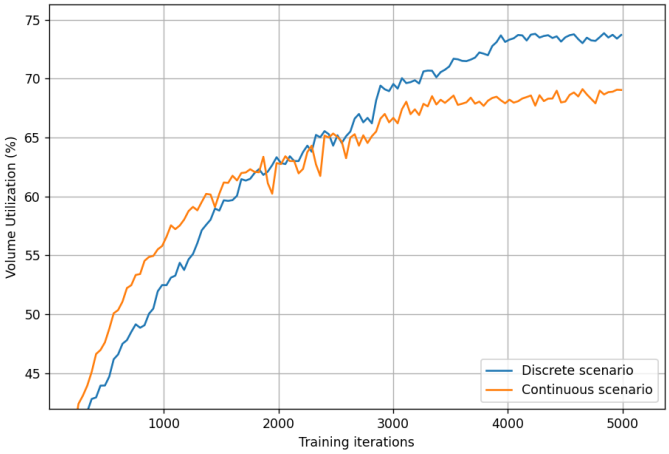


Fig. 7: Learning curves of the models trained in the discrete and continuous setting.

V. Discussion

The experimental results across both the discrete and continuous bin packing scenarios clearly demonstrate the state-of-the-art performance achieved by the proposed approach. In the complex continuous setting, where item dimensions exhibit high heterogeneity, the method obtained an average volume utilization exceeding 69%, outperforming the packing capabilities typically observed in humans. For reference, Zhao et al. [38] evaluated over 50 human participants on the same continuous dataset and reported an average volume utilization of only 56.3%. Furthermore, the versatility of the approach is highlighted by its impressive volume utilization of 69.7% when the model trained on the continuous data was evaluated in the discrete setting. This cross-scenario generalization capability likely stems from the method's inherent operation in continuous spaces without relying on heuristics.

Analyzing the learning curves in Fig. 7 offers insights into the reward shaping and training dynamics. In the early stages, the agent trained in the continuous setting outperformed its discrete counterpart. This is likely attributed to the tailored reward coefficients, which encouraged the continuous scenario agent to adopt a wall-building strategy. This strategy is relatively simpler to learn and can lead to reasonable volume utilization rather quickly. In contrast, the discrete agent aimed to learn the more complex layer-building approach, facing a steeper learning curve during the early training phases. However, once this strategy was mastered, the discrete scenario agent rapidly

surpassed the continuous agent's performance, reflecting the ability to leverage layer-building for improved space utilization with weakly heterogeneous items.

VI. Real-world evaluation

To transition the simulated policy into a real-world context, a corresponding online bin packing scenario was recreated. A robotic system, equipped with a Yaskawa robotic arm and two RGB-D sensors, was developed to assess the actual packing performance. The container for packing items is defined as a $650 \times 650 \times 600\text{mm}^3$ 3D region in front of the manipulator, situated directly across from the the picking area. This setup is shown in Fig. 1.

The main challenge encountered involved constructing the environment state based on the current container configuration. The novel representation removes the need to partition the base area into cells and alleviates concerns about their sizes, setting it apart from previous methodologies. Instead, Fizyr's detection software was employed to generate bounding boxes for items placed within the container. These bounding boxes serve as the foundation for constructing a three-dimensional representation of the scene, used to determine the graph structure and compute the features. The picking software is also responsible for the real-time estimation of the item orientation and dimension during the picking process. Ensuring the accuracy of this estimation is crucial, as any error at this stage propagates forward and introduces discrepancies between the actual and perceived placement configurations.

An additional challenge emerged during implementation when it was observed that the trained agent displayed a tendency to distribute items uniformly on the base plane of the bin. In practice, the robot arm was located beside the bin rather than above it, posing a risk of colliding with items in close proximity to the manipulator. To address this issue, following insights from [39], slight adjustments were made to the packing strategy, ensuring the start of the stacking process from the farthest corners. Moreover, the addition of strategically placed waypoints further reduced collision risks by guiding the arm's movements with precision. In real-world tests, the method demonstrated robust performance, frequently achieving volume utilization rates ranging from 65% to 75%. The significance of this result is amplified by the fact that the agent, trained on the continuous dataset, was not explicitly fine-tuned for the real-world dataset used during these tests.

VII. Future work

There is considerable interest in extending the proposed method to address more complex variants of the bin packing problem. Such extensions may include scenarios where multiple items are simultaneously observable within a buffer zone, or those without constraints on item orientation. In the former, the model's feature vectors will require reconfiguration to capture the characteristics of multiple items. For the latter, an expansion of the action space is necessary to accommodate all potential placement configurations generated from node-action pairs. Additionally, adapting the model to enable the packing of irregularly shaped items [40] represents another interesting direction for research.

VIII. Conclusion

This work addressed the challenge of efficiently solving the online 3D bin packing problem, a task with profound implications for optimizing logistics and container loading operations in real-world applications. The proposed solution formulated the problem as a Markov decision process and applied the reinforcement learning framework. A key innovation is the introduction of a graph-based representation for the packing scenario, which facilitates the efficient encoding of task-specific high-level features. In addition, this representation enables the reinforcement learning agent to learn an optimal packing policy while operating inherently in the continuous domain. Extensive evaluations across both simulated environments and a real-world setting validated the versatility and effectiveness of the proposed method.

Acknowledgment

I am truly grateful to Cosimo Della Santina, my thesis supervisor, for his guidance and support throughout this project. A heartfelt thank you to Ronald Poelman, my supervisor at Fizyr, for his insightful feedback and mentorship. Additionally, I would like to thank the entire team at Fizyr for providing a stimulating environment for my research.

References

[1] "Global parcel shipping volume between 2013 and 2027," Statista. (2022), [Online]. Available: https://www.statista.com/statistics/1139910/parcel-shipping-volume-worldwide/.

[2] H. Dyckhoff, "A typology of cutting and packing problems," *European journal of operational research*, vol. 44, no. 2, pp. 145–159, 1990.

[3] S. Ali, A. G. Ramos, M. A. Carravilla, and J. F. Oliveira, "On-line three-dimensional packing problems: A review of off-line and on-line solution approaches," *Computers & Industrial Engineering*, vol. 168, p. 108 122, 2022.

[4] O. X. do Nascimento, T. A. de Queiroz, and L. Junqueira, "Practical constraints in the container loading problem: Comprehensive formulations and exact algorithm," *Computers & Operations Research*, vol. 128, p. 105 186, 2021.

[5] F. Parreño, R. Alvarez-Valdés, J. M. Tamarit, and J. F. Oliveira, "A maximal-space algorithm for the container loading problem," *INFORMS Journal on Computing*, vol. 20, no. 3, pp. 412–422, 2008.

[6] S. Liu, W. Tan, Z. Xu, and X. Liu, "A tree search algorithm for the container loading problem," *Computers & Industrial Engineering*, vol. 75, pp. 20–30, 2014.

[7] H. Hu, X. Zhang, X. Yan, L. Wang, and Y. Xu, "Solving a new 3d bin packing problem with deep reinforcement learning method," *arXiv preprint arXiv:1708.05930*, 2017.

[8] N. Bansal and A. Khan, "Improved approximation algorithm for two-dimensional bin packing," in *Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms*, SIAM, 2014, pp. 13–25.

[9] S. Martello, D. Pisinger, and D. Vigo, "The three-dimensional bin packing problem," *Operations research*, vol. 48, no. 2, pp. 256–267, 2000.

[10] T. G. Crainic, G. Perboli, and R. Tadei, "Extreme point-based heuristics for three-dimensional bin packing," *Informs Journal on computing*, vol. 20, no. 3, pp. 368–384, 2008.

[11] K. Karabulut and M. M. İnceoğlu, "A hybrid genetic algorithm for packing in 3d with deepest bottom left with fill method," in *International Conference on Advances in Information Systems*, Springer, 2004, pp. 441–450.

[12] W Fernandez de La Vega and G. S. Lueker, "Bin packing can be solved within 1+ ε in linear time," *Combinatorica*, vol. 1, no. 4, pp. 349–355, 1981.

[13] D. S. Johnson, "Fast algorithms for bin packing," *Journal of Computer and System Sciences*, vol. 8, no. 3, pp. 272–314, 1974.

[14] O. Kundu, S. Dutta, and S. Kumar, "Deep-pack: A vision-based 2d online bin packing algorithm with deep reinforcement learning," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, IEEE, 2019, pp. 1–7.

[15] H. Zhao, Q. She, C. Zhu, Y. Yang, and K. Xu, "Online 3d bin packing with constrained deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 741–749.

[16] Z. Yang, S. Yang, S. Song, *et al.*, "Packerbot: Variable-sized product packing with heuristic deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 5002–5008.

[17] R. Verma, A. Singhal, H. Khadilkar, *et al.*, "A generalized reinforcement learning algorithm for online 3d bin-packing," *arXiv preprint arXiv:2007.00463*, 2020.

[18] S. Yang, S. Song, S. Chu, *et al.*, "Heuristics integrated deep reinforcement learning for online 3d bin packing," *IEEE Transactions on Automation Science and Engineering*, 2023.

[19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[21] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[22] V. Mnih, A. P. Badia, M. Mirza, *et al.*, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, PMLR, 2016, pp. 1928–1937.

[23] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[25] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," *Advances in neural information processing systems*, vol. 30, 2017.

[26] T. Haarnoja, A. Zhou, P. Abbeel, and 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.

[27] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[28] J. Zhou, G. Cui, S. Hu, *et al.*, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[29] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[30] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[31] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[32] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[33] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *arXiv preprint arXiv:2105.14491*, 2021.

[34] R. Hu, J. Xu, B. Chen, M. Gong, H. Zhang, and H. Huang, "Tap-net: Transport-and-pack using reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.

[35] V. Makoviychuk, L. Wawrzyniak, Y. Guo, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[36] M. Schuster, R. Bormann, D. Steidl, S. Reynolds-Haertle, and M. Stilman, "Stable stacking for the distributor's pallet packing problem," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 3646–3651.

[37] C. T. Ha, T. T. Nguyen, L. T. Bui, and R. Wang, "An online packing heuristic for the three-dimensional container loading problem in dynamic environments and the physical internet," in *Applications of Evolutionary Computation: 20th European Conference, EvoApplications 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings, Part II 20*, Springer, 2017, pp. 140–155.

[38] H. Zhao, C. Zhu, X. Xu, H. Huang, and K. Xu, "Learning practically feasible policies for online 3d bin packing," *Science China Information Sciences*, vol. 65, no. 1, p. 112 105, 2022.
[39] S. Martello, D. Pisinger, D. Vigo, E. D. Boef, and J. Korst, "Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem," *ACM Transactions on Mathematical Software (TOMS)*, vol. 33, no. 1, 7–es, 2007.
[40] F. Wang and K. Hauser, "Robot packing with known items and nondeterministic arrival order," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 1901–1915, 2020.

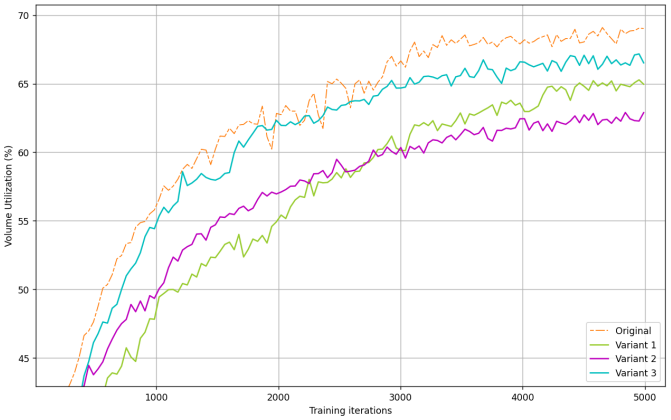


Fig. 9: Learning curves of the four models, trained in the continuous setting.

APPENDIX A
RESULTS AND DISCUSSION OF APPROACH VARIATIONS

This section delves into the impact of specific hyperparameters on the performance of the solution, with a particular focus on reward coefficients and node features. To highlight their effects, the results from three distinct variants of the proposed approach are presented. Each variant involves a modification to a single element, leaving the rest of the system unchanged, to isolate and examine the impact of that component on overall performance. These evaluations were conducted within the continuous scenario, which more clearly delineates the disparities in performance due to its complexity.

The first variant adjusts the reward coefficients α_P and α_C to 0.5, mirroring the configuration used in the discrete scenario. This balance promotes a layer-building strategy, encouraging the sequential horizontal placement of items to form layers, a technique typically suited to datasets featuring low heterogeneity in items' dimensions.

In contrast, the second variant, with reward coefficients set to $\alpha_P = 0.85$ and $\alpha_C = 0.15$, primarily favors a wall-building policy. This strategy involves stacking items vertically to create column-like structures or walls.

The third variant maintains the original reward coefficients ($\alpha_P = 0.7$, $\alpha_C = 0.3$) but modifies the node feature vectors. Instead of employing the 'distance to height change' metric, it introduces a 'max supported square size' feature. This feature, as shown in Fig 8, consists of four values indicating the maximum size of a square-based item that, when paired with the node in the four configurations (top-left, top-right, bottom-left, bottom-right), would have a fully supported base. This metric considers only four configurations since rotating square-based items around the vertical axis would not affect the outcome. Despite its seemingly arbitrary nature, this feature provides crucial spatial insights, enabling informed decision-making by the agent.

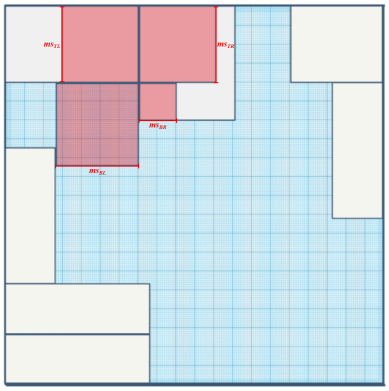


Fig. 8: Visualization of the 'max supported square size' feature from a top-view.

Fig. 9 compares the performance of the three variants against the original implementation. While Variants 1 and 3 showcased state-of-the-art performance in terms of volume utilization, Variant 2 was notably less effective. The three models achieved, respectively, an average volume utilization of 65.4%, 62.3%, and 66.8%. The discussion will initially focus on the first two variants, as their adjustments represent contrasting approaches.

Variant 2, which predominantly adopted a wall-building strategy, outperformed its counterpart in the initial stages of training. However, its progress plateaued, ultimately resulting in a policy that packed, on average, 2.3 fewer items per container than the original solution. In contrast, Variant 1, employing a layer-building strategy, achieved volume utilization on par with the current state-of-the-art method. Nonetheless, it failed to ensure vertical stability as training concluded. This is showcased in Fig. 11, which presents a graph of 'Instability-Induced Episode Ends' (IIEE). This metric, representing the percentage of training episodes terminated due to unstable configurations, was not previously discussed as vertical stability is treated as an absolute requirement, with the expectation for solutions to obtain a final IIEE of zero. Indeed, during the final thousand iterations, the original model, along with Variants 2 and 3, approached this value. Achieving a consistently zero IIEE during training proved challenging due to the probabilistic nature of action selection. However, all three methods effectively overcome this issue during testing by simply selecting the highest-scoring action.

On the other hand, Variant 1 interestingly displays an increase in IIEE during the later stages of training, aiming to achieve higher long-term rewards. This is highlighted by a corresponding rise in volume utilization. To mitigate the risk of unstable configurations in real-world applications, introducing

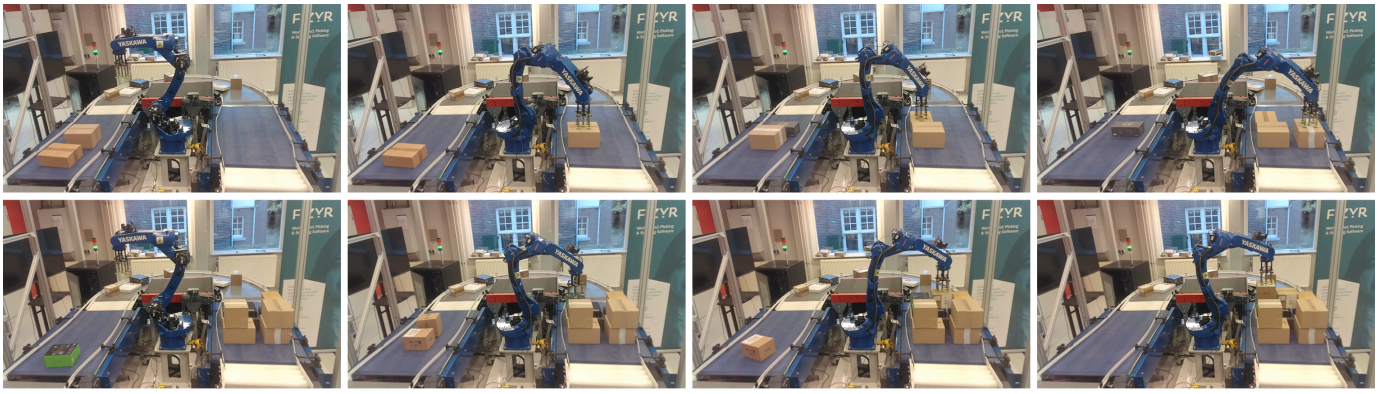


Fig. 10: Illustration capturing the initial and final stages of the manipulator implementing the trained policy in the real-world setting.

Fig. 11: Percentage of Instability-Induced Episode Ends for the four models trained within the continuous setting.

a stability mask to filter unstable actions through heuristic criteria could be considered, though it may impact overall performance. In addition, increasing the penalty for reaching an unstable configuration might improve the balance with the rewards, leading to better training outcomes.

Finally, Variant 3 achieves similar performance to the original implementation, differing by only a few percentage points in average volume utilization. Given that both metrics represent similar spatial characteristics, this underscores the influence of node features on the effectiveness of the policy. Moreover, it suggests that a further exploration of additional features could provide greater insights into the problem and potentially surpass the original method's performance, highlighting the versatility of the proposed approach.

Appendix B
Real-World Implementation

Fig. 10 presents snapshots illustrating the real-world implementation of the approach. Crucial to this transition is the use of Fizyr's detection software, which provides three-dimensional bounding boxes for items within both the picking and packing areas. In the first phase, the pick is executed based on the estimated position, while the dimensions contribute to the environment state, and the orientation is preserved for later calculating the precise placement pose. When multiple items are present in the picking area, the adopted convention is to consider the tallest item the only observable by the agent.

During the packing phase, the bounding boxes enable the creation of a three-dimensional simulated model of the current container configuration. This simulation is crucial for generating the graph structure and node features. The graph-structured representation, including the incoming item's normalized dimensions, is fed to the pre-trained model, which evaluates all possible placement configurations. The algorithm then selects the highest scoring one and, employing the simulated representation as reference and considering the orginal estimated orientation of the item, converts it into a three-dimensional pose for the manipulator to place the item. The accuracy of the detection software was remarkable, ensuring high precision during item placements. This was particularly impressive considering that any inaccuracy from the picking phase would be propagated forward, resulting in mismatches between actual and expected configurations.