Deep-Pack: A Vision-Based 2D Online Bin Packing Algorithm with Deep Reinforcement Learning

Olyvia Kundu, Samrat Dutta and Swagat Kumar

Abstract— **This paper looks into the problem of online 2D bin packing where the objective is to place an incoming object in a way so as to maximize the overall packing density inside the bin. Unlike off-line methods, the online methods do not make use of information about the sequence of future objects that are going to arrive and hence, are comparatively difficult to solve. A deep reinforcement learning framework based on Double DQN is proposed to solve this problem that takes an image showing the current state of the bin as input and gives out the pixel location where the incoming object needs to be placed as the output. The reward function is defined in such a way so that the system learns to place an incoming object adjacent to the already placed items so that the maximum grouped empty area is retained for future placement. The resulting approach is shown to outperform existing state-of-the-art-method for 2D online packing and can easily be extended to 3D online bin packing problems.**

I. Introduction

Bin Packing Problem (BPP) is a classical combinatorial optimization problem whose solution is widely sought after by the logistic and the packing industry constantly striving to reduce their shipping or packing cost, a major component of which is associated with the volume of the space utilized to pack a given set of objects. In general, the bin packing problem (BPP) is known to be NP-Hard and has attracted a significant amount of research interest since the 1970s. There are several variants of this problem. Depending on the dimensions being used for maximizing the space utilization, it could be termed as 1D, 2D or 3D BPP problems. Another classification can be made on the basis of whether one has the knowledge of all the objects dimension in advance or not. The former one is more appropriately known as *offline* methods [1][2] [3] which is more common and assume the knowledge of all object dimensions apriori and aim at finding an appropriate sequence as well as orientations (or pose) of these objects required for maximizing the space utilization or minimizing the wasted space inside a given bin. The other class of methods, termed as *online* methods [4] [5], do not have the knowledge of all objects to be packed in advance and hence, can only achieve local optimization for packing efficiency. This later class of methods is more challenging and is currently getting more traction in the e-commerce warehouse industry which has to respond almost in real-time with the varying consumer demands. Some variants of on-line BPP make use of 'look ahead' [6] or batch size [7] information to improve packing efficiency. A number of approaches have been proposed in the literature to solve both of these two categories of BPP. Since it is difficult to find exact solutions for these problems, most of the research has been focussed on finding approximate and heuristic algorithms [8] [9] [10] [11] including several variants of genetic or evolution algorithms [12] [13]. Many of these algorithms do not generalize well with varying situations leading to limited application to real-world problems. Recent advances in AI, particularly Deep Reinforcement learning (DRL) has been shown to provide promising results in solving several combinatorial problems [14] [15], in some case, outperforming human-level competence, e.g. Alpha Go [16]. This has encouraged a few researchers to apply DRL to solve bin packing problems as well. For instance, the authors in [17] [18] use a Pointer Net (Ptr-Net) combined with a policy-based reinforcement learning algorithm to solve an off-line 3D Bin Packing problem.

In this paper, we propose a Deep Reinforcement Learning based framework to solve a vision-based 2D online Bin Packing Problem (O2D-BPP). The proposed framework takes an input image showing the current state of the bin and gives the pixel location as output where the next object will be placed. The items to be placed inside the bin arrive in sequence on a conveyor and the algorithm does not have any information about the future objects that are going to arrive. The vision-based methods have the advantage of being tolerant to sensor noises as these methods do not require the knowledge of precise object dimensions. In addition, it is possible to deal with more irregular shapes without having to approximate them with the nearest rectangles or polygons as demonstrated in [19]. Currently, we are presenting an initial version where the objects to be placed are assumed to be rectangular and the incoming objects are to be placed without any rotation. It is not a restriction as a robotic system can be deployed to re-orient the object to its desired pose before placement in the bin. If the item cannot be placed in the bin for lack of space it has to be discarded immediately as it comes. The objective of the algorithm is to achieve maximum packing efficiency for a given bin by minimizing the empty spaces which is same as minimizing the total number of bins required for packing a given set of items. The proposed DRL framework makes use of a Double DQN [20] architecture where a CNN is used to approximate action-value function $Q(s, a)$. The DDQN algorithm makes use of a target network and experience replay to improve its performance. The reward functions are designed so that the items are placed adjacent to existing objects in the bin, thereby minimizing the wasted space and maximizing the occupied surface area. The resulting framework is shown to

Corresponding author Email IDs: {olyvia.kundu,d.samrat, swagat.kumar}@tcs.com

provide higher packing density compared to other state-of-the-art methods in this field. The key contributions in this paper are as follows:

- Although 2D online packing is an old problem and people have addressed it with various algorithms, this is the first attempt to solve this complex problem with Deep Reinforcement Learning. This end-to-end unsupervised approach does not use heuristic methods, therefore, the scalability of usage of the proposed method increases vastly.
- A novel State definition by concatenating the bin and item representation. This definition is effectively a transformation of the image having information about the bin and the container. This type of State representation together with CNN can deal with real-life scenarios, therefore, suitable to be applied on autonomous systems and it also could be adapted to other shapes as well.
- A novel Reward structure which positively reinforces compact placements of items in the bin so that maximum empty space is available in the bin for placement of the future items. Such a reward function enables the system to learn to place items efficiently and hence results in a very high packing efficiency at the end.

II. Related Work

A huge number of methods exists in the literature for packing problems [21] which can be broadly classified into categories, offline and online. Genetic algorithm [1], local search [2], GRASP [3], ant colony [2] are some of the well-known methods for offline packing. Online packing is also studied extensively in the literature A popular strategy to approach this problem is to use heuristics based methods [8] to find out optimal position for placement. In this category, fitting methods such as First Fit (FF) [22], Next Fit (NF) [23], Best fit (BF) [24] are the oldest approach to traditional online BPP. In First fit, the item is placed the first bin that has space to accommodate it. On the contradictory, Best fit [24] scans all opened bins at that time point and place the item in that bin, where space is available for that item and minimum space is left after placement. Shelf Fit [25] is another common method where the bin space is divided into shelves and the incoming item is added based on FF or BF or NF rule. The paper [26] implements skyline method based on least waste map. The main problem of the heuristic-based methods is that they are not easily adaptable to different scenarios as required by automation systems. Although this difficulty is addressed by hyperheuristic algorithms [27] which is basically heuristics to choose and combine the best heuristics for a given scenario, but the dependency still exists.

Nowadays Deep Learning is very popular as it offers lots of flexibility in terms of computation time (in testing phase), handling real environments, modelling a generalized network for various scenarios. Sequence-to-sequence model by [14], Pointer network [28] motivates the use of deep learning in the combinatorial optimization problems. The paper [18] have combined a heuristic-based approach with DRL to pack all items using minimal surface area. The heuristics were used to determine the box placement. Later they[29] avoided the heuristics by training a separate supervised model. The DRL techniques in the past works are focused on offline BPP, not for online variants. As evident from the literature, none of the methods is suitable to work with robotics systems and to the best of our knowledge, this is the first attempt to use DRL based model to solve 2D online packing problem.

III. Proposed Method

In this problem statement, we consider that only one rectangular bin is available and we have to find the optimized placement of the incoming rectangular items so that the maximum surface area is occupied. Another constraint we have considered is that each item has to be placed in the bin as it comes, $i.e$ without rotation. If the item cannot be placed in the bin for lack of space it has to be discarded immediately and will never be considered again for placement. Consider one rectangular item R_t is coming in a production belt at each time point $t = \{1, 2, \ldots, T\}$ which has to be placed inside a rectangular bin $W \times H$. The t^{th} item R_t has width I_{w_t} and height I_{h_t} where, $0 < I_{w_t} \leq W$ and $0 < I_{h_t} \leq H$. In this case, at time t, all the futuristic rectangular items are unknown. The main objective of this problem is to maximize the packing efficiency (**PE**, defined in equation 1) at the end of the sequence.

$$\text{PE} = \frac{\text{total surface area used in the bin}}{\text{total surface area of the bin}} \qquad (1)$$

As discussed in the Introduction section, this problem has applications in robotics, we consider an agent (i.e a robot) is learning to place rectangular items sequentially in the bin at every time point. For learning the agent uses a deep reinforcement learning technique. This learning paradigm has been modelled mathematically as a Markov Decision Processes or MDPs. An MDP is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is a finite set of states that the system can be in, $\mathcal{A}$ is the possible number of actions taken by the agent. The transition function $\mathcal{P}$ is defined as $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$. The reward function $\mathcal{R}$ is defined as $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. For this set-up of the system, we define each of $\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$ and $\mathcal{R}$ in the following subsections.

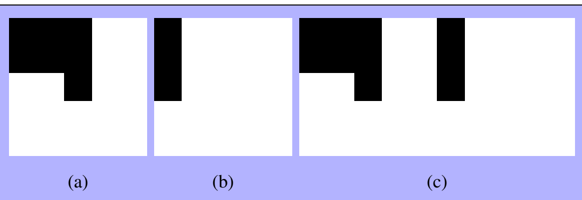


Fig. 1: Binary representation of the bin and item where occupied positions are represented by '0' and unoccupied ones by '1'. (a) A bin of size $W \times H$ is filled with items and (b) an item to be placed inside that bin. The state (c) is defined by concatenating these two binary image where the first matrix of size $W \times H$ represents the bin and the next matrix of size $W \times H$ represents the item to be placed.

A. States

The rectangular bin is represented by a binary image of size $W \times H$ where occupied positions are represented by '0' and unoccupied one by '1' as shown in figure 1. It can be obtained via a thresholding operation of the depth image obtained from an overhead sensor or directly assigning the pixels corresponding to the occupied positions as '0' and the unoccupied positions as '1'. Similarly, the image corresponding to the item is also binarized. Then, this binary representation of the bin and the item is concatenated to form the **state** of the system. Hence, at time point t the state s is represented by a matrix of size $W \times 2H$ where the first matrix of size $W \times H$ is the binary representation of the bin and the next $W \times H$ matrix is the binary representation of the item to be placed.

B. Actions and Reward

In every state, there is $WH+1$ number of actions available to the agent. WH actions pertain to the potential places where an item can be placed in the bin. By a potential place, we mean the unit square where the left top corner of the rectangular item is placed. Since there are WH unit squares present in the rectangular bin of size $W \times H$, there are WH possible places in which the item can be placed which gives rise to WH actions. There is one additional action which corresponds to not placing the rectangular item in the bin at all. Hence, in total there are $WH+1$ number of actions for every state for the agent.

It is very easy to see that even though there are $WH+1$ actions available in each state of the system, many of the actions should not be feasible as an item will not be placed in an already occupied place in the bin. Similarly, if the item cannot be placed fully inside the bin, then also it gives rise to an infeasible action. To accommodate for such constraints, the reward function $\mathcal{R}(s,a)$ where $s \in \mathcal{S}$ and $a \in \mathcal{A}$ has been defined in the following way:

$$\mathcal{R}(s,a) = \begin{cases} -5, & \text{if the item does not fit} \\ & \text{inside the bin or if the item} \\ & \text{is placed in an already} \\ & \text{occupied place in the bin} \\ \text{cluster size}\times & \\ \text{compactness}, & \text{otherwise} \end{cases}$$

If the action is feasible as per the above criteria, then the agent places the item inside the bin. The reward for a feasible action is directly proportional to the size of the cluster that the placed item forms with the existing item(s) inside the bin. The idea here is to place the incoming item adjacent to the already placed bin item(s) so that the maximum useable area is left for future placements. After every successful action, we scan the bin to determine the size of the cluster size that is formed due to the current action and the reward for the action is assigned based on how big is the cluster size. Another component in the reward is the compactness of the cluster formed by the current action. Below we describe how we evaluate cluster size and effectiveness of the formed cluster, compactness of a cluster.

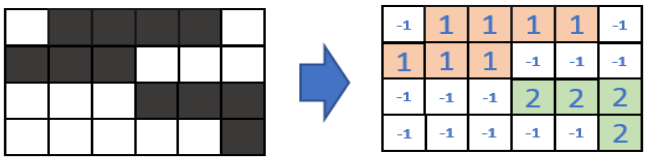


Fig. 2: Formation of a cluster with a grid of size 6×4 based on 4-connectivity. Left: The representation of the bin. Right: Clustering label. We get two cluster labelled with 1 and 2. Note that 1 and 2 do not share an edge and hence those are two different clusters. Unoccupied elements are denoted by -1.

Cluster Size and Cluster Formation: As proposed in section III-A the bin is represented as a 2D matrix call it B, of size $W \times H$. On the matrix B, we group each element by Connected Component Labelling (CCL) where neighbouring elements with the same values are assigned to the same cluster. The neighbour of an element is defined by 4-connectivity i.e for an element located at (i,j) in B, the adjacent elements location will be $(i \pm 1, j)$ or $(i, j \pm 1)$. In other way, an element will be a neighbour to other if they share an edge i.e they will be connected with each other either horizontally or vertically. The size of a cluster is defined as the number of elements in the cluster. A representative example is shown in Figure 2 where the size of cluster denoted by 1 is 7 and that of the cluster denoted by 2 is 4.

Different actions result in different reward amounts based on the cluster size formed by the action. As shown in Figure 3 an item which occupies grid of size 2×1 is to be placed inside a partially filled bin of grid size 6×4 at a time t. Out of the all feasible action as an example, we compare the quality of the action 'b' (at $(0,0)$) and 'c' (at $(0,5)$). Action 'b' gives rise of a cluster of size 7 whereas based on action 'c' the cluster size is 5. Therefore action 'b' will give rise to higher cluster size.

Compactness: Let us assume that at time t the Connected Component Clustering leads to the creation of a total of M clusters denoted by $(g_1, g_2, \ldots, g_M)$ in the bin and action at time t denoted by a_t updates i^{th} cluster g_i. Consider the cluster g_i with size S_{g_i}. The compactness of the cluster g_i is denoted by C_{g_i} and is defined as

$$C_{g_i} = \frac{S_{g_i}}{A_{g_i}} \tag{2}$$

where A_{g_i} is the area of the smallest rectangle that completely encloses the cluster g_i. Although a feasible action is awarded by the cluster size the item formed with the existing items, but sometimes it can lead to a zigzag cluster as shown in Figure 3 when action 'c' is taken. However when action 'b' is taken, the items in the cluster are less haphazard giving rise to a higher value of compactness. So, compactness of the cluster is a measure of how grouped the items are

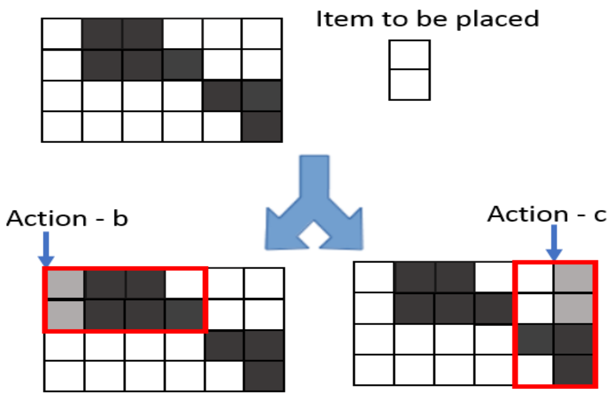


Fig. 3: Formation of a cluster at time t with a bin of size 6×4. The bin is partially filled (represented by darker shades) and the incoming item is of size 2×1. Two possible actions 'b' and 'c' are shown here. The arrow points to the top-left corner for each action. Action 'b' results a cluster size of 7 while action 'c' gives rise to a cluster of size 5. Therefore, the cluster size for action 'b' will be higher. The figure also shows the compactness of the cluster formed by action 'b'. The red rectangle is the smallest rectangle that completely encloses the cluster formed by action 'b'. The compactness is given by $\frac{7}{8} = 0.875$. Right: The compactness of the cluster formed by action 'c' is $\frac{5}{10} = 0.5$.

within a cluster. It also, in turn, makes remaining empty places more grouped and hence increases the possibility of placing more boxes inside the bin in future. In Figure 3, reward $\mathcal{R}(s,b) = 0.875 \times 7 = 0.6125$ by taking action 'b' and $\mathcal{R}(s,c) = 0.5 \times 5 = 0.25$ by taking action 'c'. Hence, action 'b' yields more reward in this case. An iteration will stop when each item in the sequence is considered for placement or there is no room inside the bin for further placement.

1) Reward at the last step: Each action gets positive or negative reward based on the formulation of $\mathcal{R}$ for each item in the sequence.the We believe that this reward formulation will help us to maximize packing efficiency. But to emphasize it more, an additional reward $r_l = K \times PE$ is added at the end of each iteration (after all boxes have arrived). K is a constant and PE is the packing efficiency defined in Equation 1.

2) Q-Network Architecture: We model the bin packing problem using Double Q-Learning [30] where action-value function $Q(s,a)$ is learned over the time via a neural network. The CNN takes the states (s) as input and generates the 'quality' of every action (a) as output. The CNN network as shown in Figure 4 is constructed using 5 convolution layer, 3 max-pooling layer and 2 fully connected layers at the end. The input (states) is applied to $conv1$ and the output of $conv1$ is applied to $conv2$, two convolution layers with $16, 5 \times 5$ and $32, 5 \times 5$ filters respectively. A max-pooling operation is applied with 2×2 filter with a stride of 2×2 which is then followed by a convolution operation ($conv3$) with $16, 3 \times 3$ filters and a pooling operation. Again the same two layers are repeated twice ($pool$, $conv4$ and $pool$, $conv5$) where parameter for pooling operation is same as earlier pooling layer and $conv4$ has 16 filters of size 3×3 and $conv5$ has 1, 3×3 filter is applied and then, two consecutive fully connected layer ($fc1$ and $fc2$) leads us to the final predicted value where $fc1$ has a 64 nodes and $fc2$ has nodes equal to the number of actions.

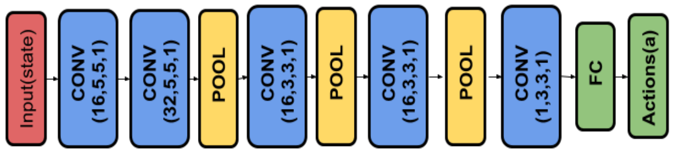


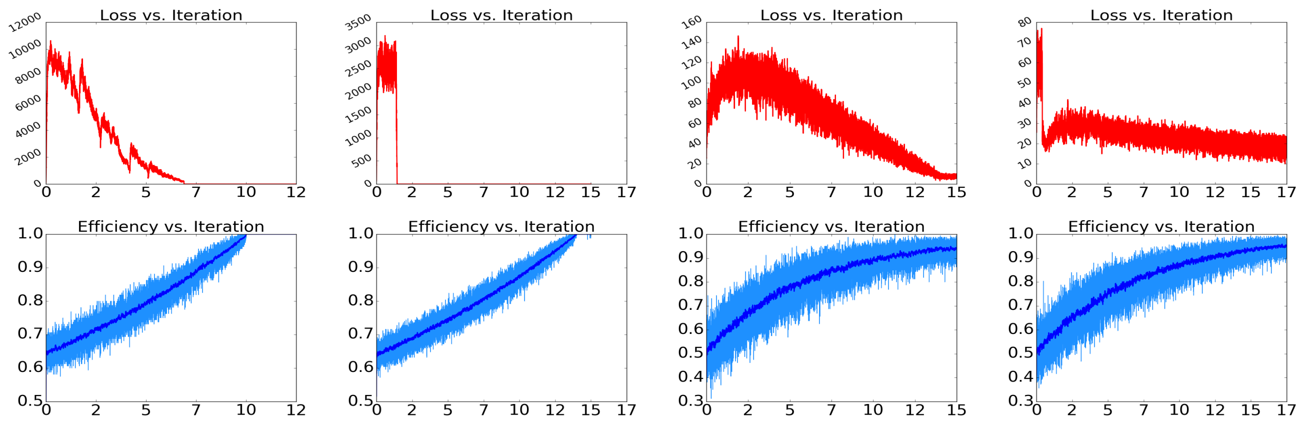
Fig. 4: Architecture of the CNN model.

At the start of ith time step, Q network generates output as the $quality$ of every action for that state (a bin configuration) out of which maximum Q-valued action a is selected. The agent executes the action i.e places the incoming item inside bin and gets a reward r from the environment due to that placement. The target network calculates the Q-value of that action. Essentially, we are computing how good is a by adding r and Q-value for that action by using the target network. We want $Q(s,a)$ to be equal to this value (y_i). The difference between these two is the loss L function. The parameters of the network are updated such that the loss is minimized.

IV. Experiments

As discussed in the earlier section, the network returns a set of Q-values corresponding to each of the action in the action space which is of size $WH+1$. The CNN has a decaying learning rate starting from $10e^{-4}$ and AdamOptimizer [31] for backpropagation. The discount factor (gamma) for RL is 0.95 and the replay memory [32] has a maximum size of 20000. The value of constant K is fixed to 2. All the models are trained with above-mentioned configuration with ϵ decreasing from 1 to 0 in linear steps.

We present results with square sized bins but our algorithm performs well for rectangular bins as well since the proposed algorithm does not depend on the type (squared or rectangular) of the bin. In an iteration during training, a sequence comprising of Rectangular items are to be packed in a bin of size $W \times W$. For that, we first generate W^2 random items with varying sizes $I_{w_t} \times I_{h_t}$, where $0 < I_{w_t} \leq W$ and $0 < I_{h_t} \leq H \forall t \in [1, 2, \ldots, W^2]$ and then we sequentially send these items one by one to the algorithm. We first consider the case where a sequence of W^2, unit size (1×1) items are to be packed inside te bin. For this, in an iteration during the training phase, W^2 items are sent sequentially and a suitable reward is assigned after each action. We present the results for this case with $W = 4$ (Figure 5(a)) and $W = 5$ (Figure 5(b)). The model is trained for 200000 iterations for $W = 4$ and 300000 for $W = 5$. Figure 5(a) and Figure 5(b) show the loss (top figure) and packing efficiency plot (bottom figure) with respect to iterations in the training phase for 4×4 and 5×5 bin. One unit of 'x' axis corresponds to 20000 iterations in the training phase. It can be observed from Figures 5(a)-5(b) that the loss is decreasing and the efficiency is reaching to 1.0 with iterations. For the testing



(a)W=4, unit square items (b)W=5,unit square items (c)W=4, rectangular items (d)W=5, rectangular items

Fig. 5: Top Row: Loss (y) and Bottom Row: Efficiency (y) vs. Iteration (x) plot during training phase with unit Square and Rectangular items inside a squared bin ($W \times W$). One unit in 'x' axis corresponds to 20000 iterations in training phase.

phase we again sequentially send W^2 unit size (of size 1×1) items and evaluate the packing efficiency and plot it in Figure 6. From the figure, it can be seen that for each of $W = 3$, $W = 4$ and $W = 5$, the packing efficiency is 1 (red dotted line) for unit sized items.

TABLE I: Performance of the method with unit (1×1), Square (S) and Rectangular (R) items

Item Type	4 × 4 bin		5 × 5 bin	
	Sequence $freq - item$	**Efficiency**	**Sequence** $freq - item$	**Efficiency**
Unit	16 − 1×1	1.0	25 − 1×1	1.0
S	1 − 4×4	1.0	1 − 5 × 5	1.0
	1 − 3×3, 7 − 1×1	0.84	4 − 2×2, 9 − 1×1	0.91
	3 − 2×2, 4 − 1×1	0.97	1 − 3×3, 1 − 2×2 4 − 1×1	0.95
R	1 − 3×2, 1 − 1×4 1 − 2×2, 1 − 1×2	0.93	4 − 2×2, 1 − 1×4 1 − 4×1, 1 − 1×1	0.93
	1 − 1×1, 1 − 1×3 1 − 3×3, 1 − 3×3	0.98	1 − 2×5, 2 − 3×3 1 − 3×2	1.0
	1 − 4×2, 2 − 2×2	1.0	1 − 4×4, 1 − 1×4 1 − 4×1, 1 − 1×1	1.0

TABLE II: Performance of the method for Rectangular (R) items with Different Bin Dimension

Bin Dimension	Efficiency
3×3	90 %
4×4	95 %
5×5	91 %

BinPack Method	Efficiency
Shelf	72 %
ShelfNextFit	73 %
Skyline	85 %
Proposed Method	91 %

Note that, when the CNN is learnt efficiently, all W^2 unit items are expected to be placed in a bin of size $W \times W$ occupying all the bottom surface area of the bin which leads to an efficiency of measure of 1. However, if the sequence of items is comprising of Rectangular or Squares of varying sizes which are to be packed in a bin of size $W \times W$, it may not be possible to optimally pack the items so that the surface area of the bin is fully occupied. Just for an illustration if the sequence of items are of size 2×2, 1×3, 2×2 to be packed in bin of size 3×3 then the optimal solution is to

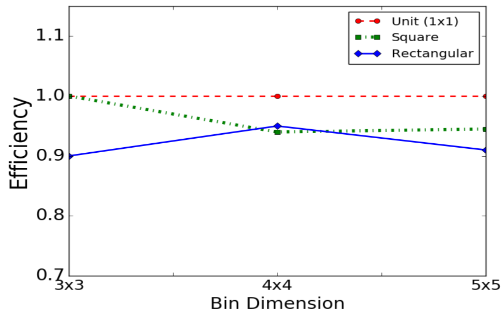


Fig. 6: Packing Efficiency of proposed method with unit, square, rectugar items

place the items of size 2×2 and 1×3 into the bin which will lead to an optimal efficiency measure $\frac{7}{9} = 0.78$.

In the next phase, the packing performance with Rectangular and Square items inside the bin of size $W \times W$ is discussed. In an iteration during training, W^2 random items with varying sizes $I_{w_t} \times I_{h_t}$ are send sequentially to the algorithm. With bin size of $W = 3$ and $W = 4$, the model is trained for 300000 iterations and for $W = 5$, the model is trained for 350000 iterations. It can be observed from Figures 5(c)-5(d) that during training the loss is decreasing with iterations (one unit of 'x' axis corresponds to 20000 iterations in training phase) and the efficiency increases close to 1 but can not achieve 1. This is because of the fact that the optimal efficiency may not be exactly equal to 1 for Rectangular and Square items.

During testing phase with varying Rectangular and Square items to be packed in a bin of size $W \times W$, the obtained efficiency is given in Table II and in Figure 6. In the testing phase, the varying sequence is carefully chosen such that in the optimal packing configuration all the items can be placed in the bin with full occupancy. At first, the experimentation with square items is done with 20000 varying sequence of only Square items and finally, the efficiency is the average of

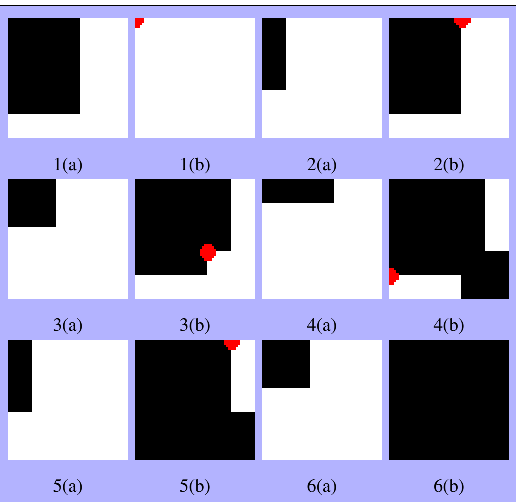


Fig. 7: Bin Filling with a sequence of rectangular items. The sequence consists of 6 items which has to be placed sequentially. For figure 1-6, the first image (a) of the pair shows the item and second one (b) is the bin condition. The red dot in each bin 1-6(b) represents the placement position of the top left corner of the item (1-6(a)). In the last case (Figure 6), there is no red dot which corresponds to 'no action'.

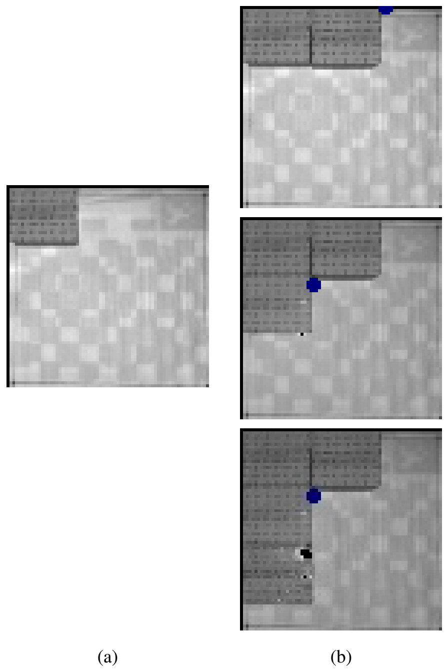


Fig. 8: Experimental results performed with an overhead depth sensor. The First column (a) shows the item to be placed inside the bin where the second column (b) represents the current bin condition where the item has to be placed. For these item and the bin, the corresponding action is pointed by the *blue* dot in each bin. An action denotes the position where top left corner of the item will be placed.

the packing efficiencies obtained in those 20000 experiments. We plot (green dashed line) the efficiency in Figure 6 when the considered items are only Square and the bin size is 3×3 (efficiency = 1), 4×4 (efficiency=0.94) and 5×5 (efficiency = 0.95).

Next, in the testing phase, a sequence of Square and Rectangular items is considered. For each such sequence, we evaluate the packing efficiency with 50000 varying sequence of Rectangular and Square items. The final efficiency is the average of the efficiencies obtained in those experiments. We plot (blue line) the efficiency in Figure 6 when the items considered are Rectangular and Square and the bin size is 3×3 (efficiency = 0.9), 4×4 (efficiency=0.95) and 5×5 (efficiency = 0.91). In Table II we present the efficiency for a few pre-defined sequences of items with varying size. The ordering of the items are randomized for each of the 5000 experiments. The average of the efficiencies of those 5000 experiments is shown in Table II against the pre-defined sequence. One such example of packing is shown in Figure 7 where six rectangular items has to be placed in a bin. Images 1-6 of Figure 7 have shown the act of placement sequentially. Each 1-6 figures consists of a pair of images where the first one (a) is the item and second one (b) is the bin where the item has to be kept. The red dot in each (b) figure represents the 'action' where the item would be placed. As we can notice here, the bin is full after $5th$ placement (6(b)) and it cannot place the $6th$ item therefore leading to choose 'no action'.

The performance in terms of efficiency of our proposed algorithm is compared with some of the state-of-the-art methods like shelf fit [25], ShelfNextFit [23], skyline fit [26] and the results are shown in table II(c). Our proposed algorithm produces an efficiency of 91% which outperforms each of the state-of-the-art methodologies. The comparison results are calculated by passing the same sequence of items to all of the state-of-the-art methodologies as well as our proposed algorithm in an online manner. The bin considered is of size 5×5. The experiment is repeated 5000 times with 5000 varying sequences of items and then the efficiency is calculated as the average of packing efficiencies obtained in 5000 sequences. All the above methods are Heuristic methods and they do not have the advantage of reinforced learning capabilities that the proposed RL based method has. Hence, while in the long run, the RL based method does learn to place items more efficiently, the heuristic based methods always produce the same solution again and again. The strength of Deep RL algorithms in terms of noise handling, approximating complex functions, getting feedback in order to improve its own performance will help our method to be integrated with robotic (AI) systems with required performance and can also be very easily adapted with different scenarios which may not be the case for heuristic solutions.

Experimentation in Real Environment: Apart from the simulated environment, we have also performed testing with real environment with binarized depth Images obtained frim IFM camera. The bin is of size 60×60 with each item of size 20×20. It is quantized by 6×6 grid where unit grid distance represents 10 points in bin. A grid is considered occupied if any part of the grid is occupied. It removes the constraint that the items has to be integer multiple of the grid. The grid is then resized to form the states suitable for RL technique. Figure 8 illustrates some example cases where the image in the first column represents the item to be placed inside the bin (second column (b)). With the current bin and item configuration, the blue dot in Figure 8(c) denotes the action i.e the position of top left corner of the item. The run time per item in the testing phase is almost $4ms$ making the method suitable for real time applications. From our experiments it is evident that our proposed method works well in real environments that too in real time.

V. Conclusion

Packing problem is popular due to its huge number of applications where the main objective is to maximize bin utilization. Towards that end, we have proposed a new method for online 2D packing leveraging Deep Reinforcement techniques. The novel state definition helps our method to work in real life environments. The reward function makes sure to place the incoming items close to existing items inside the bin. As evident from the results, our method yields good performance. In comparison to a few state-of-the-art methods for 2D BPP, our proposed method works better than each of these state-of-art methods. The efficiency of our proposed method has been at 91% for different configurations of bin sizes and items while each of the state-of-the-art methods yields an efficiency below 90%. The efficiency of our methods has been tested rigorously on different configurations of bean sizes and different ordering of the sequence of items arriving one by one for placement in the bin. It is found that the efficiency of our method is more than 90% for each of such configurations. Although this work is focused on 2D packing and it can be extended towards 3D packing in the real environment with robotic systems.

References

[1] E. Hopper and B. Turton, "A genetic algorithm for a 2d industrial packing problem," *Computers and Industrial Engineering*, vol. 37, no. 1, pp. 375 – 378, 1999. Proceedings of the 24th international conference on computers and industrial engineering.

[2] J. Levine and F. Ducatelle, "Ant colony optimization and local search for bin packing and cutting stock problems," *Journal of the Operational Research Society*, vol. 55, pp. 705–716, Jul 2004.

[3] A. Layeb and S. Chenche, "A novel grasp algorithm for solving the bin packing problem," *International Journal of Information Engineering and Electronic Business*, vol. 4, pp. 8–14, 04 2012.

[4] S. S. Seiden, "On the online bin packing problem," *Journal of the ACM (JACM)*, vol. 49, no. 5, pp. 640–671, 2002.

[5] X. Han, F. Y. Chin, H.-F. Ting, G. Zhang, and Y. Zhang, "A new upper bound 2.5545 on 2d online bin packing," *ACM Transactions on Algorithms (TALG)*, vol. 7, no. 4, p. 50, 2011.

[6] E. F. Grove, "Online bin packing with lookahead.," in *SODA*, vol. 95, pp. 430–436, 1995.

[7] J. Balogh, J. Békési, G. Galambos, G. Dósa, and Z. Tan, "Lower bound for 3-batched bin packing," *Discrete Optimization*, vol. 21, pp. 14–24, 2016.

[8] M. Haouari and M. Serairi, "Heuristics for the variable sized bin-packing problem," *Computers and Operations Research*, vol. 36, no. 10, pp. 2877 – 2884, 2009.

[9] E. C. man Jr, M. Garey, and D. Johnson, "Approximation algorithms for bin packing: A survey," *Approximation algorithms for NP-hard problems*, pp. 46–93, 1996.

[10] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali, "Approximation and online algorithms for multidimensional bin packing: A survey," *Computer Science Review*, vol. 24, pp. 63–79, 2017.

[11] A. Lodi, S. Martello, M. Monaci, and D. Vigo, "Two-dimensional bin packing problems," *Paradigms of combinatorial optimization: Problems and new approaches*, pp. 107–129, 2014.

[12] D. Liu, K. C. Tan, S. Huang, C. K. Goh, and W. K. Ho, "On solving multiobjective bin packing problems using evolutionary particle swarm optimization," *European Journal of Operational Research*, vol. 190, no. 2, pp. 357–382, 2008.

[13] J. C. Gomez and H. Terashima-Marín, "Evolutionary hyper-heuristics for tackling bi-objective 2d bin packing problems," *Genetic Programming and Evolvable Machines*, vol. 19, no. 1-2, pp. 151–181, 2018.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

[15] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *CoRR*, vol. abs/1611.09940, 2016.

[16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[17] R. Jin, "Deep learning at alibaba.," in *IJCAI*, pp. 11–16, 2017.

[18] H. Hu, X. Zhang, X. Yan, L. Wang, and Y. Xu, "Solving a new 3d bin packing problem with deep reinforcement learning method," *CoRR*, vol. abs/1708.05930, 2017.

[19] A. Bouganis and M. Shanahan, "A vision-based intelligent system for packing 2-d irregular shapes," *IEEE Transaction on Automation Science and Engineering*, vol. 4, pp. 382–394, July 2007.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[21] M. S. Levin, "Towards bin packing (preliminary problem survey, models with multiset estimates)," *CoRR*, vol. abs/1605.07574, 2016.

[22] B. S. Baker, "A new proof for the first-fit decreasing bin-packing algorithm," *Journal of Algorithms*, vol. 6, no. 1, pp. 49 – 70, 1985.

[23] Z. Zhu, J. Sui, and L. Yang, "Bin-packing algorithms for periodic task scheduling," in *2010 WASE International Conference on Information Engineering*, vol. 2, pp. 207–210, Aug 2010.

[24] G. Dósa and J. Sgall, "Optimal analysis of best fit bin packing," in *Automata, Languages, and Programming* (J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, eds.), (Berlin, Heidelberg), pp. 429–441, Springer Berlin Heidelberg, 2014.

[25] J. Csirik and G. J. Woeginger, "Shelf algorithms for on-line strip packing," *Information Processing Letters*, vol. 63, no. 4, pp. 171 – 175, 1997.

[26] L. Wei, D. Zhang, and Q. Chen, "A least wasted first heuristic algorithm for the rectangular packing problem," *Comput. Oper. Res.*, vol. 36, pp. 1608–1614, May 2009.

[27] P. Garrido and M.-C. Riff, "An evolutionary hyperheuristic to solve strip-packing problems," in *Intelligent Data Engineering and Automated Learning - IDEAL 2007* (H. Yin, P. Tino, E. Corchado, W. Byrne, and X. Yao, eds.), (Berlin, Heidelberg), pp. 406–415, Springer Berlin Heidelberg, 2007.

[28] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 2692–2700, Curran Associates, Inc., 2015.

[29] H. Hu, L. Duan, X. Zhang, Y. Xu, and J. Wei, "A multi-task selected learning approach for solving new type 3d bin packing problem," *CoRR*, vol. abs/1804.06896, 2018.

[30] H. V. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems 23* (J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, eds.), pp. 2613–2621, Curran Associates, Inc., 2010.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.