A Physics-enabled Simulation Environment for Solution of O3D-BPP using Feedback-Driven DRL Technique

Aditya Jain[1], Aniruddha Singhal[2] and Richa Verma[3]

***Abstract*— In this paper, we propose a robotic solution to the problem of container loading in warehouses and distribution centers. The container loading problem is modeled as an Online 3D Bin Packing Problem (O3D-BPP), where the information about future sequences of the objects to be packed is not known apriori unlike the offline problem which necessitates aprior knowledge. We propose a novel DQN-based deep reinforcement learning (DRL) approach to solve the O3D-BPP. We have developed a physics-enabled PyBullet simulation environment of a warehouse sorting center in which a reinforcement learning agent is trained and tested to optimally solve the O3D-BPP on a custom dataset with known baseline efficiency. The simulation environment is able to mimic real-life scenarios like toppling, gaps in packing and orientation changes. This work will abridge the gap between simulation and real-world scenario and the trained algorithm will be easily deployed in a real-world system. We aim to deploy the model learnt in the simulation on a prototype of the robotic sorting centre created in our lab.**

I. Introduction

In a typical Distribution Center (DC), the internal transportation of goods and parcels is majority of the work. The work of internal transportation comprises of recognition of type of goods, tight packing of goods in containers and optimal path planning of the vehicles carrying these containers. In this paper we will discuss the solution to the problem of tightly packing goods inside a container. It is mathematically equivalent to Three Dimensional Bin Packing Problem (3D-BPP) which is also a well known problem in computer science. It is a NP-hard problem and therefore it has no known optimal solution. A more difficult variant of the problem is called Online 3D-BPP (O3D-BPP) where the information about the boxes to be packed is not known in advance. The best known solutions to O3D-BPP uses a greedy approach and performs only marginally better than simple greedy approaches like first fit, next fit and worst fit.

The difficulty of O3D-BPP is dependent upon the heterogeneity of the boxes to be packed, the number of containers available for packing, allowed orientations of boxes, weight restrictions and stability constraints. A scenario with less heterogeneity and more constraints is easier to solve because the possible positions for placement of boxes is limited and therefore a partially exhaustive search becomes feasible. Another challenge associated with on-line problems like O3D-BPP is unavailability of a ground truth for comparison and requires some work to generate base line cases so that meaningful comparisons can be made.

The authors work at TCS Innovation Labs, New Delhi - India
[1]jain.adi, [2]aniruddha.singhal, [3]richa.verma -@tcs.com

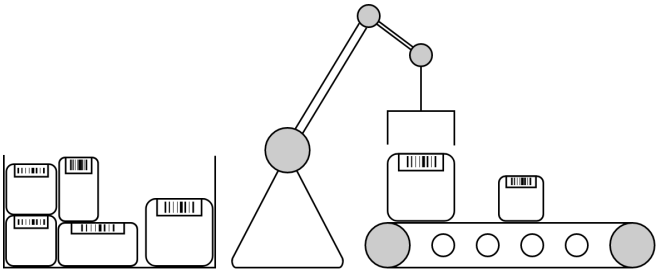


Fig. 1: The figure illustrates online 3D bin packing problem for a distribution center where parcels are coming on the conveyor belt and a robotic arm is packing the parcels in a container. The incoming stream of parcels have a limited visibility of at-most k parcels at a time and the robotic system takes real-time placement decision based on the visible sequence of parcels and training experience.

In this paper we propose a robotic solution to pick and pack heterogeneous sized boxes in containers for a warehouse or a DC. The tight packing of boxes in a container is both a algorithmic as well as an implementation challenge. A robotic packing system has several challenges associated with physical placement of a box. When a box is placed inside a container, the errors associated with toppling of box, imprecise placement, unstable placement etc. accumulate, because of which the expected position of placement and the actual position of placement of the box are different. In the problem of O3D-BPP it is essential that the actual state of the container and the expected state of the container updated by the algorithm should be identical. A mismatch in these states can lead to a suboptimal packing plan which will either effect efficiency of the packing or will cause a collision of boxes and container with the robotic arm.

Therefore, to solve the problem of robot packability and O3D-BPP, we propose a physics enabled open AI gym environment to mimic real-life bin packing problem which accounts the difference between the actual and observed state of the container. We also explore a novel Deep Q-Network (DQN) -based Deep Reinforcement Learning (DRL) method to solve O3D-BPP in such an environment.

II. Related Work

Offline Bin Packing Problem (both 2D and 3D) has been studied widely from a theoretical perspective. O3D-BPP has been surveyed from a warehouse point of view in a parcel loading context and has been formulated as a knapsack problem [1]. A more detailed study of nD-BPP is done in [2] with comparisons of different algorithms in the literature.

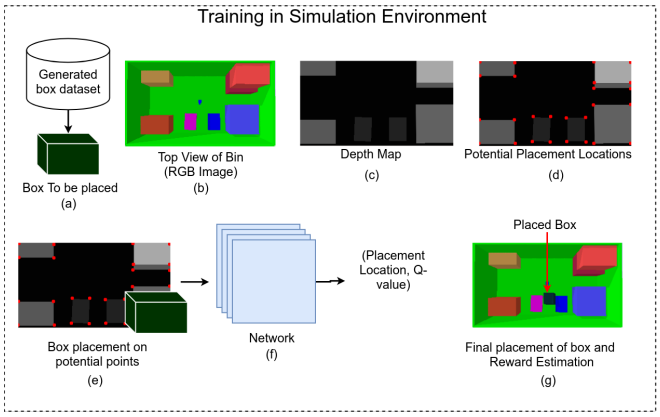


Fig. 2: The figure illustrates entire pipleline of the proposed system

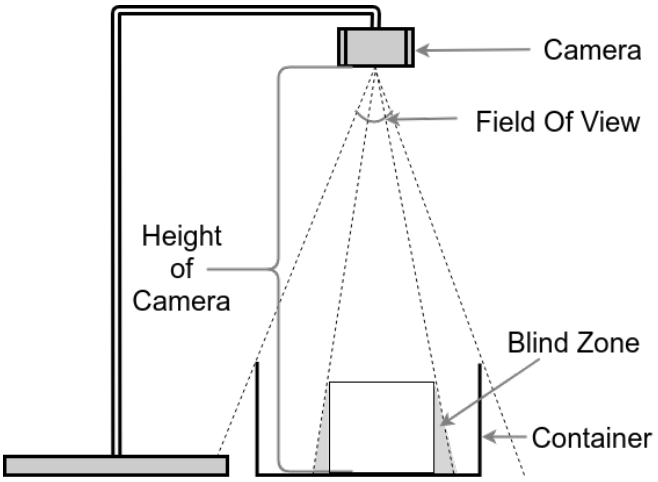


Fig. 3: This figure shows the setup of the depth camera along with the container. The camera captures depth images from a bird's-eye view. The field-of-view (FOV) is larger than the container and so the depth image is cropped before training. The blind zone area area varies inversely with the camera height

Most of the existing online methods focus on finding the theoretical bounds for approximate or heuristic methods for online bin packing [3] [4] [5]. The above heuristic methods are known to have poor generalization leading to very limited practical applications. Recent advancements in DRL have shown promising results in solving various combinatorial problems [6] [7]. This has encouraged people to use DRL for solving bin packing problems [8] [9], but mostly restricting them to the offline version. Thus the DRL framework has not been explored exhaustively and hence, lays the foundation for this work.

Robot packable solution for 3D bin packing problem have been discussed in [10], [5], [11], [12]. Recently [13] discussed packing strategy of known items arriving in random order.

A. RL approach to Combinatorial Problems

Heuristic algorithms are built specifically for a given problem and hence cannot be used for a general purpose problem. This challenge can be overcome by the use of reinforcement learning techniques because of their high adaptability. DRL techniques have shown promising results in large state spaces [14], as well as large or continuous action spaces [15]. Recent developments in DRL, have shown to perform equally well or better than heuristic algorithms in their respective problem areas and in some cases, even their human counterparts too, AlphaGo [16]. A value-based RL algorithm called TD(λ), which uses temporal difference between the q-values to update the policy of the agent, has been used in [17] for Job-Shop Scheduling problem for NASA payload processing task. Authors in [18] also propose a Q-Learning (Q-Table) approach for Job-Shop Scheduling problem by selecting proper state variables and implementing 2-step scheduling rules and epsilon-greedy strategy for action selection.

B. Closed Loop and Transfer Learning to Real Systems

Singh et al. [19] used a similar method to learn optimal policies using reinforcement learning techniques by directly training from raw sensory inputs, such as camera images. The authors first trained the agent in a simulation set up and then trained on real-robotic system to perform tasks like pushing, cloth draping and book keeping on the shelf. However, it requires goal examples apriori to train a classifier and also manual labeling during real-world interaction.

III. Robotic solution to O3D-BPP

The proposed solution has a robotic assembly as shown in Fig. 1 and Fig 3. It has a robotic arm and a depth camera to estimate the pose of the incoming box for picking. The packing decision is made by an RL agent based on the feed of the depth camera. The RL agent has learned the optimal way of packing by experimenting with millions of boxes in a realistic physics enabled simulation environment. In section IV we explain how this simulation environment is created and in section V we discuss how the agent is trained in this environment.

IV. Simulation Environment

For the purpose of training and testing the RL agent, we created two simulation environments. The first one is a static environment built on structured 2D array. This environment ensures non-overlapping of boxes, stable placement and robot packibility, but cannot mimic real-environment uncertainties like toppling, imprecise and unstable placement. The second environment is a physics-enabled Open AI Gym environment developed in PyBullet [20]. The pybullet environment exactly replicates the real world and takes account of all the uncertainties mentioned above. In such an environment, the RL agent can be better trained to pack boxes in a real world and can also learn certain hacks of packing which are possible only in a physics enabled environment. Following subsections contains details of implementation of each environment:

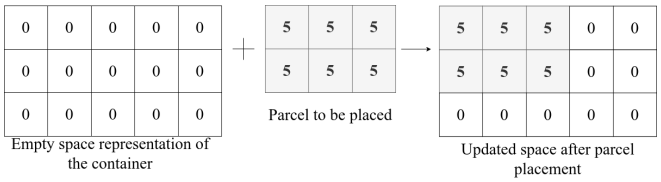


Fig. 4: A box being placed in a structured array env.

A. *Structured array environment*

The structured array-based environment consist of 2D array of dimensions $L \in N$ and $B \in N$ representing a container in which boxes are to be packed. There can be n such containers and therefore n 2D arrays representing those containers. The height of the container is the maximum value allowed at any location (x, y) such that $0 < x \leq L$ and $0 < y \leq B$. The non-overlap of boxes is modeled by consideration of space availability for the placement of a new box. A rectangular grid is overlayed on the point of placement of the box and the values in the rectangular grid are checked. If the all are equal and the height of the box does not protrude the container, then the box is placed, otherwise another placement location is sought. A visualization of 2D array for one empty container and a box is shown in Fig 4.

B. *Physics-enabled OpenAI Gym environment*

The Physics-enabled OpenAI Gym environment is created in PyBullet (Fig. 5). In contrast to structured array environment, the physics enabled pybullet environment can simulate real world properties. The PyBullet simulation environment is compatible with OpenAI Gym framework for ease of testing with different libraries like Tensorflow, PyTorch, Theano etc.

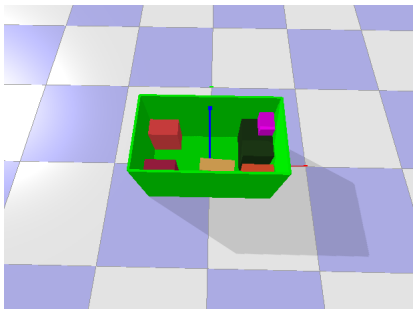
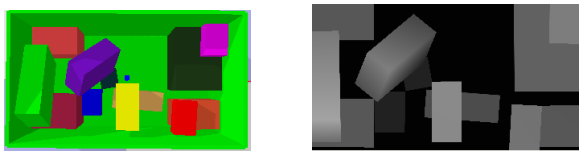


Fig. 5: PyBullet Simulation: Boxes of varied dimensions being packed in a green container



(a) RGB Image (b) Depth Map

Fig. 6: Top view of the container. As evident, the physics-enabled simulation takes into account the toppling, gaps while packing and orientation changes.

The environment consist of containers with boxes placed inside them and depth cameras viewing each container from the top (Fig. 3). When the environment is asked to return the current state, the depth camera takes snapshot of the container and returns it to the algorithm for decision making of box placement (Fig. 6). The environment is capable of simulating gravity, friction, toppling, slips etc. making it perfect replica of real world. The richness of environment helps in training of a robust decision making agent.

A practical challenge with the simulation environment is that the view of camera can be hindered by the placed boxes, thereby creating blind zones where no box can be placed as the place is not visible (Fig. 3).This problem was solved by increasing the height of the camera upto an extent from where the blind zones became insignificant.

One of the biggest advantages of this custom-built PyBullet environment is that it helps to build a closed-loop system. In real-world scenario, the box is not kept at the intended location in the container due to inherent calibration errors and thus there is always some offset in the placement. In many situations, the box also topples due to larger box weight than the box beneath. Thus the actual state space becomes very different from what would have been believed to be. If this is not taken into account in the algorithm, the error will propagate with every step of box placement and might lead to a crash of the entire robotic system.

V. DRL Algorithm

The solution of O3D-BPP is proposed using a combination of heuristics and RL techniques. The following sections presents the DRL network architecture, reward functions and algorithm flow:

A. *DRL Network Architecture*

The neural network architecture used is shown in Fig. 7. The current state and the believed state is concatenated before passing it through a max-pool layer. The vector is then flattened. The first hidden layer contains 800 units, the second layer has 400 hidden units and third has 200 hidden units. The output of the network is a one dimensional scalar which represents the Q-value estimate.

B. *Choosing placement location*

The placement location will be any point in a $m \times n$ grid. Initially, we were processing all the points in a matrix for every action. However, this method was computationally

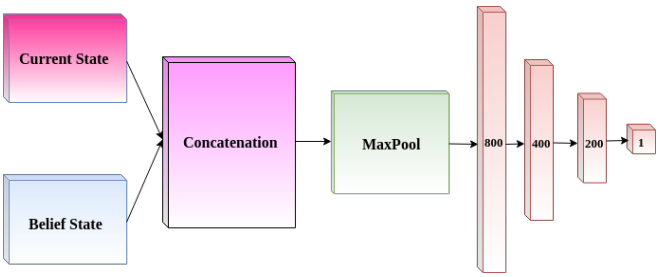


Fig. 7: The proposed DQN architecture

expensive and unrealistic, especially in the case of PyBullet simulation, wherein the state space can be as large as 1000×1000 pixels. Hence, only the corner positions are chosen as the potential points to keep incoming boxes. The $pointlist$, starts with a $[0, 0]$ or the top-left corner in its list. As the boxes are packed in the container, the corner positions of the added boxes are appended in the list.

C. Reward Function

At each time step t, the agent performs an action by placing a box at one of the feasible locations and receives a reward based on the below formulation:

$$r_t = \beta_1 \frac{v_{box,t}}{V_{container}} - \beta_2 \frac{v_{hole,t}}{V_{container}} + \beta_3 std_t - \beta_4 gradient_t$$

where β_i are the constant scalars. The first term in (1) is proportional to the volume of the box placed in the container while the second takes into account holes/vacant spaces created by keeping the box, if any. It has been seen that level packing (as humans do) ensures greater packing efficiency and stability. Thus, the third term is proportional to the standard deviation of the state space: $std_t = \frac{1}{(standard-deviation(statespace))}$, lower the standard deviation (i.e. more leveled the height is), the greater the reward is. The fourth term checks the gradient along the four sides of the newly added box. If the added box does create a new height, the gradient will be higher and thus, a lower net reward.

A terminal reward is also computed to check the performance of the agent during the entire episode.

$$T = \frac{V_{packed}}{V_{container}}$$

where V_{packed} is volume of the boxes packed in the container and $V_{container}$ is the volume of the container itself. This terminal reward is back-propagated through all the steps in the episode with a discount factor ρ = 0.99 before insertion into the replay buffer.

VI. Transfer Learning to Lab Prototype

A prototype of sorting centre was developed in the lab using a UR10 robot. The system consists of a conveyor belt on which parcels of different dimensions come in a stream. The parcels are ordered and singulated using a different module of the system which is out of scope of this study. The singulated parcel is picked up by the robotic arm and place inside the container. We aim to fine-tune the model learnt in the simulation by doing few trial runs and ultimately, test the efficacy of the model by placing heterogenous box sizes in the containers.

VII. Future Work

We have tested the above mentioned DQN algorithm in the matrix-based environment whose results are shown in Fig. 8. We aim to test the same in the PyBullet environment after incorporating more capabilities in the simulation system. We are also working on adding complexities in neural net architecture by making it multi-input and multi-ouput, so that it can extract better features from the input states. The last leg of the project will include turning the lab prototype of the robotic sorting centre into a closed-loop system, and achieving packing efficiency of atleast 80%.

Algorithm 1 Deep Q-learning for Container Packing

Initialize replay memory $\mathcal{M}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, E$ **do**
  for file $= 1, F$ **do**
    Reset the container state s
    $cornerlist$ = [0, 0]
    for $t = 1, T$ **do**
      Read the incoming box dimensions
      Possible placement locations (actions) will be in $cornerlist$
      With probability ϵ select a random action (location) a_t
      else select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
      Execute action a_t in environment and observe reward r_t and state s_{t+1}
      Update $cornerlist$ with the box corners
      Store transition (s_t, r_t, s_{t+1}) in $\mathcal{M}$
    Backpropagate terminal reward T through all the steps in the episode
    Sample random minibatch of transitions (s_j, r_j, s_{j+1}) from $\mathcal{M}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta) & \text{for non-terminal } s_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation **??**

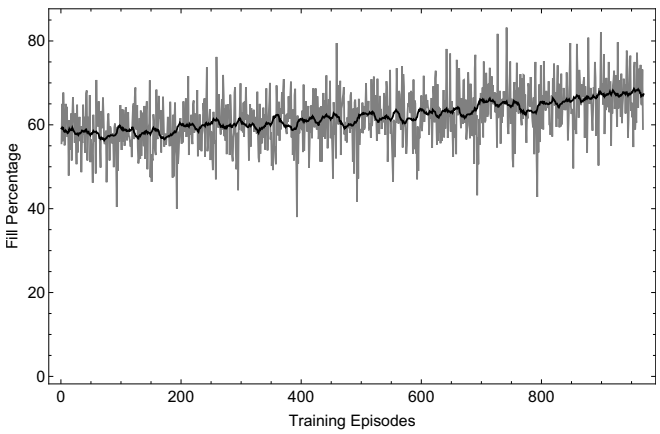


Fig. 8: The DRL agent learns to efficiently fill a container in structured env. scenario

References

[1] P. Kolhe and H. I. Christensen, "Planning in logistics: A survey," Georgia Institute of Technology, 2010.

[2] 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.

[3] S. S. Seiden, "On the online bin packing problem," in *International Colloquium on Automata, Languages, and Programming*, pp. 237–248, Springer, 2001.

[4] L. Epstein, "On online bin packing with lib constraints," *Naval Research Logistics (NRL)*, vol. 56, no. 8, pp. 780–786, 2009.

[5] F. Wang and K. Hauser, "Stable bin packing of non-convex 3D objects with a robot manipulator," *arXiv preprint arXiv:1812.04093*, 2018.

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

[7] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

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

[9] 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.

[10] D. Pisinger, 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, p. 7, 2007.

[11] E. D. Boef, J. Korst, S. Martello, D. Pisinger, D. Vigo, E. den Boef, J. Korst, S. Martello, D. Pisinger, and D. Vigo, "A note on Robot-packable and Orthogonal variants of the three-dimensional bin packing problem," in *Diku-rapport 03/02*, pp. 1–11, n, 2003.

[12] B. Mahvash, A. Awasthi, and S. Chauhan, "A column generation-based heuristic for the three- dimensional bin packing problem with rotation," *Journal of the Operational Research Society*, 2017.

[13] F. Wang and K. Hauser, "Robot packing with known items and nondeterministic arrival order,"

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[16] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. v. d. Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, p. 354, 10 2017.

[17] W. Zhang and T. G. Dietterich, "A reinforcement learning approach to job-shop scheduling," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, (San Francisco, CA, USA), pp. 1114–1120, Morgan Kaufmann Publishers Inc., 1995.

[18] Y. Wei and M. Zhao, "A reinforcement learning-based approach to dynamic job-shop scheduling," *Acta Automatica Sinica*, vol. 31, no. 5, p. 765, 2005.

[19] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-end robotic reinforcement learning without reward engineering," *arXiv preprint arXiv:1904.07854*, 2019.

[20] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." urlhttp://pybullet.org, 2016–2019.

[21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.