Towards adapting human behaviour in packing logistics through imitation learning

1st Hermann Foot
Packaging and Retail Logistics and AutoID-Technologies
Fraunhofer-Institute for Materialflow and Logistics
Dortmund, Germany
hermann.foot@iml.fraunhofer.de

2nd Benedikt Mättig
Packaging and Retail Logistics and AutoID-Technologies
Fraunhofer-Institute for Materialflow and Logistics
Dortmund, Germany
benedikt.maettig@iml.fraunhofer.de

***Abstract*—Despite numerous breakthroughs in the field of AI, there are still a large number of real-world application areas where humans cannot be replaced on an ad-hoc manner. Performance built through experience and knowledge is difficult to replicate, let alone surpass. Imitation learning offers an approach to make such strategies numerically accessible using demonstration data. In this paper we present an approach of transferring complex behaviour by experienced workers in packing logistics to a technical system. This method is investigated and evaluated on the basis of experiments motivated by real-life use cases. It is shown that imitation learning can be used to identify and describe characteristics of expert behaviour for this application. The intend is to use this as a basis for subsequent computation in especially volume minimization.**

***Index Terms*—imitation learning, bin packing, logistics, human-machine-transfer**

I. Introduction

Bin packing is one of the best-known algorithmic problems and is still relevant in practice due to its usefulness especially in logistics. It serves as the basis for optimization in the area of volume minimization for transport and storage. However, bin packing is known to be NP-hard, leading to the development of various approximation algorithms and heuristics for different scenarios. Despite these advancements, many approaches tend to focus solely on volume utilization, disregarding other important factors such as stability. This oversight can result in damages during transportation and avoidable return rides. Furthermore, the varying use cases in different industries create diverse requirements, making it challenging to define a universally applicable optimization function. Instead, it becomes necessary to tailor the optimization function according to specific practical requirements.

To address these challenges, the present work proposes an approach based on inverse reinforcement learning (IRL). This approach leverages the knowledge and expertise of experienced employees to define an application-specific optimization function. The outcome of this approach will serve as a basis for further load carrier optimizations or in the field of bin picking.

II. Related Work

The underlying concept of this work is based on the human-in-the-loop (HITL) framework, where humans play a crucial role in machine learning [11]. In this framework, humans utilize their expert knowledge, which is then incorporated into the technical system through interaction. This approach enables the knowledge that an employee has built up through experience to be mapped and reproduced in a technical system, which in turn can be used, for example, to train new employees, enabling knowledge transfer [7].
Imitation learning is presented as a concrete method for implementing the HITL framework. This involves integrating demonstrations from experts, who can be either humans or technical systems, into the learning process. Previous research has demonstrated the wide range of applications for this approach (eg. [8], [3], [2]). IRL is a particular type of imitation learning. The goal is to define a suitable optimization or reward function based on demonstration data.

In logistics, such approaches can be found in particular in the area of warehouse automation through driverless transport systems. But also picking robots can be supported by demonstrations during the training of motor skills [6]. For the algorithmic side of packing, bin packing, machine learning approaches also find their way into the field in addition to classical approximation algorithms and heuristics. Especially due to the developments in the field of deep neural networks, many approaches based on deep reinforcement learning have emerged, where the agents acts as the packer (e.g. [10], [12] [9]).
Although approaches have been published that make use of demonstrations in the learning process, this ignores the varying tasks and requirements that arise in practice which leads to approaches of limited use. IRL offers a way to make requirements and strategies from practice numerically tangible, but its application to the bin packing problem has not yet been evaluated.

III. Fundamentals

A. Bin Packing

The underlying algorithmic problem of the packing process is the three-dimensional bin packing problem. We assume that we have a load carrier that is limited in width and length. Since in practice there is often no physical restriction on the height, as in the case of a pallet, for example, the load carrier

is considered to be unrestricted. The packages are supposed to be cuboid objects, which are placed orthogonally on the load carrier. Formally, the considered problem can be defined as follows [4]:

Input: Load carrier width W und length L with $W, L \in \mathbb{R}_{>0}$, n Packages with width w_i, length l_i height h_i und weight g_i with $w_i, l_i, h_i, g_i \in \mathbb{R}_{>0}$ for $i \in \{0, ..., n\}$

Output: Coordinates (x_i, y_i, z_i) for $i \in \{0, ..., n\}$, such that:

$$\min \quad F(x, y, z)$$

$$\text{s.t.} \quad 0 \leq x_i + w_i \leq W \tag{1}$$

$$0 \leq y_i + l_i \leq L \tag{2}$$

$$s_{ij} + d_{ij} + u_{ij} = 1 \tag{3}$$

$$(x_j - x_i) \cdot s_{ij} \geq w_i \cdot s_{ij} \tag{4}$$

$$(y_j - y_i) \cdot d_{ij} \geq l_i \cdot d_{ij} \tag{5}$$

$$(z_j - z_i) \cdot u_{ij} \geq h_i \cdot u_{ij} \tag{6}$$

$$s_{ij}, d_{ij}, u_{ij} \in \{0, 1\} \tag{7}$$

$$x_i, y_i, z_i \geq 0 \tag{8}$$

for $i, j \in \{0, ..., n\}$.

The dimensions and weights of the packages are given, as well as the constraints defined by the load carrier. The resulting placements of the packages are given by the output variables x_i , y_i and z_i for $i \in \{0, ..., n\}$, where n corresponds to the number of elements to be packed. The coordinates give the position of the front left corner of the package on the carrier, where its front left corner of the load carrier defines the point $(0, 0, 0)^T$. The variables s_{ij}, u_{ij}, and d_{ij} are indicator variables, which determine whether a package p_i is located to the left, below or behind of another package p_j.
The conditions (1) and (2) ensure that each placed package does not extend beyond the dimensions of the load carrier. Conditions (3)-(6), on the other hand, ensure that the dimensions of the ensure that the dimensions of the packages do not overlap.
The optimization function F is represented by an abstract function, which is parameterized by the coordinates x, y, z of the packages. Looking at similar problems in bin packing, these are often limited to minimizing the height and volume and do not reflect the complex behaviour represented by the knowledge of an experienced worker. It is therefore necessary to replace the function F by an adequate optimization function motivated by the explicit and implicit expert knowledge.

B. Inverse reinforcement learning

Replicating the knowledge of an expert in an artificial system requires a way to represent its behaviour. The field of imitation learning addresses this challenge by using recordings in the form of demonstration data as a basis for reproducing the behaviour. One method used for this purpose is IRL. For this, the problem is modeled in the form of a modified Markov decision process. This is defined as follows:

A **Markov Decision Process without Reward** $\mathcal{M}_{\not\mathcal{R}}$ (MDP/R) is a stochastic process that can be described by the triple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$):

- The set of **states** $\mathcal{S}$ of $\mathcal{M}$
- The set of **actions** $\mathcal{A}$ of $\mathcal{M}$, where the subsets $\mathcal{A}_s \subseteq \mathcal{A}$ represent the possible actions in state $s \in \mathcal{S}$
- The **transition function** $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ of $\mathcal{M}$ specifies the probabilities of transitioning from a state $s \in \mathcal{S}$ to a state $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}_s$.

The strategy executed by the expert in this case can be defined by the policy function $\pi : \mathcal{S} \mapsto \mathcal{A}$, which specifies the corresponding action for each state. The sequence of states executed by the expert's actions is called a trajectory τ.
While in reinforcement learning a policy is sought, in the case of IRL it is already given. It is represented by the expert's dataset. It is assumed that the expert's policy is optimal. Therefore, the desired reward function $\mathcal{R}^*$ is maximized by the policy and can be described by the following expression:

$$\pi_E = \underset{\pi \in \Pi}{\operatorname{argmax}} \, \mathbb{E}[\mathcal{R}^*(s, a)], \, \forall s \in \mathcal{S} \text{ and } \forall a \in \mathcal{A}_s$$

In this work, the approach of Boularias et al. **Relative Entropy Inverse Reinforcement Learning** (REIRL) is implemented [1]. The idea is to minimize the relative entropy (Kullback-Leibler divergence) between an empirical distribution and a learned distribution. The missing reward function is a linear combination of different features that describe a state of the MDP/R. The development coefficients of the linear combinations represent the weights of the individual features. The sought reward function can be defined as follows:

$$\mathcal{R}^*(s, a) = \sum_{i=1}^{k} \theta_i f_i(s, a) = \theta f(s, a)$$

The features $f_i(s, a)$ for $s \in \mathcal{S}$ and $a \in \mathcal{A}_s$ numerically describe specific properties of a state. The development coefficients θ_i weight these features. Due to the form of presentation, a major advantage of the method compared to other approaches of the IRL is the better interpretability.
The features occurring in the expert demonstrations can be determined as follows:

$$\hat{f} = \frac{1}{|\mathcal{D}|} \sum_{(s,a) \in \mathcal{D}} f(s, a)$$

The function to be minimized can be formulated as follows:

$$g(\theta) = \sum_{i=1}^{k} \theta_i \hat{f}_i^{\tau} - \ln Z(\theta) - \sum_{i=1}^{k} |\theta_i| \epsilon_i$$

with

$$Z(\theta) = \sum_{\tau \in \mathcal{T}} \left(Q(\tau) \exp(\sum_{i=1}^{k} \theta_i f_i^{\tau}) \right)$$

where Q is a baseline policy. With the help of Importance Sampling, the gradient is estimated on the basis of samples, whereby the weights of the reward function can be trained. The procedure is described as pseudocode in the algorithm 1.

Algorithm 1 Relative entropy inverse reinforcement learning

Require: demonstration data $D_E \sim \pi_E$, trajectory samples $\mathcal{T}_{\pi_N} \sim \pi_N$, learning rate β, stopping criteria μ
Ensure: weights θ

$\theta^0 \leftarrow \vec{0}$
$j \leftarrow 1$
$\hat{f} = \frac{1}{|D|} \sum_{(s,a)\in D} f(s,a)$
repeat
  for all $\tau \in \mathcal{T}_{\pi_N}$ **do**
    $P(\tau|\theta^j) = \frac{\exp\left(\theta^j f^\tau\right)/\pi_N(\tau)}{\sum_{\tau\in T_B} \exp\left(\theta f^\tau\right)/\pi_N(\tau)}$
  end for
  for $i = 0$ **to** k **do**
    $\Delta\theta_i^j \leftarrow \hat{f}_i - \sum_{\tau\in\mathcal{T}_{\pi_N}} P(\tau|\theta^j) f_i^\tau - \alpha_i \epsilon_i$
    $\theta_i^{j+1} \leftarrow \theta_i^j + \beta\left(\Delta\theta_i^j\right)$
    $j \leftarrow j + 1$
  end for
until $||\theta^j - \theta^{j-1}||_2 < \mu$
return θ

C. Bin-Packing in the context of IRL

1) States and Transitions: A packing order contains, in addition to the available load carrier, a quantity of packages to be placed. The initial state is represented by the empty load carrier. If we place a package, we change the load. This corresponds to a transition, where the placement of the package represents the action and the loadings before and after the placement represent the states involved. The state space $\mathcal{S}$ of the MEP is thus composed of the possible loadings of the carrier. These are connected by actions, i.e. the placements of packages, which are determined by the selection and position of the package. However, the removal of a package is excluded. In addition, other subsequent changes in the position of packages, such as those caused by tipping effects, are ignored. We assume that the desired position of a package corresponds to the actual position. The transition function is thus deterministic.

2) Reward: In order to be able to describe a suitable reward function in the context of IRL, features must be defined to be able to describe the behaviour. These are properties of the load, which are described with the help of numerical key figures. For this purpose, features were defined and compiled within the scope of this work to explain the strategy of a worker. For our approach, ten features coming from the domain were used. In addition to features describing height and weight distribution, these also include values that can be used to evaluate the stability of a load.

IV. Experiments

A. Data Collection

In order to investigate the behaviour of the method suitable demonstration data is required. These were generated synthetically within the scope of this work. For this purpose a pack simulator was implemented for data acquisition. This is a graphical 3D application in which a user can virtually process packing orders (see Fig. 1). A load carrier in the form of a pallet and a selection of packages are made available to the user. If the user successfully completes the packing order, i.e. places all packages validly on the load carrier, the application exports the result as a demonstration.

The tool was used to recreate three different strategies that occur in logistics:

1) In the first strategy, the entire surface of the load carrier should be used to be able to distribute the packages as well as possible and at the same time keep the maximum height of the load carrier to a minimum.
2) The second strategy is said to behave similarly to the first, with the difference that a small margin should be left to the edges of the load carrier. In practice, this is particularly interesting if the load is to be subsequently protected with cushioning material. Without a margin, this would extend beyond the dimensions of the load carrier and thus lead to problems during storage.
3) The aim of the third strategy was to generate a load that was as stable as possible. In particular, unlike the two previous strategies, emphasis was placed on the weight and its distribution on the load carrier. It should be noted, however, that a pallet is evaluated as a whole and not each individual package.

For each of the above strategies, 100 trajectories were generated. Each trajectory included 8-12 packages.

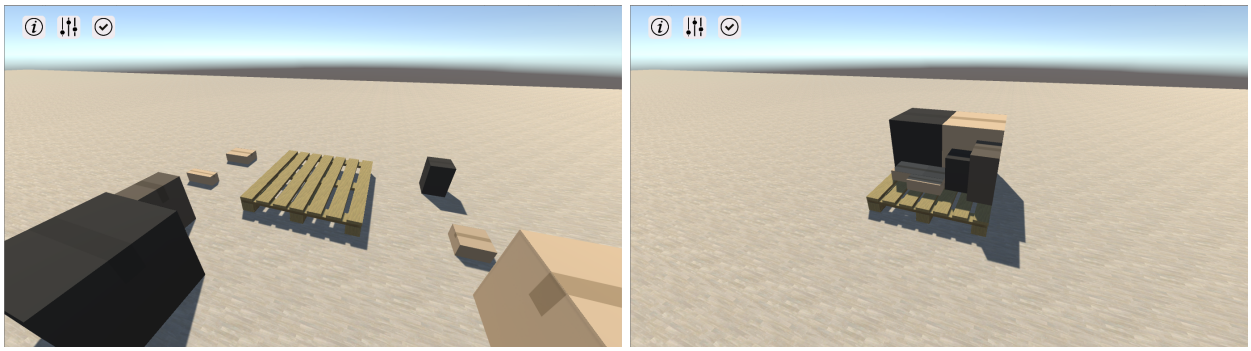
B. Features

The features are chosen in such a way that they satisfy the following properties: on the one hand the values of the features should lie in the same range of values, so that a comparability between them is guaranteed. between them is guaranteed. Here, the interval $[0, 1] \in \mathbb{R}$ is suitable. In addition, the features should be calculated with respect to the order, so that a comparability between different packing orders can be ensured.
The features used for the REIRL to describe a loading are defined as follows:

- Relative Maximum Height

$$RH = 1 - \frac{(\text{height} - \text{height}_{min})}{(\text{height}_{max} - \text{height}_{min})}$$

where height_{min} and height_{max} specify the theoretical minimum and maximum loading height given by the packages.



(a) Simulator at the beginning of a packing order.

(b) Simulator after completion of a packing order.

Fig. 1: Pack Simulator: The user of the software can virtually process packing orders. To do this, packages can be placed on the load carrier using drag-and-drop. When all packages are on the pallet, the user can confirm the loading and save it.

- Relative Maximum Weight

$$RW = 1 - \frac{(\text{weight} - \text{weight}_{min})}{(\text{weight}_{max} - \text{weight}_{min})}.$$

where weight_{min} and weight_{max} specify the theoretical minimum and maximum weight load given by the packages.

- Relative Margins

$$RUM = \frac{UM}{\max_i w_i}$$

$$RBM = \frac{BM}{\max_i w_i}$$

$$RRM = \frac{RM}{\max_i l_i}$$

$$RLM = \frac{LM}{\max_i l_i}$$

where UM, BM, RM and LM specify the upper, bottom, right and left margin given by the placements.

- Relative Packing Density

$$RD = \frac{\sum_{i=i}^{n} V_i}{(\max_x - \min_x) \cdot (\max_y - \min_y) \cdot \text{height}}$$

where V_i specifies the volume of package i.

- Overbuilt Ratio

$$OBR = 1 - \frac{\#\text{Packages} - \#_1\text{Packages}}{\#\text{Supporting Surfaces} - \#_1\text{Packages}}$$

where $\#_1$Packages specify the number of packages in the first layer, supported by the load carrier itself.

- Overhang Ratio

$$OHR = 1 - \frac{\#\text{Overhanging Packages} - \#_1\text{Packages}}{\#\text{Packages} - \#_1\text{Supporting Surfaces}}$$

where a surface is called overhanging if it is not supported by another supporting surface, given by other packages or the load carrier.

- Relative Barycarrier

$$RBC = 1 - \frac{c_z}{\text{height}}$$

where c_z specifies the height of the center of gravity given by

$$c_z = \frac{1}{\sum_{i=1}^{n} w_i} \cdot \sum_{i=1}^{n} z_i \cdot g_i$$

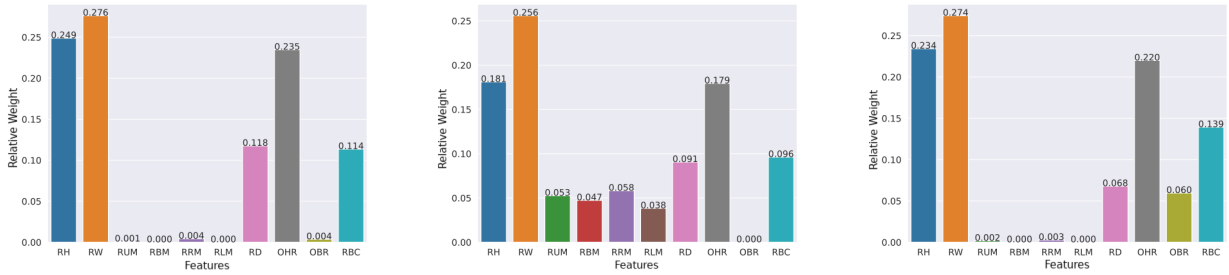
C. Implementation

The simulator was developed in Unity3D and uses the proprietary physics simulation to ensure valid loading. The resulting loadings from the program, which correspond to the states of the MDP, are repärsent using three-dimensional interval trees. The packages span one interval per dimension, which is managed in a separate tree data structure. The data structure allows efficient validation of placements in an asymptotic runtime of $\mathcal{O}(k + \log n)$, where k is the output set and n is the number of packages already present in the tree [5].

D. Results

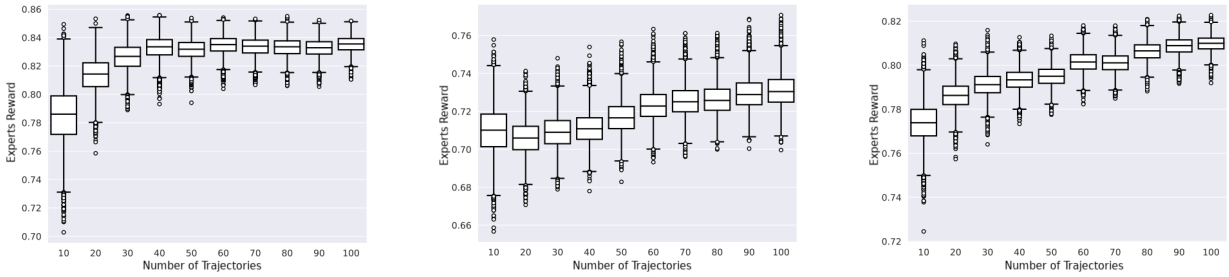
As part of the evaluation, two specific questions were examined:

1) Can we generally recognize and represent different packing strategies using the methodology?
2) How many trajectories are needed to recognize a packing strategy?



(a) Strategy 1 (b) Strategy 2 (c) Strategy 3

Fig. 2: Resulting weights of REIRL evaluation. The weight of a feature is an indicator of the degree to which it is expressed within the demonstration data.



(a) Strategy 1 (b) Strategy 2 (c) Strategy 3

Fig. 3: Variances of experts performance with increasing number of available trajectories.

To answer the first question, the weights of the respective features can be examined. These are visualized in the Figure 2. Here, the bars represent the weights of the features.

It can be seen that both height and weight distribution (RH and RW) have a high value in all three strategies, which is also desirable for all of those. Also, the values of Overhang Ratio (OHR), Relative Density (RD) and Relative Barcenter (RBC) have a notable share in all three strategies. Compared to Strategy 1, the relevance of the margin is evident in Strategy 2. The four features describing it (RUM, RBM, RRM, RLM) each show roughly equal weight, suggesting a uniformity of margin in the demonstration data. The fact that the edge was detected, in contrast to strategy 1, speaks to the method used. The third strategy represents the most complex. Especially the Relative Density (RD), Overbuild Ratio (OBR) and Relative Barycenter (RBC) are suitable to evaluate the stability of a loading. The three values should be maximized as far as possible. It can be seen from the results that the OBR in particular has gained weight compared to the other two strategies. The RBC can also show a higher percentage. Only the RD is lower than in the other two experiments. It would be desirable that here this would be higher at the expense of the RW or RH.

To investigate the number of expert demonstrations required, the average score determined by the REIRL was examined using the calculated reward function. For this purpose, the available amount of demonstrations was limited in a stepwise manner and the effects were studied. The results are visualized in Figure 3. The X-axis represents how many of the available trajectories were used. The specified quantity was drawn equally distributed from the entirety. For each subset, 10,000 experiments were performed. On the Y-axis the average expert performance is plotted, which results from the evaluation of the entirety of the trajectories by means of the found reward function. The box plots show the distributions of the results. For the first strategy, we can see a strong scatter with small amounts of data. If we increase this, on the one hand the average performance of the expert increases, on the other hand we also reduce the scatter. Strategies 2 and 3 tend to show a similar picture, but in strategy 2 in particular the dispersion is greater than in the other two strategies, even with high data volumes. In addition, it can be seen here that strategy 1 not only achieves higher performance than the other two strategies, but also requires less data to do so.

V. Discussion and Outlook

The evaluation showed that the application of IRL is a possibility for the identification of packing strategies in logistics. Due to the interepretability of the resulting reward function, the features of individual strategies could be modeled and recognized. The evaluations also give us a scope for the

number of trajectories needed to do this.
However, this approach required defining the features by hand, which runs the risk of disregarding relevant aspects, especially for even more complex strategies. Here the application of alternative IRL methods, which do without an explicit definition of features, could be a possibility to avoid this problem. Also, for the possible transfer into practice, an alternative to data collection has to be created. Here, systems using image and sensor information offer the possibility to record the packing process in an automated way in a warehouse and to gain expert demonstrations, even in much larger quantities. In addition, interaction with the system should be made feasible in the context of the HITL framework.
The intention behind the work was also to combine the result, i.e. the factors to be optimized, with a bin packer. Here it is necessary to design and implement suitable procedures of a further utilization of the reward function. Due to the technical approach, methods of reinforcement learning, for example, are a feasible option here.

REFERENCES

[1] A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 182–189, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[2] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4693–4700, 2018.

[3] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. *CoRR*, abs/1603.00448, 2016.

[4] H. Hu, X. Zhang, X. Yan, L. Wang, and Y. Xu. Solving a new 3d bin packing problem with deep reinforcement learning method, 2017.

[5] D. T. Lee. *Computational Geometry I*, page 1. Chapman and Hall/CRC, 2 edition, 2010.

[6] J. Mahler and K. Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. In S. Levine, V. Vanhoucke, and K. Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 515–524. PMLR, 13–15 Nov 2017.

[7] B. Mättig and H. Foot. Approach to improving training of human workers in industrial applications through the use of intelligence augmentation and human-in-the-loop. *International Conference on Computer Science and Education*, 15(1):201–213, 7 2020.

[8] K. Mülling, A. Boularias, B. Mohler, B. Schölkopf, and J. Peters. Learning strategies in table tennis using inverse reinforcement learning. *Biological cybernetics*, 108, 04 2014.

[9] A. V. Puche and S. Lee. Online 3d bin packing reinforcement learning solution with buffer, 2022.

[10] R. Verma, A. Singhal, H. Khadilkar, A. Basumatary, S. Nayak, H. V. Singh, S. Kumar, and R. Sinha. A generalized reinforcement learning algorithm for online 3d bin-packing, 2020.

[11] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*, 135:364–381, 2022.

[12] J. Zhang, B. Zi, and X. Ge. Attend2pack: Bin packing through deep reinforcement learning with attention, 2021.