

# Using Reinforcement Learning to Solve a Variation of the 3D Bin Packing Problem

AnyLogic Conference 2021

**Damien Lopez**

Senior Consultant - Deep Reinforcement Learning Engineer

**Peter Riley**





Senior Consultant - Simulation Developer

**Yi (Joy) Kuo**

Consultant - Simulation Developer

DECISION LAB

# Agenda

	<b>Background</b>	Introduction to Decision Lab
	<b>Problem Statement</b>	Variation of the 3D bin packing problem
	<b>Building the Sim</b>	Use of the AnyLogic Reinforcement Learning Experiment
	<b>Microsoft Bonsai</b>	Training a Reinforcement Learning Brain to make decisions

# INTRODUCTION

# Introduction to Decision Lab

- Decision Lab is an award-winning technology company that solves real-world challenges for some of the biggest and most respected businesses in the UK.
- Microsoft Autonomous System Integration Partner



We are experts in

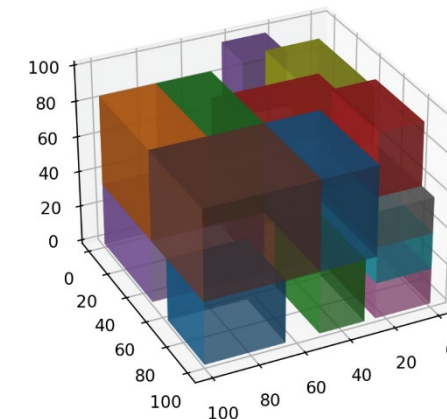
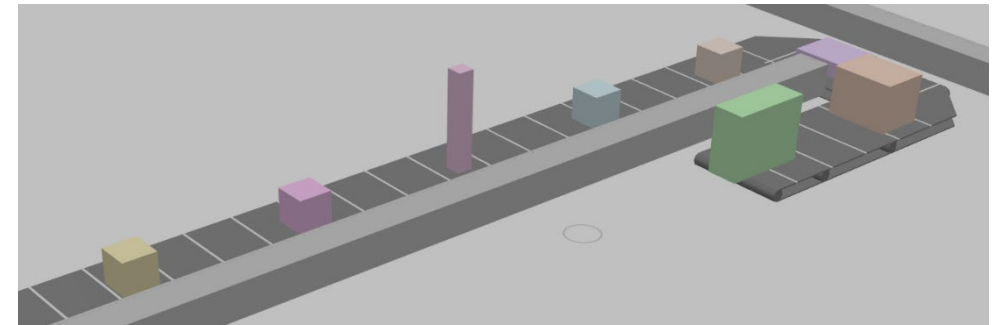
- Simulation,
- Reinforcement Learning,
- Mathematical Modelling,
- Optimisation,
- Data Science.

# Problem Statement

**Standard problem:** 3D bin packing places cuboid items into a bin (i.e. a container or pallet), whilst attempting to maximise the utility of the bin.

## Our variation:

- Items arrive in sequence and are not all present at the beginning
- Items must be dealt with individually and in the sequence in which they arrive
- Items should not be placed in a way that would topple over
- The decision-making agent only gets to see 1 box ahead when making the placement decision



# Suitability of reinforcement learning vs. optimisation

- Some variations of the 3D bin packing problem can be solved using optimisation.
- Decision Lab have worked on another project to create a Logistics Optimisation (LOGOS) for effective item packing in Deployed Military Logistics Hubs.



## Reinforcement Learning

- Explores uncharted territory and can learn appropriate actions for a range of scenarios.
- Deals far better with delayed rewards
- Can handle imperfect information in uncertain environments

## Traditional optimisation

- Isn't appropriate when not all items are visible at the beginning
- Relies on perfect information being available at the start of the planning process
- Is sub-optimal the minute the real-world deviates from the plan



# THE SIMULATION

# Simulation Agents

**Items** are created and queued for packing

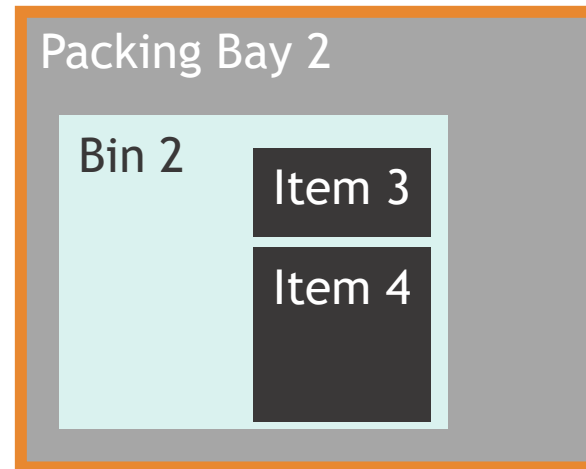
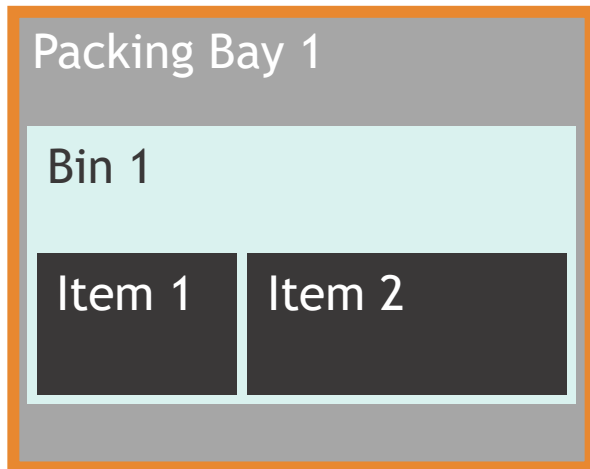
- Can be generated through user input for a fixed amount of items with specified sizes
- Can be generated randomly with a given range of dimensions

**Bins** parked in packing bays - may be sent off when the bin is full/done packing

- Can specify the length/width/height of bins to pack into
- New bins can be generated and will occupy empty packing bays when available

**Packing Bays** are fixed areas for bins to be packed, and determine the number of bins that can be packed simultaneously

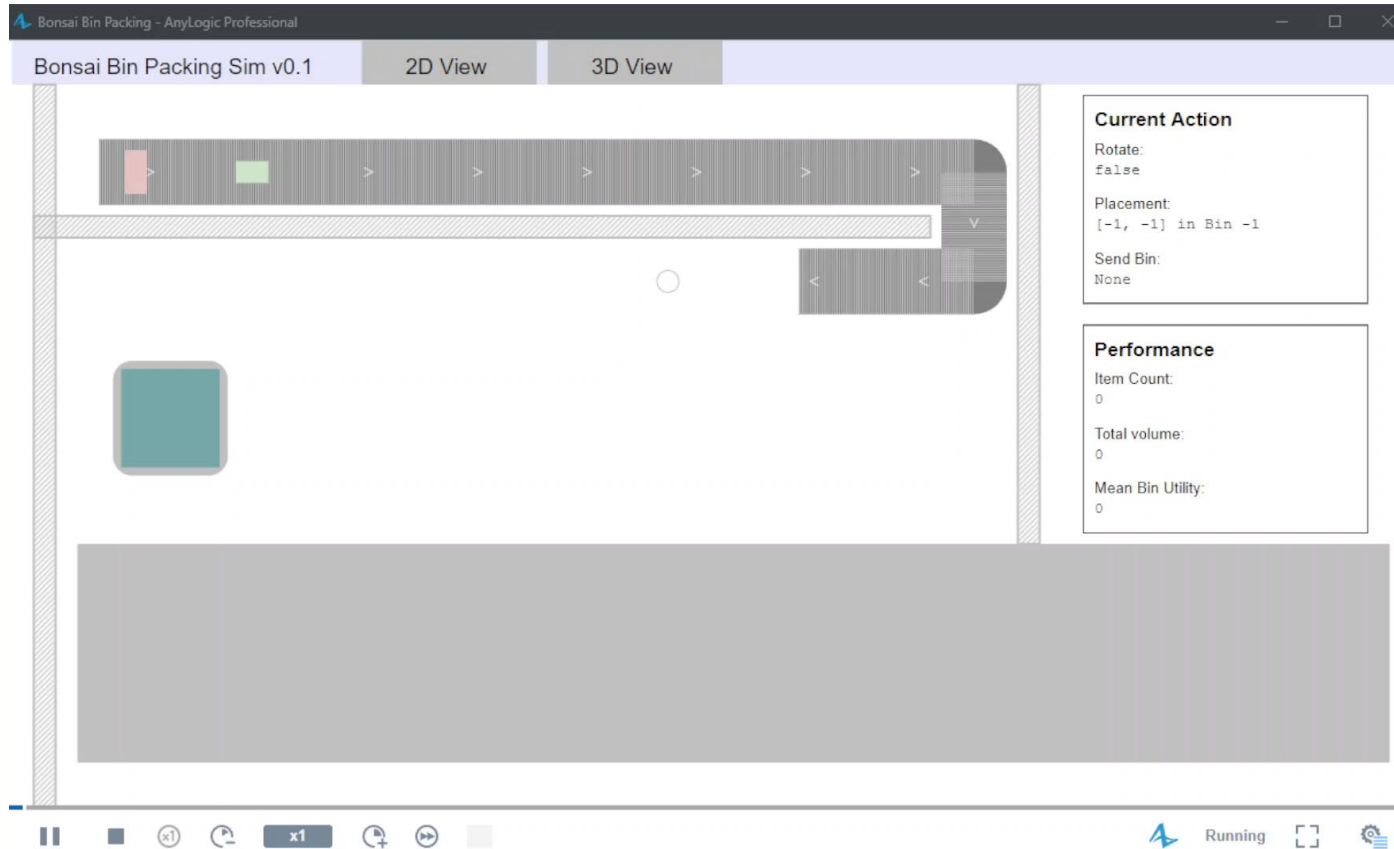
- Only 1 bin per packing bay



- **Actions** are taken to determine:
  - Which bin to pack into
  - The (x, y) coordinates and horizontal rotation for the item to be placed (item always placed from the top, dropping down)



# Visualisation



- 2D and 3D views
- Solid items + transparent bins to allow view of item positioning and stacking
- Current action details and run performance displayed



# What is Bonsai and Why did we use it?

- Microsoft Bonsai is a complete toolchain to build, train and deploy Reinforcement Learning Brains.
  - Part of the Microsoft Autonomous Systems Team
- It allows for ease of use for Subject Matter Experts without a machine learning background to program their expertise directly into an AI model and generate controller to produce optimised control actions.
- It allows for easy integration with various simulators.
  - Including AnyLogic
- All that is required from the user is to create the simulator, specify the reward function/goal of the agent, and train.
  - The DRL algorithm utilised can be even be chosen automatically
- Trained online using Microsoft azure and can be exported to be deployed offline to the real world environment.

# Run Modes

## User Placement

- User **specifies item specifications** (length, width, height) and item **arrival order**
- User **specifies item placement (action)**, including placement xy coordinate, item rotation, and bin sending

## Rule-based

- User **specifies configuration** in the simulation parameters
- Items are placed based on rule of first available position of existing items' adjacent positions (first item starts in the far-left corner (0,0))

## Bonsai Run Modes

### RL Training

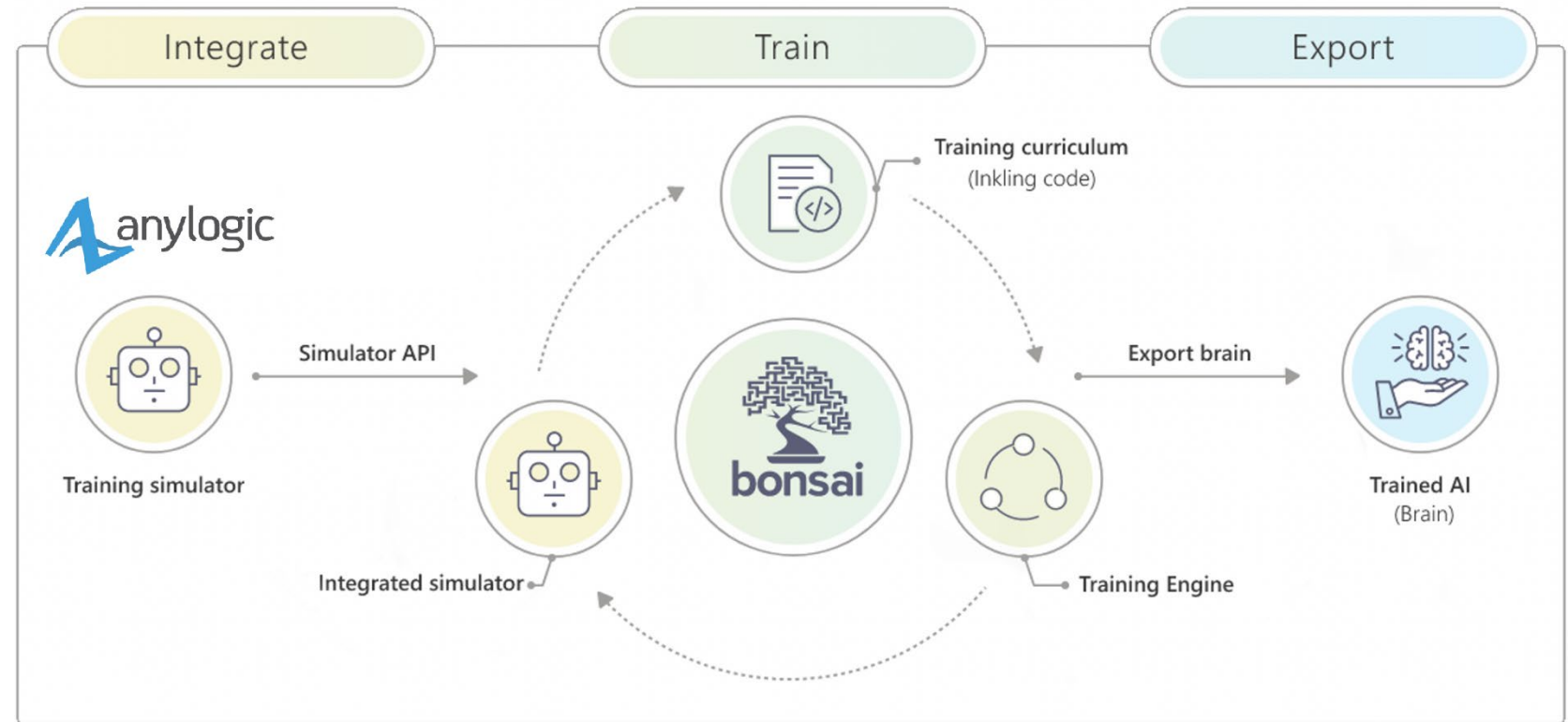
- Configuration specified through **Bonsai Inkling**
- Items are placed based on Bonsai brain's action
- Training can be visually observed if running the training simulation locally

### RL Playback

- Trained Bonsai brain (from RL Training) can be **exported and hosted as an app service**
- AnyLogic connects via the Bonsai Connector to the playback service URL to provide RL actions given simulation states

# Bonsai Integration for RL training

**AnyLogic Reinforcement  
Learning Experiments**  
(RL configuration)  
+  
**Bonsai Library**  
(local training/debugging, trained Bonsai  
brain playback)

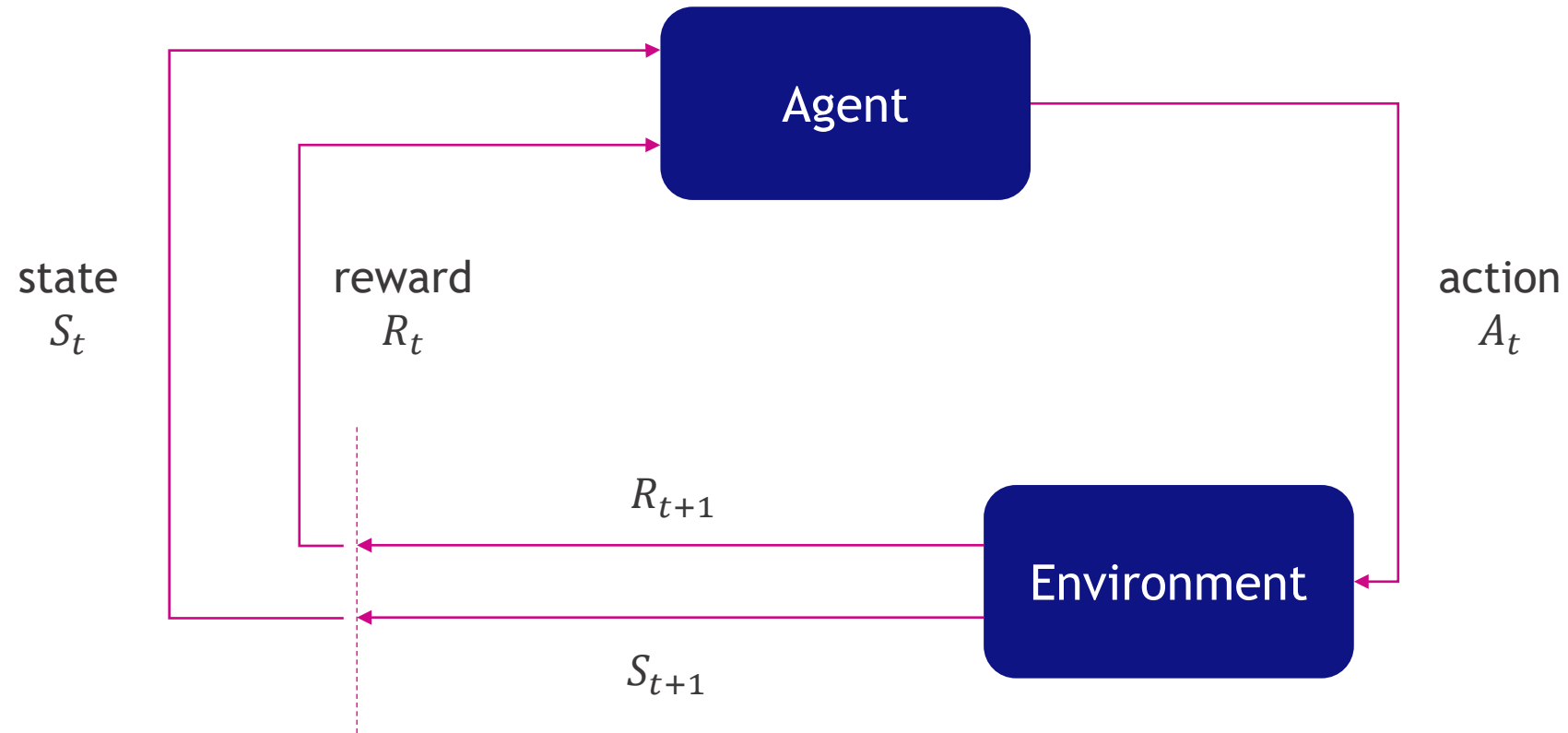


# REINFORCEMENT LEARNING

# When to use RL

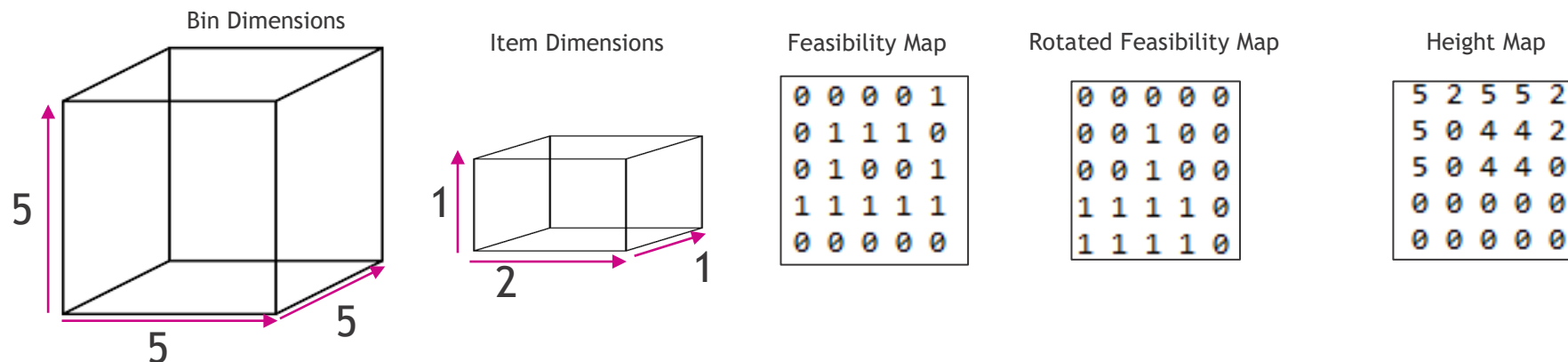
- Reinforcement learning is a control optimisation technique useful when the performance of a sequence of decisions can only be measured after a prolonged period of time.
- In a reinforcement learning scenario the decision made in the present state will impact the decision made in the next.
- It is also applicable in scenarios where context matters, which differs from a normal optimisation approach where the optimal solution is found unconditionally.
- Traditional optimisation works better when all information is available whereas reinforcement learning is better able to adapt to unseen future information and deal with partial observations.
- It is particularly proficient at dealing with stochasticity.
- Also useful when multiple goals need to be optimised or when adaptability to an environment or use case is desired.
- Especially powerful when a simulator is available.

# Overview of Reinforcement Learning



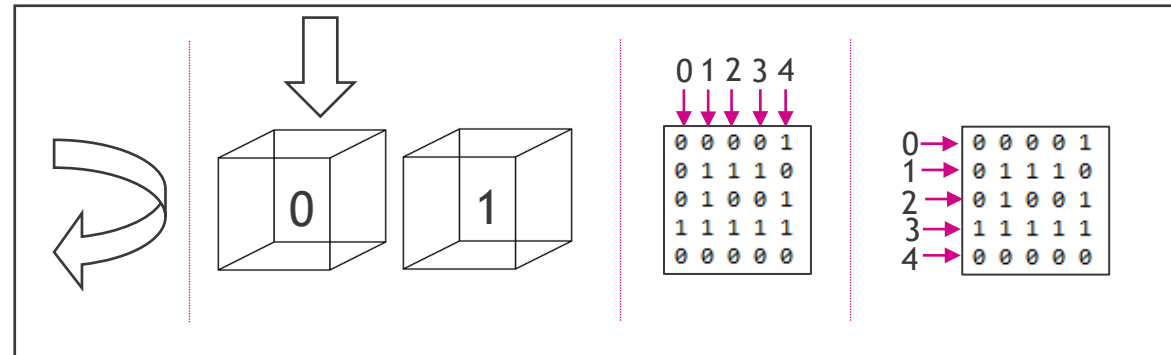
# Configurable Parameters and Observable State

- We can vary parts of the simulation to test out different scenarios, namely the number of items on the conveyor belt, the dimensions of both items and bins and the number of bins available.
- The observations are a combination of the height map of each bin, a measure of the current utility of each bin, the mean bin utility and the feasibility map of placing the current item in the bin in its present orientation. The current, next item and bin dimensions are also given.



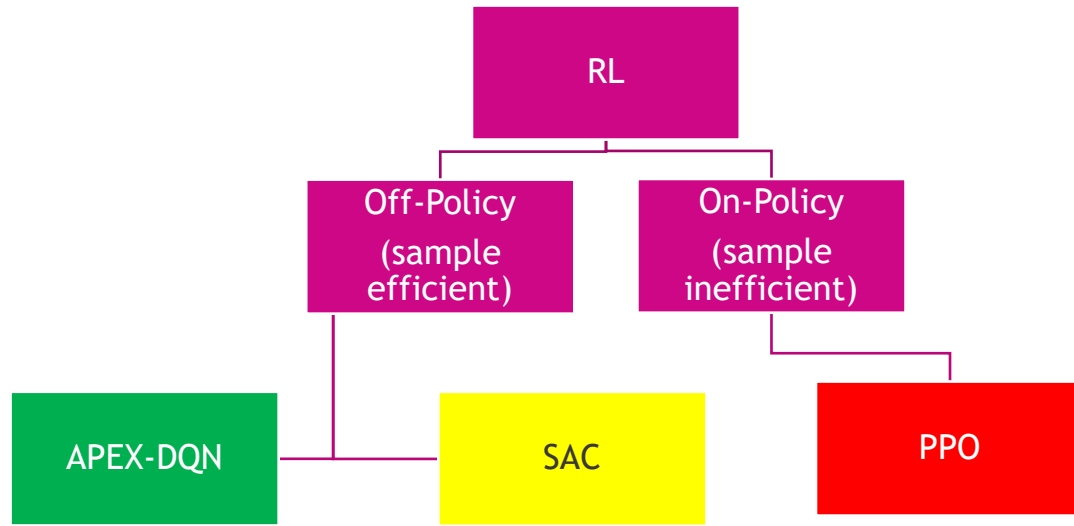


## Actions & Rewards

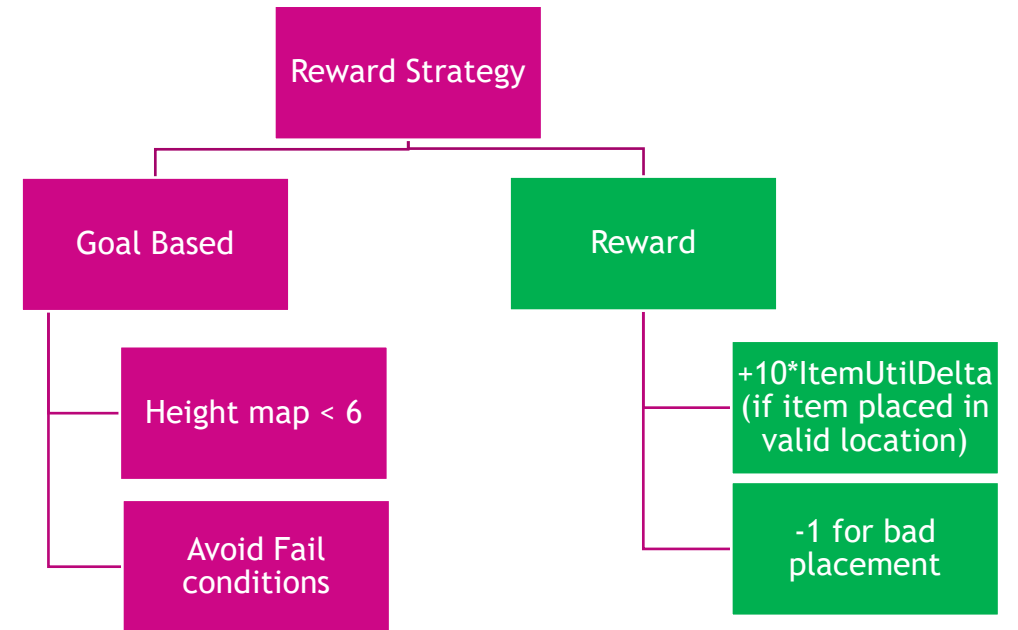


- We use multiple discrete action spaces: rotation, bin placement, x-coordinate and y-coordinate. Rotation rotates the current item 90 degrees, bin placement specifies which bin to place the item in and x and y state which coordinate in the chosen bin to place the item into.
- The Reward is simple as we use a feasibility map conditioned on whether the item was rotated or not. The agent is rewarded by the change in bin utility (multiplied by a factor of 10) given by the item being placed if placed in a valid position as specified by the feasibility map. If it was not placed on a feasible location the agent gets a penalty of -1 and the episode is terminated.

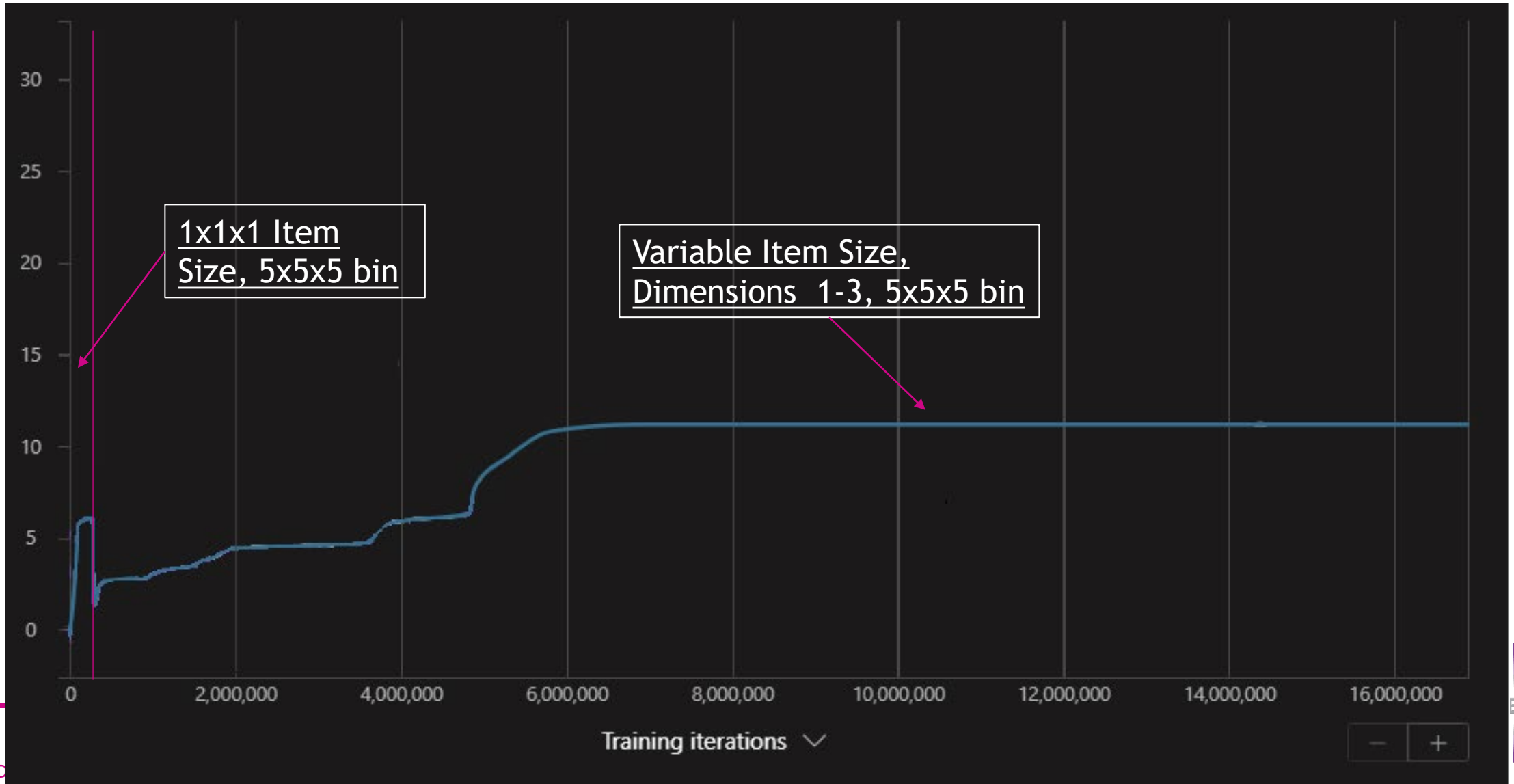
# Training Algorithm



# Reward Strategy



## Lessons and Performance - Reward Based



# BENCHMARKING & PERFORMANCE

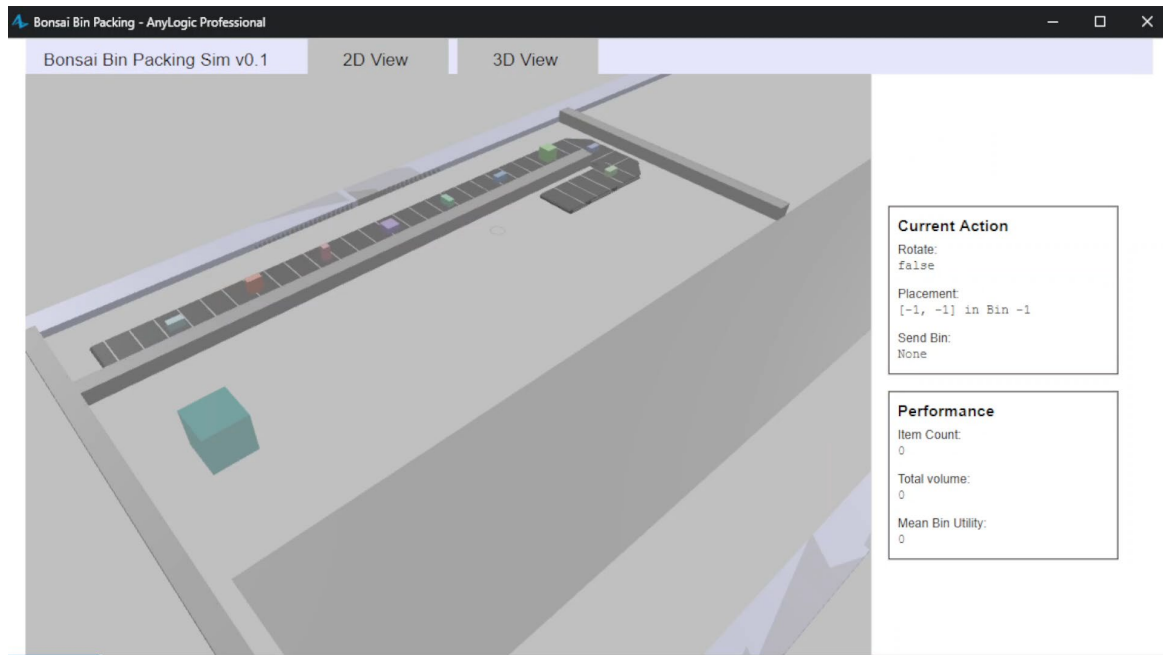
# Benchmark against rule-based algorithm

Scenario	Action Type	Total packed items	Total packed volume
Item size: 1 x 1 x 1 Bin size: 5 x 5 x 5 Total items: 100	Rule-based	100	100
	Bonsai Brain (RL)	100	100
Item size range: [1 - 2] Bin size: 5 x 5 x 5 Total items: 100	Rule-based	25.24	83.56
	Bonsai Brain (RL)	80.08	272.12

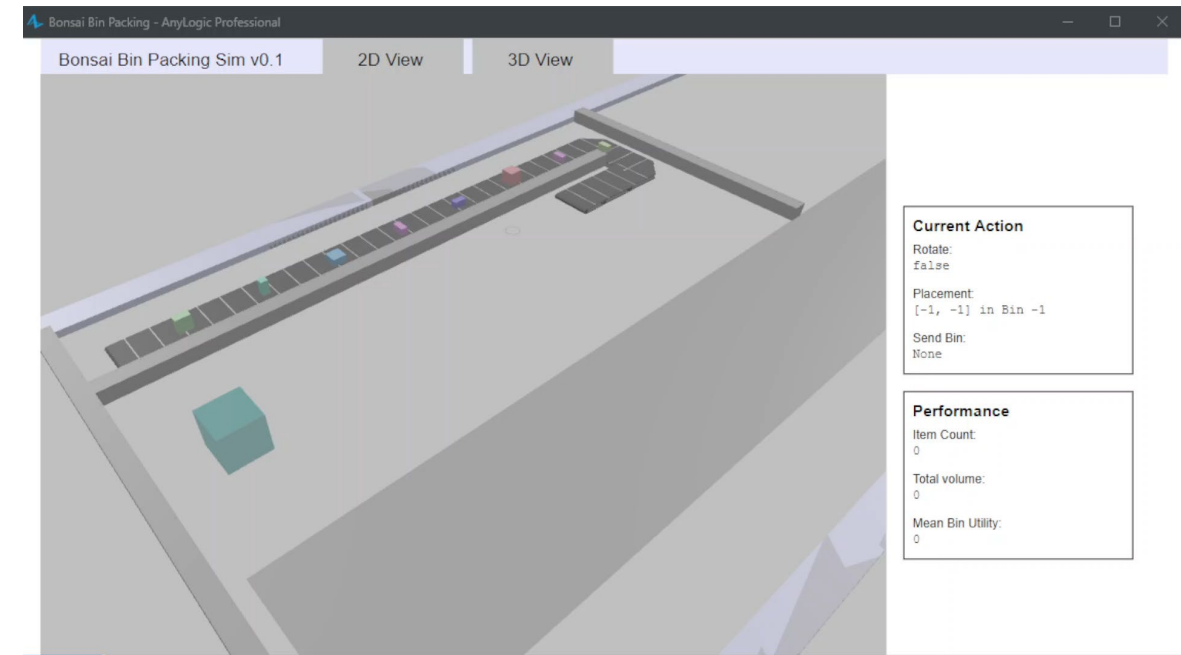
*\*experiment conducted with random sampling of item sizes over 25 episodes for each scenario configuration*

# Item size range [1-2]: rule-based vs. RL agent

## Rule-based



## RL agent



PREPARED BY

**Damien Lopez**

Deep Reinforcement Learning Engineer  
damien.lopez@decisionlab.co.uk

**Peter Riley**

Simulation Developer  
peter.riley@decisionlab.co.uk

**Yi (Joy) Kuo**

Simulation Developer  
yi.kuo@decisionlab.co.uk

V301 Vox Studios  
1-45 Durham Street  
London  
SE11 5JH

Phone: 020 3735 8580

Email: hello@decisionLab.co.uk

# THANK YOU