Reinforcement Learning for Structural Block Assembly

Rayan Gauderon, Tom Stanic, Andrej Kotevski, Alexandre Misrahi

Oskar Dabkowski, Mouhamad Rawas, Leonardo Martella

`Group: BlockRL`

Abstract

In this project, we investigate the application of reinforcement learning (RL) algorithms to 2D structural block assembly tasks. We use a physics-based simulation environment in which agents are trained to reach target points in space by building stable structures such as bridges and towers, and strategically placing blocks of various shapes. A central challenge in this domain is the presence of a vast and highly constrained action space, where most actions lead to unavailable, colliding or unstable configurations. To address this, we explore action masking as a strategy to filter out infeasible actions before policy sampling, as well as various types of rewards and state representations. We evaluate multiple RL algorithms including REINFORCE, DQN, and PPO. Our results demonstrate that action masking significantly improves learning efficiency and task performance, with Masked-PPO outperforming all other approaches. The underlying environments explored in this study act as simulation interfaces for equipping robots with algorithms to assemble blocks in a stable way. In this report, we outline the design decisions for our algorithms and analyze experimental results, contributing to research on learning policies that simulate real-world robotic construction tasks. **Our code is available at** `https://github.com/rayangdn/BlockAssembly`.

1 Introduction

Block assembly tasks provide a challenging domain for RL, blending aspects of robotics, physics simulation and structural reasoning. In this project, we aim to train RL agents to autonomously construct stable structures (e.g. bridges, towers) by strategically placing rigid blocks in a simulated environment. An existing implementation of the environment is provided by the Swiss Data Science Center (SDSC), which simulates rigid body physics and supports complex structural tasks.

These tasks introduce several challenges that require careful design. An episode ends when the agent *reaches* the target, i.e., places a block (a two-dimensional polygon) that contains the target point. If multiple targets are present, all targets need to be reached for an episode to end. This entails **sparse rewards**: the agent receives meaningful feedback only upon successful completion of complex, multi-step goals. We therefore experiment with smoother rewards to improve learning efficiency, which however do not take into account the highly constrained nature of the environment. Indeed, blocks must be placed without violating stability or collision conditions and the structure must remain upright at all times, which entails **complex physics constraints**. At every step, the agent must select an action from a **large action space** by choosing where and how to place a block, with multiple possible shapes, orientations and attachment options. Among these, multiple lead to **numerous infeasible actions** that lead to unavailable, unstable, or otherwise colliding configurations.

To address these issues, we expand upon the provided environment by introducing an action masking mechanism that dynamically filters out infeasible actions substantially improving the agent's learning efficiency.

This report presents the details of our environment design, learning setup and algorithmic choices. We compare standard RL approaches like DQN and PPO with masking-augmented versions and highlight how incorporating domain knowledge through action filtering significantly boosts performance.

2 Related Work

Reinforcement Learning. RL is a framework where agents learn to make decisions by interacting with an environment to maximize cumulative rewards. Classical RL algorithms, such as Q-learning and policy gradient methods, have been foundational in this field (Sutton and Barto, 2018). The integration of deep learning with RL, known as Deep Reinforcement Learning, has enabled agents to handle high-dimensional inputs, leading to breakthroughs in areas like game playing and robotics (Mnih et al., 2015).

Action Masking in RL. In environments with large or constrained action spaces, many actions may be infeasible or suboptimal. Action masking is a technique used to prevent agents from selecting such actions, thereby improving learning efficiency. Huang and Ontañón (2022) provided a theoretical justification for action masking in policy gradient algorithms, demonstrating its importance as the proportion of infeasible actions increases. Tang et al. (2020) proposed incorporating action masks into the Proximal Policy Optimization (PPO) algorithm, showing that this integration leads to higher returns and more efficient learning in environments with infeasible actions. Furthermore, while traditional action masking relies on predefined rules, recent approaches aim to learn **state-specific action masks**. Wang et al. (2023) introduced a method to learn action masks that filter out both infeasible and redundant actions based on bisimulation metrics. This approach allows for dynamic adaptation of the action space, improving policy learning in environments with complex constraints.

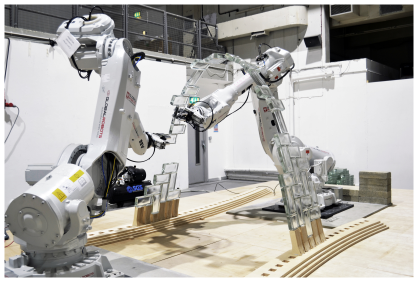
Structural Assembly. Structural assembly tasks present challenges due to physical constraints and combinatorial action spaces. Kim et al. (2021) proposed a DRL framework for 3D combinatorial construction, incorporating action masking to enforce physical constraints like stability and collision avoidance. Their results demonstrated that action masking significantly improves the feasibility and efficiency of learned assembly policies. The literature also contains **stability-aware designs**. Designing stable structures requires considering physical constraints. Kao et al. (2022) introduced the Coupled Rigid-Block Analysis (CRA) method, which combines equilibrium and kinematic constraints in a penalty formulation to assess the stability of complex 3D assemblies. This CRA method is integrated into our environment to provide stability assessment which accurately simulates physical constraints.

3 Methodology

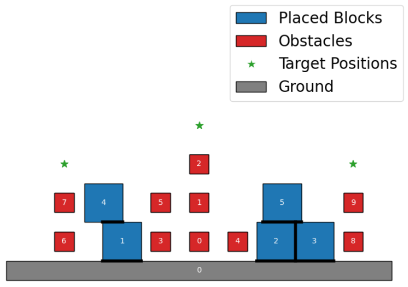
In this section, we describe the design of our block assembly environment, the representation of states and actions, the reward structure and our action masking strategy.

3.1 Environment Design

The environment consists of a 2D physics-based simulation for structural block assembly tasks. An example environment is shown in Fig 1b, along with a real-life scenario in Fig 1a where RL-guided robots assemble blocks. The environment is built around rigid body equilibrium principles. The engine simulates rigid body dynamics to ensure placed blocks remain stable. At each time step, the action space includes a variety of block shapes, including cubes, trapezoids, hexagons and wedges. Each task involves reaching the specific target positions by constructing appropriate structures. In other words, we start off by training one policy per environment, where an environment consists of a set of targets, obstacles, available blocks, and the targets and obstacles' positions in space. Note that generalizing to moving targets is explored later on. Overlapping blocks are penalized and placements that violate spatial constraints are rejected. Agents receive rewards based on how closely a placed block aligns with the target and penalties for infeasible actions.



(a) Real-world environment with robots assembling blocks. *Credit: Stefana Parascho, CRCL.*



(b) Environment visualization with targets (green), placed blocks (blue) and obstacles (red).

Figure 1: Real-world environment and virtual environment representation.

3.2 Action Space Representation

Each action corresponds to a decision about how and where to place a new block. The action vector is defined as a cross product over the following sub actions.

- **Target block:** A previously placed block, to which we attach the new block (typically up to 10 possible choices, depending on how many blocks the agent already placed).
- **Target face:** The face of the target block where the new block will be connected (usually 4 possible faces).
- **Shape selection:** Which type of block, i.e., shape of the new block, e.g. square (typically 2 possibles shapes in our experiments).
- **Attachment face:** The face of the new block that will be used to connect (usually 4 possible faces).
- **Offset:** A horizontal shift along the contact face to offset the placement position (typically 5 discrete offsets in our experiments, this is a parameter that can be modified).

In the worst case, this yields an action space of size 1600 in our experiments. Note that, given this approach, most of the action space results in infeasible actions. These fall into two categories:

1. **Unavailable actions:** actions that our action-enumeration procedure never lists (e.g., positions or attachments that are not geometrically possible in the current state)
2. **Invalid actions:** actions that violate physical constraints (such as unstable placements) or cause collisions with obstacles.

3.3 Reward Shaping

The reward function is designed to guide the agent toward effective and stable construction. Fig. 2 shows our Gaussian-shaped reward centered on the target position encourages precise placements. A penalty is applied for actions that result in unstable or collisions placements. Furthermore, the reward is clipped based on the vertical height to discourage placing blocks far above the target.

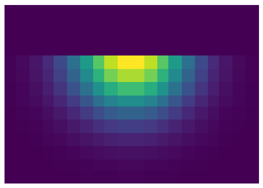
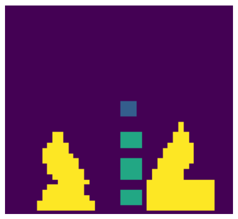


Figure 2: Cropped-gaussian reward centered on target position.

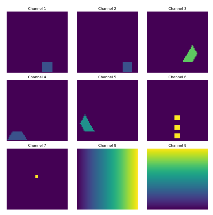
3.4 State Representation

To capture the configuration of the environment, we represent the state using multi-channel images. We focused on three main encoding strategies in a 64×64 grid (see Fig 3). In the first, rudimentary, case we encode the block, reward, and obstacle spatial information into a **single channel** (Fig. 3a). Our second representation is a **4-channel image**, where the first and second channels encode the block spatial information together with the block index and the face index of the bottom face of the same block, respectively. The third and fourth channels encode the spatial information of the targets and obstacles, respectively. Finally, we experiment with another multi-channel representation (Fig. 3b) with a **separate channel for each block** encoding its spatial information together with the face index of their bottom side. This representation includes information about the targets and obstacles in the same fashion as the second representation.

As we work exclusively with convolutional neural networks, which are translation equivariant and are insensitive to absolute position, we further experiment with the approach introduced in Liu et al. (2018) which adds two additional channels that provide explicit positional information to the network.



(a) Single-channel encoding



(b) multi-channel encoding

Figure 3: State representation visualization to encode environment elements.

3.5 Action Masking

One of the key innovations in this project is the use of action masking to improve learning performance (see Fig. 4). Our action masking pipeline begins by enumerating all geometrically plausible actions from the current state. Each action then undergoes a validity check to determine feasibility, including availability and potential for collisions or instability. This generates a binary mask where valid actions receive a value of 1 and infeasible actions receive 0. During action sampling, this mask suppresses infeasible choices, enabling the agent to focus exclusively on feasible options.

In practice, computational constraints limit our feasibility checks to availability rather than full physics-based validation. While comprehensive stability and collision detection for all potential actions would be computationally prohibitive, we employ a fast geometric filter and allow the agent to learn to avoid unstable or colliding placements through trial and error. This approach balances computational efficiency with learning effectiveness.

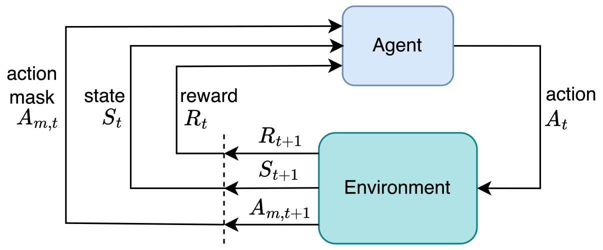


Figure 4: RL framework with action masking: The agent receives the current state and a mask of valid actions from the environment.

4 Experiments

We evaluated four reinforcement learning algorithms on the structural block assembly task: DQN, PPO, Masked-REINFORCE and Masked-PPO (further outlined in Tab. 1). Each agent was trained in several of our environments using the same set of block shapes and target configurations. The goal was to assess the impact of action masking and compare learning efficiencies across algorithms. We proceed by describing each algorithm.

4.1 Algorithms

Deep Q-Learning **(DQN)** (Mnih et al., 2013) is a value-based approach using Q-learning and experience replay. The action space is fully discretized, but the abundance of infeasible actions leads to poor exploration efficiency. Proximal Policy Optimization **(PPO)** (Schulman et al., 2017) is a policy gradient method with a clipped objective to stabilize updates. It improved on DQN but still struggled due to the large proportion of infeasible actions. REINFORCE (Williams, 1992) with masking **(Masked-REINFORCE)** is a vanilla policy gradient algorithm augmented with action masking. This eliminated infeasible actions from consideration during sampling and acts as a baseline for our masking strategy. Finally, **Masked-PPO** modifies the PPO implementation to incorporate dynamic action masking. This approach yielded the best results in both training efficiency and final task performance.

Table 1: Comparison of Reinforcement Learning Algorithms

Algorithm	**Approach**	**Comments**
DQN	Value-based	Struggled with infeasible actions, inefficient exploration.
PPO	Policy gradient	More stable learning, but impacted by large action space.
Masked-REINFORCE	Policy gradient with masking	Reduced infeasible actions, moderate improvement.
Masked-PPO	Masked policy gradient	Best performance, significantly more efficient learning.

4.2 Neural Network Architecture

For Masked-REINFORCE, we implemented a custom policy network consisting of a 2D convolutional layer (16 output channels, 3×3 kernel, stride 1, padding 1) followed by ReLU activation, a fully connected layer with 64 hidden units, another ReLU activation, and a final linear layer that projects to the action space dimensions. For PPO, Masked-PPO and DQN, we employed the default `MlpPolicy` architectures provided by Stable-Baselines3[1].

4.3 Training

Fig. 5 shows the reward collected during training for one full episode. DQN does not perform well, likely due to the sparsity of the reward. PPO without masking seems to struggle to generate rewarding policies at first, but after 60k training steps it seems to have converged to some local minimum and generating some reward. Masking does seem to improve the learning ability of the agent, since Masked-REINFORCE converges much faster than PPO without masking (still to a local minimum). In addition, it is clear that the Masked-PPO agent achieved the highest rewards and converged faster than other agents.

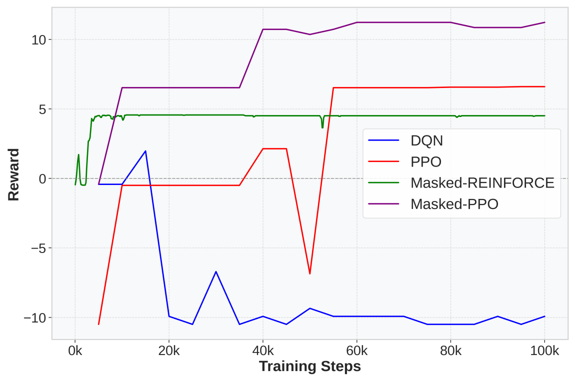


Figure 5: Training reward over time for different algorithms. Masked-PPO shows fastest convergence and highest final performance.

[1]SB3 PPO MlpPolicy and SB3 DQN MlpPolicy

4.4 Evaluation Metrics[2]

We additionally provide three metrics: **Unavailable action rate** measures how frequently the agent attempts to select actions that are not geometrically feasible in the current state. **Invalid action rate** measures the frequency of agent actions that result in invalid placements. **Reward collected** denotes the cumulative reward earned by the agent during one evaluation episode. Tab. 2 summarizes these metrics for all considered models. We observe that PPO seems to have a better inductive bias in learning to select feasible actions, compared to DQN. This is directly translated into a greater cumulative reward (6.6 vs 2.0). In fact PPO has a greater cumulative reward than Masked-REINFORCE, even though the latter has 0% unavailable action rate due to masking, which highlights the importance of the training algorithm used. Finally, we confirm the significant performance boost when using masking, since Masked-PPO shows a +4.6 reward gain compared to PPO.

Table 2: Evaluation Set Performance Metrics for Each Algorithm

Algorithm	Unavailable Action Rate	Invalid Action Rate	Reward Collected
DQN	88.9%	63.5%	2.0
PPO	27.3%	43.0%	6.6
Masked-REINFORCE	0%	41.3%	4.6
Masked-PPO	**0%**	**24.7%**	**11.2**

When looking at the learned policies in practice, we observe that the Masked-PPO agent successfully learned to build complex structures in three different environments (Fig. 6). The agent demonstrates strong performance on the single bridge task, successfully constructing a functional bridge structure. However, performance gaps emerge with more challenging configurations. In the double bridge environment, while the agent reaches the target position, it fails to form the complete bridge structure. Similarly, in the tower environment, the agent struggles to reach the target, leaving the construction incomplete.

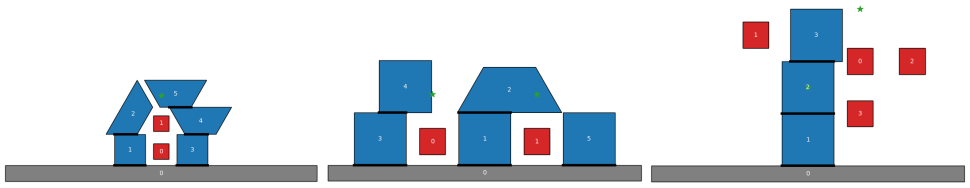

(a) Bridge (b) Double Bridge (c) Tower

Figure 6: Structures built using masked-PPO agent for different environments.

4.5 Experiments on reward shaping and positional encoding

Cropped-gaussian reward is a better solution for reward shaping than a sparse reward located only at the target point. While it yields better results, it still suffers from poor propagation. In the Bridge example (Fig. 6a) with two obstacles, one needs to place 2 blocks with a marginal reward (in our examples block number 3 and 4) in order to be able to place a block with high reward (number 5).

To help propagate the reward also to the pieces placed at the beginning, we tested another reward shape: cropped cylindrical reward as shown in Fig. 7. It encourages the algorithm to make constructions as closely to the projection of the target onto the base as possible, but avoiding obstacles. This may be great for the bridges in general: the closer are the bridge's pillars, the more stable the bridge will be.

We tested the new reward shape with the Masked-PPO algorithm on 3 variations on positional encoding, in order to check the expressiveness of different settings:

The results of the experiment can be seen on Fig. 8. For this reward, we can observe that the blocks are placed closer to the obstacles than in the case of cropped-Gaussian, as in Fig. 6.

[2]All results presented use the single-channel state encoding and the cropped-gaussian reward.

- 14-channel representation, with one channel for each block, two channels for spatial information on obstacles and targets and two channels for gradient-like positional encoding (as on the bottom-right of Fig. 3b)
- 6-channel representation, with two channels encoding block spatial information on block and face index respectively, two channels for spacial information on targets and obstacles, and two channels for gradient-like positional encoding
- 4-channel representation, as described above, but without latter positional encoding

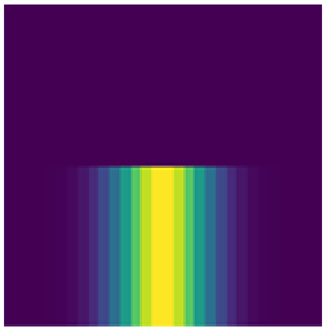


Figure 7: Cylindrical reward centered on target position.

We can notice that the policy based on 4-channel state representation provides the most compact bridge construction - it uses the least number of blocks and they are the most concentrated around the target out of all the ablations. We can conclude that positional encoding should be excluded to get better performance.

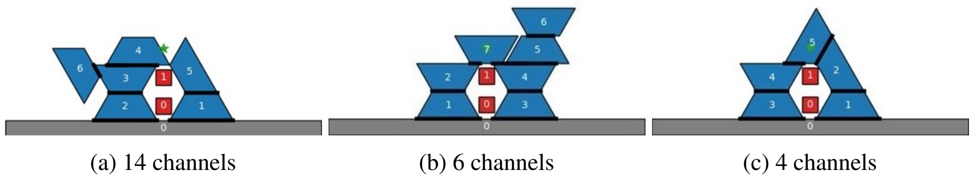

(a) 14 channels (b) 6 channels (c) 4 channels

Figure 8: Structures built using masked-PPO agent for different ablations.

4.6 Generalization through Moving Targets

A key limitation of the current approach is that agents trained on fixed configurations fail to generalize to environments with different target positions. We address this through moving targets during training and multi-channel state representation (see Fig 3).

This representation includes dedicated goal position channels, enabling adaptation to new target layouts without retraining. Here, however, positional encoding channels help the convolutional network overcome shift invariance for location-aware strategies, while orientation encoding provides spatial awareness essential for moving targets.

To enhance generalization, we exploit observation space symmetry. For single-goal environments, we standardize by mirroring scenes to place goals in the left half, with corresponding action mirroring. This data augmentation reduces the state space size and improves generalization.

During training, we periodically shift target positions while maintaining structural constraints. This forces agents to learn generalizable construction strategies rather than memorizing fixed placement sequences, developing more robust policies for novel configurations as demonstrated in Figure 9 .

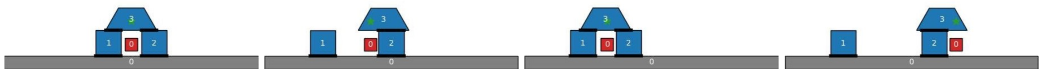


Figure 9: Bridges built using masked-PPO agent with moving targets to improve generalization.

5 Conclusion

In this project, we explored various RL frameworks for structural block assembly tasks in a physics-based simulation environment. We benchmarked standard learning algorithms, and additionally motivated the use of an inductive-bias in the masking strategy, effectively reducing the action space size by approximately 85%. We perform additional ablations and experiments with various types of rewards and state representations, confirming intuitions with concrete examples.

There are several limitations and potential future research directions. In section 4.6, we outline ways to generalize training to different targets for a same set of obstacles. A possible extension of our project would be to generalize to moving obstacles, or even to the most general setting of moving targets *and* obstacles. Ideally, one would want a single trained agent able to generalize to a set of obstacles and target(s). Furthermore, expanding from 2D to 3D block placement will allow for more realistic scenarios and bring us closer to real-world applications. However, this will come with other challenges such as multiplication of the action space size and therefore challenges in efficient learning and generalization. One possible way to address these challenges is curriculum learning, where one gradually increases task complexity which could allow agents to generalize better across different assembly scenarios. Lastly, our current environments are limited to relatively simple structures. One could think of scaling deep networks to learn much more complex building patterns.

Overall, this project outlines various learnings for training RL agents for complex construction tasks.

References

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms. *The International FLAIRS Conference Proceedings*, 35, May 2022. ISSN 2334-0762. doi: 10.32473/flairs.v35i.130584. URL `http://dx.doi.org/10.32473/flairs.v35i.130584`.

Cheng-Yen Tang, Chien-Hung Liu, Woei-Kae Chen, and Shingchern D. You. Implementing action mask in proximal policy optimization (ppo) algorithm. *ICT Express*, pages 200–203, 2020. ISSN 2405-9595. doi: https://doi.org/10.1016/j.icte.2020.05.003. URL `https://www.sciencedirect.com/science/article/pii/S2405959520300746`.

Yue Wang et al. Learning state-specific action masks for reinforcement learning. *Algorithms*, 17(2): 60, 2023.

Seung Wook Kim et al. Combinatorial construction with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 1234–1245, 2021.

Gene Ting-Chun Kao, Antonino Iannuzzo, Bernhard Thomaszewski, Stelian Coros, Tom Van Mele, and Philippe Block. Coupled rigid-block analysis: Stability-aware design of complex discrete-element assemblies. *Computer-Aided Design*, 2022. ISSN 0010-4485. doi: https://doi.org/10.1016/j.cad.2022.103216.

Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution, 2018. URL `https://arxiv.org/abs/1807.03247`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. URL `https://arxiv.org/abs/1312.5602`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992696. URL `https://doi.org/10.1007/BF00992696`.