# Reinforcement learning for scaffold-free construction of spanning structures

Gabriel Vallat
vallatga@gmail.com
EPFL
Lausanne, Switzerland

Jingwen Wang
jingwen.wang@epfl.ch
EPFL
Lausanne, Switzerland

Anna Maddux
anna.maddux@epfl.ch
EPFL
Lausanne, Switzerland

Maryam Kamgarpour*
maryam.Kamgarpour@epfl.ch
EPFL
Lausanne, Switzerland

Stefana Parascho*
stefana.parascho@epfl.ch
EPFL
Lausanne, Switzerland

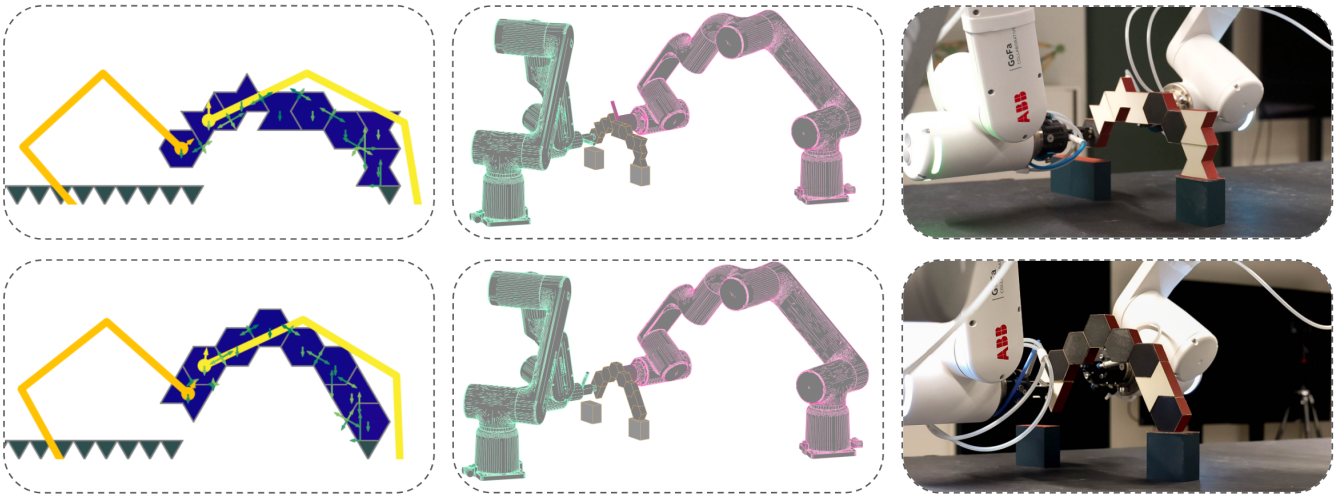


**Figure 1: Multi-robotic assembly of reinforcement learning designed structures**

## ABSTRACT

In construction robotics, a conventional design-to-fabrication workflow starts with designing a structure, followed by task and robotic motion planning, and ultimately, fabrication. However, this approach can prove unsuccessful, as we may only discover the infeasibility of a design at the final stages of the process. This can result in rework and a considerable waste of time and resources. To overcome this challenge, we propose a design method based on reinforcement learning (RL) where the agent makes decisions at every step of the sequential assembly of the structure while considering assembly's stability. In this way, we take the construction constraints into consideration at the design stage. The research particularly focuses on the design of spanning structures that multiple robot arms can construct without the need for scaffolding.

A series of experiments were conducted using both a centralized and a decentralized learning setup. Our results show that while the decentralized setup was successful in constructing smaller structures, only the centralized setup allowed active collaboration between robot arms, resulting in structures with larger spans. To validate our approach, we fabricated two of the designed structures with two collaborating robot arms, which confirmed the feasibility of these designs. In summary, the proposed method opens exciting possibilities for generating innovative designs that push the boundaries of architectural creativity while simultaneously fulfilling fabrication-related constraints.

*Last two authors are equal co-advisers and contributed equally


*SCF '23, October 08–10, 2023, New York City, NY, USA*

ACM ISBN 979-8-4007-0319-5/23/10…$15.00
https://doi.org/10.1145/3623263.3623359

## CCS CONCEPTS

• **Computing methodologies** → *Multi-agent reinforcement learning*; *Neural networks*; Stochastic games; **Cooperation and coordination**; **Intelligent agents**; **Reinforcement learning**; • **Applied computing** → **Computer-aided design**; • **Computer systems organization** → **Robotic autonomy**.

## KEYWORDS

multi-agent reinforcement learning, structured networks, centralized training, soft actor critic, robotic fabrication

**ACM Reference Format:**

Gabriel Vallat, Jingwen Wang, Anna Maddux, Maryam Kamgarpour, and Stefana Parascho. 2023. Reinforcement learning for scaffold-free construction of spanning structures. In *Symposium on Computational Fabrication (SCF '23), October 08–10, 2023, New York City, NY, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3623263.3623359

## 1 INTRODUCTION

Spanning structures refer to structures that create large unobstructed, column-free spaces between their supports. Many of these structures, such as arches and domes, are very material-efficient because they have excellent load-bearing capacity with low material use [Rippmann 2016]. However, the construction of these structures often requires the use of temporary supports called scaffolds or falsework, due to their inherent instability before they are completed. Unfortunately, this reliance on scaffolding significantly increases the material and labor costs of the structure [Rippmann 2016], which hinders the practical construction of spanning structures.

The incorporation of multi-arms offers a promising solution to address this issue. The use of robot arms in construction brings numerous benefits, including increased precision, sustained performance, and increased efficiency [Parascho et al. 2020]. In the context of building spanning structures, robot arms can be used both for assembling the structure and as temporary supports, minimizing the reliance on scaffolding and consequently simplifying the complicated process of erecting spanning structures.

However, the current methodology in robotic construction follows a sequential top-down approach that includes several stages: Design, discretization of the geometry, construction sequencing, robotic path planning, and fabrication of the final structure. This process often leads to the discovery that certain designs end up being unsuitable for manufacturing, resulting in a large amount of rework. To overcome this challenge, we explored an alternative approach: integrating reinforcement learning (RL) into the current workflow. By training agents to complete assigned construction tasks independently, rather than working with specific designs, we aim to create a multi-robotic fabrication-guided design process and revolutionize the existing design paradigm. With this approach, we can save time in design-related processes, reduce the need for rework, and potentially create innovative designs.

In addition, to address scenarios where robot arms act in a decentralized fashion in the construction of spanning structures, we also explored methods of multi-agent reinforcement learning (MARL). Rather than having a single agent control all robot arms, MARL allows each robot arm to be represented by an agent. Particularly, when the number of robot arms is large, MARL could improve the performance of the training in the long run. In our case, we aim to compare the performance of MARL-trained agents with the single agent controlling both arms. This allows us to evaluate what advantages but also challenges arise when MARL is used for decision-making.

In summary, we have used (multi-agent) reinforcement learning to teach robots to build spanning structures without a scaffold. The rest of this paper is organized as follows: Section 2 shows the current state of the art in robotic assembly of spanning structures, reinforcement learning in assembly tasks, and multi-agent reinforcement learning. In Section 3, we explain the training setup of the algorithm in detail and present our simulation environment, problem formulation, and agent design. Training results and discussions are shown in Section 4. Validation of the algorithm (physical construction of the structures generated by the algorithm) is presented in Section 5 and contribution, limitations, and future work are discussed in Section 6.

## 2 RELATED WORK

### 2.1 Robotic Assembly of Spanning Structures

Current research on robotic assembly of spanning structures mainly focuses on two types of structures: bar structures [Bruun et al. 2022; Huang et al. 2021] and discrete element assemblies [Frick et al. 2015]. Bar structures consist of line elements, and the main difficulty in their construction is to make connections at the joints of these line elements. In contrast, discrete elements consist of individual rigid units, and the connections between each unit rely on either cohesive materials (e.g., adhesives) or pure friction, as in unreinforced masonry structures. This paper deals with the latter type of spanning structures.

Previous studies, including [Parascho et al. 2020; Wang et al. 2023; Wu and Kilian 2020], have addressed robotic assembly of spanning discrete elements. However, as highlighted in the introduction, these works adhere to a sequential design, planning, and manufacturing process. Although some works incorporate iterative design processes to improve performance and constructability, they are still constrained by the limitations of this top-down paradigm. In contrast, our work aims to move away from this paradigm by prioritizing design goals over specific designs. By using RL algorithms to determine the optimal placement of each subsequent block, we enable the creation of innovative and unconventional designs. This approach opens new possibilities for architectural exploration and pushes the boundaries of what can be achieved in the field of robotic construction.

### 2.2 Reinforcement Learning in Assembly Tasks

Reinforcement learning (RL) has gained much interest in the last decade. Indeed, RL algorithms have already proven useful in several fields, from biochemistry [Jumper et al. 2021] to computer code generation [Li et al. 2022]. Since RL optimizes a reward function by trying different possible actions, a major problem is the large number of trials the agent must perform before obtaining a good solution. The recently proposed *Soft Actor Critic* (SAC)[Haarnoja et al. 2018a] addresses this problem by maximizing the randomness of the action choice. This behavior generates a variety of different good examples for the agent to train with. In comparison, older methods such as *Advantage Actor Critic* (A2C) [Mnih et al. 2016] explore randomly rather than focusing on promising leads. Shortly after SAC was proposed, it was shown to be more efficient to keep the randomness at which the action was performed at a target value rather than simply maximizing it [Haarnoja et al. 2018b].

This method reaches state-of-the-art results when trained to play Atari games [Christodoulou 2019]. In our experiments, we use the SAC and the A2C algorithm as the underlying (MA)RL frameworks for the task of scaffold-free construction of a spanning structure.

In construction, RL has previously been used to solve different tasks, such as assembly [Belousov et al. 2022]. RL has also been used with less constrained construction, where the agent is tasked with putting together complex shapes in order to follow a line as closely as possible [Wibranek et al. 2021]. In an experiment similar to ours, [Bapst et al. 2019] showed that RL can be efficiently used to design simulated structures by stacking rectangular blocks and using a graph-based neural network [Battaglia et al. 2018; Bronstein et al. 2021] as an internal model. Compared to [Bapst et al. 2019], we use blocks with a more complex shape and consider the task of linking two separate grounds at the same level. The resulting structure, called spanning, needs to take an arch-like shape, which is more complex than simply stacking blocks. Furthermore, to the best of our knowledge, we are the first to explore using a MARL framework in the robotic assembly of a spanning structure.

## 2.3 Multi-agent Reinforcement Learning

As discussed in the Introduction, since constructing a spanning structure requires at least two robot arms, we investigate multi-agent reinforcement learning (MARL) methods. Under the centralized training [Lowe et al. 2020], it was shown that common reinforcement learning (RL) algorithms can be adapted to a multi-agent setup. This method, however, considers that all agents share information during training, allowing each agent to use a commonly trained model for action selection. In our work, we instead try to see whether a fully decentralized framework still allows for the collaboration of the different agents.

# 3 TRAINING SETUP FOR CONSTRUCTION OF A SPANNING STRUCTURE

In this section, we consider two robot arms tasked with building a self-supporting spanning structure. To this end, we develop a simulator that models the environment in which the structure is built and can assess whether the structure is stable. Furthermore, we model the construction process of the spanning structure as a Markov decision process (MDP) [Puterman 1994] when we consider a centralized setup and as a Markov game [Shapley 1953] when we consider a decentralized setup.

## 3.1 Simulation environment for construction of a spanning structure

For ease of exposition we model the environment in which two robot arms aim to build a self-supporting arch as a two-dimensional grid. Since arch-shaped structures are hard to be built by merely using rectangular blocks without adhesive materials, an isometric grid composed of equilateral triangles is considered (see Figure 2). In this environment, the robot arms can place hexagonal blocks, hold the placed blocks, and apply force to them to maintain the created structure. To speed-up the simulation and to check whether the simulated structure is stable, only the static stability of the structure is computed rather than taking into account the dynamics of placing a new block. Computing the static stability follows the linear program formulation found in [Frick et al. 2015]. Under this formulation, a structure is considered to be stable if there exists a combination of compressive and friction forces which result in no acceleration. The detailed formulation of the simulator can be accessed in the supplementary material[1]

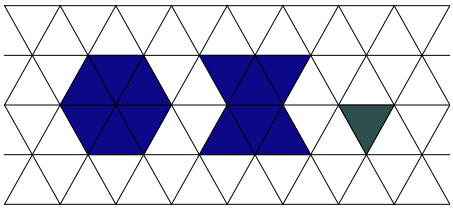


**Figure 2: From left to right in the grid: hexagon, hourglass (only used in experiment 2) and ground**

Concretely, we consider the task of linking two rows of triangles separated by a gap ranging from one to seven triangle-widths in the isometric grid (see Figure 3 for an example). The robot arms are allowed to place the hexagonal blocks against any of the already present elements in the grid which correspond to either the two ground bases or the previously placed blocks. If the two robot arms manage to build a stable spanning structure, a reward is obtained, whereas if the structure collapses or both arms simultaneously place a block at the same place, a cost is incurred. To further encourage the robot arms to connect the two bases and thereby construct a spanning structure, a reward is obtained whenever the distance in free air between the two bases is reduced.

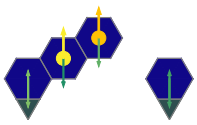
## 3.2 Formulating the spanning structure construction as a Markov decision process and a Markov game

*3.2.1 Markov decision process.* If the two robot arms are controlled by a central agent abbreviated by agent, the above setup can be described as a Markov decision process (MDP) [Puterman 1994]. An MDP is a tuple

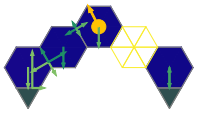
$$\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle,$$

where $\mathcal{S}$ is a set of environmental states, $\mathcal{A}$ is the set of agent's possible actions, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition dynamics which determines the next state given the current state and action, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function which specifies the reward obtained by any state-action pair, and $\gamma$ is a discount factor which represents the value of time. In the above setup, the set of states corresponds to all possible configurations of blocks on the isometric grid combined with an indicator indicating which robot arm will take an action next. The action set of the central agent consists of all possible ways to connect a new block to the current structure and then hold the block in place for one round[2]. Given the current block structure and the action taken by one of the robot arms the structure transitions to the next state, where the other robot arm will take an action next. Furthermore, the next state corresponds to either the current structure plus an added block or a collapsed structure if the chosen

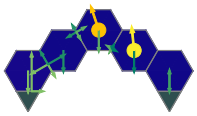
[1] https://github.com/Fllask/RLSFCSS/blob/main/Physics_simulator.pdf
[2] The exact definitions of the elements of the MDP for the above setup were omitted for ease of understanding.



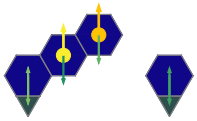
**(a) Initial situation**



**(b) The yellow robot wants to place a new block**



**(c) Result**



**(d) Initial situation**

**(e) The orange robot wants to place a new block**

**(f) The action cannot be performed**

**Figure 3: On the top row, the yellow robot is able to leave the block it is holding in 3a as the orange one is applying an oblique force stabilizing the structure. On the bottom row, the orange robot tries to leave. As the yellow one cannot stabilize the structure alone, the action cannot be performed and the simulation is ended. Note that on Figures 3e and 3f, one of the possible failure mode is highlighted with red arrows, showing which additional forces would be required to stabilize the structure.**

action causes the structure to become unstable. The reward the central agent obtains is defined as in Section 3.1. Namely, choosing an action that completes a stable spanning structure or reduces the distance between the two ground bases given the current structure leads to a positive reward. Choosing an action that results in a collapse of the current structure leads to a negative reward. Solving the MDP corresponds to finding an optimal policy that maximizes the reward over time. Assuming the reward function is specified such that it is aligned with the construction task, then if the central agent follows the optimal policy by taking a specific action given the current state a stable spanning structure is obtained. More formally, a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ maps states to a distribution over the action set. A policy $\pi^*$ is said to be optimal if it maximizes the expected discounted cumulative reward or so-called value function

$$V(\pi, s_0) := \mathbb{E}_{s_{t+1}=P(\cdot|s_t,a_t)}\left[\sum_{t=0}^{\infty} \gamma^t R_t(s_t, a_t)\middle| a_t \sim \pi(\cdot|s_t), s_0\right].$$

Hereafter, we refer to the setting where a central agent controls the two robot arms as the centralized setup.

*3.2.2 Markov game.* If the two robot arms are controlled by two independent agents, the setup can be described as a Markov game. A Markov game is a tuple

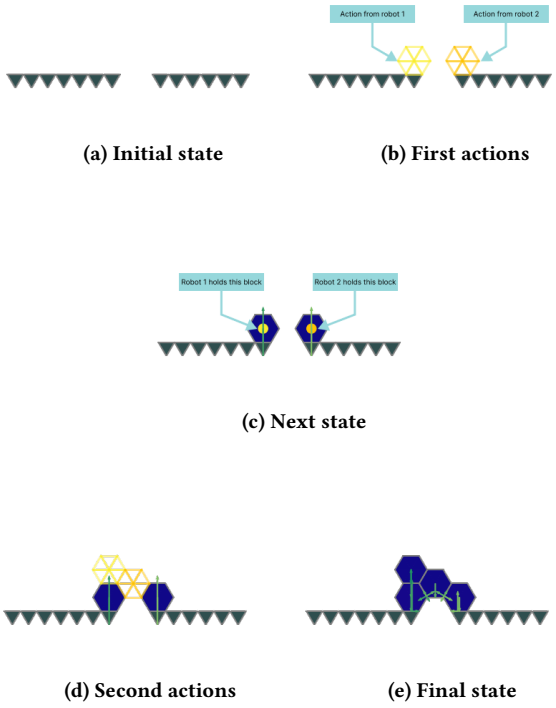
$$\langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, P, \{R^i\}_{i\in\mathcal{N}}, \gamma\rangle,$$

where $\mathcal{N}$ is a set of agents corresponding to the two independent agents controlling each robot arm in our setup. The transition dynamics remain the same as in the central setup, whereas the turn indicator of the state set is removed. Each agent controlling a robot arm then has its own action set consisting of the same actions as in the central setup, namely to place a block, but with the additional possibility to keep holding the current block. This modification allows both robots to act simultaneously as in Figure 4

Furthermore, each agent controlling one of the two robot arms has its own reward function which we assume to be identical for each agent since the two agents have the same goal, namely building a stable spanning structure. We define the reward function analogously as in the central setup. Solving a Markov game, where the rewards of the agents are identical corresponds to finding an optimal policy for each agent which maximizes the reward over time. Assuming the reward functions are specified such that they are aligned with the construction task there exists a policy pair that when implemented by the two agents, results in the construction of a stable spanning structure. More formally, a policy $\pi_i : \mathcal{S} \to \Delta(\mathcal{A}_i)$ of the agent controlling robot arm $i \in \{1, 2\}$ maps states to a distribution over its action set. Hereafter, we refer to the setting where two independent agents control each robot arm as the decentralized setup.

## 3.3 Learning agent

Regardless of whether the construction of a spanning structure is modeled as an MDP or a Markov game, the agent(s) aim to learn a policy that results in the robot arms building a stable spanning structure. A training phase consists of a number of construction phases, each one denoted by an episode. During an episode, following an initial policy, the robot arms repeatedly place and hold




(a) Initial state (b) First actions

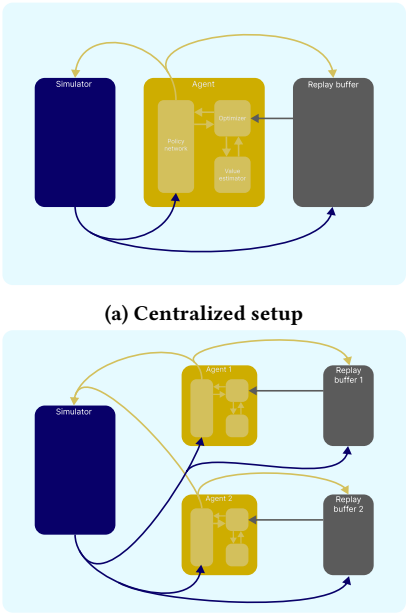
(c) Next state

(d) Second actions (e) Final state

**Figure 4: Example of a successful episode using the decentralized setup. The initial state (Figure 4a) consists of the two grounds that need to be connected. In Figure 4b, each agent instructs their robot arm to place a block. As these actions create a stable structure, the two blocks are placed in the next state (Figure 4c). Each agent instructs their robot arm to place a second block (Figure 4d) and a stable spanning structure is achieved (Figure 4e). All states and actions are stored in the replay buffers of each agent used for further training. One episode is completed.**

blocks in the grid starting from the base triangle. The episode terminates when either a stable spanning structure is built, the structure collapses, or a certain number of blocks is placed. Based on the cumulative reward obtained during the episode from the constructed structure and the need to explore different policies, the policy of the agent(s) is updated. Now following the updated policy, a new episode begins and the construction process above is repeated for the training phrase.

In this work, we consider the actor critic framework to learn a policy. The actor critic framework consists of a policy model, the so-called actor, and a value estimator, the so-called critic. The policy model is used to parameterize $\pi$ in the MDP, and the value estimator, as the name suggests, is used to estimate the value of the state $V(s, \pi)$. At each time step of the simulation, the agent(s) use its actor to select an action, and use a batch of past transitions to optimize the policy model and the value estimator. Each transition is composed of a state, the action taken by the agent, the next state, and the reward obtained for this state-action-state tuple. These transitions are stored in an agent-specific replay buffer. Figure 5 schematically shows the difference between the centralized and the decentralized setup. Two different algorithms are used to train the agent(s), namely the soft actor critic (SAC) and the advantage actor critic (A2C) algorithm. The main difference between the two algorithms is the method with which the actions are explored. The SAC algorithms implement a so-called *softmax* normalization step on the output of the actor. This encourages the agent to explore the policy space in a way that prioritizes promising actions. A2C uses a so-called $\epsilon$-*greedy* policy, where the most promising action is chosen with probability $1 - \epsilon$ and any other action is chosen with equal probability. Thus, compared to SAC, A2C does not select actions with a probability that is proportional to the likelihood of that action being a good action.




(a) Centralized setup

(b) Decentralized setup

**Figure 5: Training architecture of the centralized and decentralized setup.**

*3.3.1 Neural network architecture.* In the SAC and A2C algorithm the policy and value estimator are parameterized by differentiable functions. In our work, we discuss how two different structures of neural networks can be used to represent those functions, namely convolutional neural network (CNN) and graph neural network (GNN). These two structures are chosen for comparison because they are a common choice in RL and have been shown to perform well. In the CNN architecture, each network takes the isometric grid as input, with the up and down triangles considered as separate channels. After several convolutional layers, the encoded result is passed through a stack of fully-connected layers. Since the output layer does not use any factorization of actions, this leads to a large number of possible actions at the output layer of the network.

To mitigate this problem, the probability of infeasible actions, e.g. placing a block that collides with an existing block, is set to zero.

In the GNN architecture, the robot arms and the current structure are described as a graph. This representation is then passed through a residual gated graph convolutional neural network [Bresson and Laurent 2017] and, as in [Bapst et al. 2019], an action is chosen according to the attribute of the edge of the processed graph. GNN architecture was shown to achieve better results in the work of [Bapst et al. 2019], and allows for an elegant and compact representation. Moreover, as they are learning the relations and interactions between neighbouring blocks, [Bapst et al. 2019] have also shown that this architecture could better generalize the learned model. However, our setup differs from their work in the following two aspects: The speed of our simulator and the complexity of the block shapes used. Regarding speed, due to the simulator's low level of detail, the majority of time spent on computing a simulation step is dedicated to updating the policy. The CNN architecture is approximately five times faster than the GNN architecture, even though it has ten times more parameters. Regarding the complexity of shapes, the hourglass-shaped blocks are harder to learn with the GNN than with the CNN architecture. Therefore, in the following section, the results of our experiments are summarized only for the CNN architecture.

## 4 EXPERIMENTS AND RESULTS

We performed two experiments on the task of scaffold-free construction of a spanning structure. The first experiment evaluated how well this task can be achieved in the decentralized setup compared to the centralized setup. The second experiment analyzed what level of task complexity in terms of the width of the spanning structure can be learned in the centralized setup.

### 4.1 Experiment 1

In the first experiment, we compared both the central and the decentralized setup on the task of building a spanning structure in the simulation environment defined in Section 3.1. In both setups, we used the SAC algorithm for training the central and the two independent agents, respectively. In the decentralized setup, we were particularly interested in whether the two robot arms learn to avoid collisions even though they were independently controlled and whether they learned to coordinate their actions. By collision we mean that the robot arms do not simultaneously place blocks on the same triangle in the grid and by coordination we mean that while one robot arm is holding the last placed block to maintain the structure, the other robot arm places a new block in a way that allows the first one to move. Such coordination is essential for the construction of scaffold-free spanning structures.

In our experiment, the two ground bases that the robot arms aimed to connect were randomly set with a distance of one to seven triangles at the beginning of each episode. The central and the two independent agents were each trained on 40'000 episodes to learn a policy. During the training the network parameters of the SAC algorithm were optimized 136'000 approximately times. Each time the parameters were optimized we denote as an optimization step. An illustrative example of an episode in which a successful spanning structure is built is depicted in Figure 4.

*4.1.1 Results.* We divide the task of connecting two ground bases into two categories based on the distance between the ground bases, where a task is classified as easy and hard, depending on whether the distance between the bases is one to two or three to five, respectively. Hard tasks require that the robot arms coordinate, namely that they alternate between one robot arm placing a new block against the block held by the other robot arm, and then the roles are swapped. Easy tasks do not require such cooperation between the robot arms. The results are shown on Figure 6

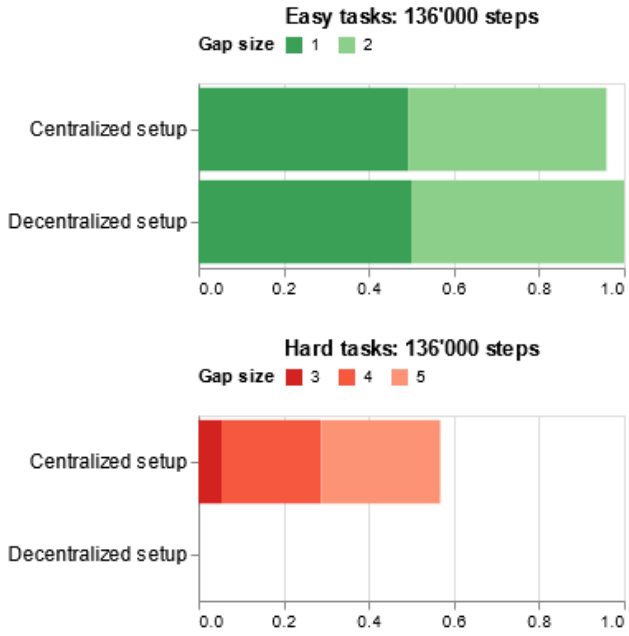



**Figure 6: Success rate of the centralized and decentralized setup for easy and hard tasks.**

Figure 7 illustrates episodes in which a spanning structure was successfully built both for the central and decentralized setup, whereas Figure 8 and 9 show the last episode of training, where, both in the centralized and decentralized setup, the robot arms did not manage to complete a spanning structure.

*4.1.2 Discussion.* In the centralized setup, the central agent learns a policy that results in the robot arms successfully building a spanning structure, both for easy and hard tasks. This is also the case for easy tasks in the decentralized setup. For hard tasks, however, the two robot arms do not learn to build a stable spanning structure. This can be seen from the success rate shown in Figure 6. The success rate $s(t)$ is an exponential moving average over the number of successful episodes in relation to the total number of episodes, where episodes further in the past have an exponentially decaying impact on the success rate. The detailed formula used to describe the success rate is

$$s(t) = \begin{cases} 0.99s(t-1) + 0.01 & \text{in case of success} \\ 0.99s(t-1) & \text{otherwise} \end{cases}$$

.

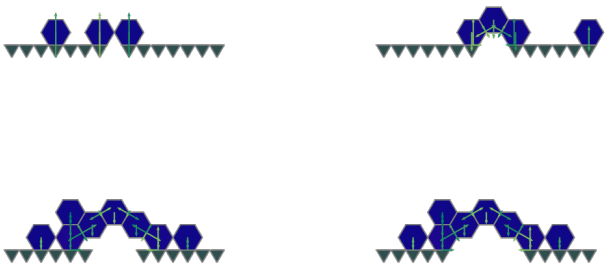


**Figure 7: Successful construction of spanning structures achieved in the central setup. Similar spanning structures are also achieved in the decentralized setup.**

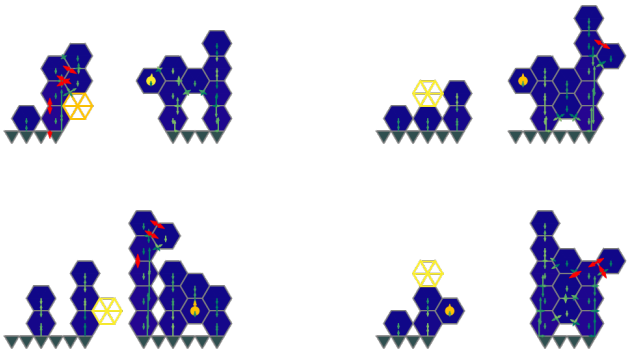


**Figure 8: Failed attempts to build a stable spanning structure shown for the last episode of training in the centralized setup.**

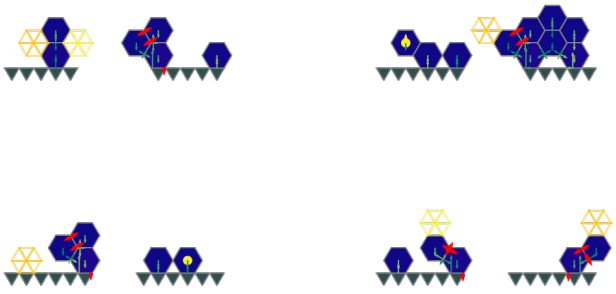


**Figure 9: Failed attempts to build a stable spanning structure shown for the last episode of training in the decentralized setup.**

Such a success measure gives insight into the probability of the current policy being successful, rather than taking into account less optimized previous ones. Note that for hard tasks in the decentralized setup, the success rate is zero. The reason why learning in the decentralized setup is successful for easy tasks but not for hard tasks is that easy tasks do not require any coordination between the agents. In our experiments, for hard tasks in the decentralized setup we notice that the agents are unable to coordinate sufficiently. However, we note that the independent agents do learn a policy that results in the robot arms avoiding collision during the construction phase. Namely, the agents do not simultaneously place blocks on the same triangle in the grid, and each specializes in placing blocks where the other agent does not do so.

In conclusion, our experiment shows that although at least two robot arms are necessary to build a scaffold-free spanning structure, having a central policy model that control both robot arms outperforms the case where two agents independently learn a policy to each control one simulated robot arm.

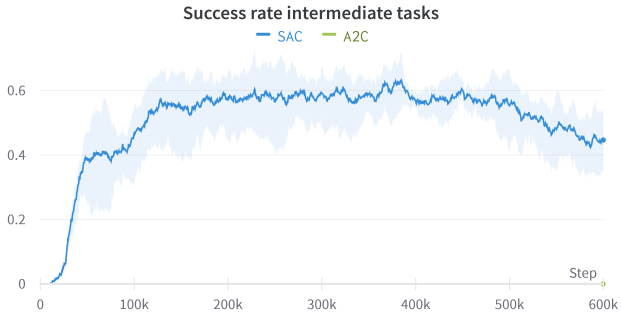
## 4.2 Experiment 2

The purpose of this experiment is to illustrate what level of task complexity can be learned in the central setup when the two robot arms are tasked with building a large spanning structure. We consider only the centralized setup due to its success shown in experiment 1 in Section 4.1. To be able to build larger spanning structures, we modify the simulation environment specified in Section 3.1. Firstly, in addition to hexagonal blocks we also consider hourglass-shape blocks (see Figure 2) for the construction of a spanning structure. This increases the number and variety of feasible spanning structures. Secondly, we require that new blocks must be placed adjacent to the last placed block rather than against any already placed block. This drastically reduces the number of possible actions that can be explored by the two robot arms at each stage of the construction process. We are interested in whether under these modifications the two robot arms can learn to build wider spanning structures. Concretely, we compare the performance of the SAC algorithm which encourages exploration to the A2C algorithm which serves as a baseline method in achieving this task. Note that the training process of 600'000 optimization steps takes around one and a half day using either algorithm, by using a cluster of 36 CPUs and a Nvidia Tesla V100-SXM2-32GB GPU.

*4.2.1 Results.* We divide the task of connecting two ground bases into four categories based on the distance between the ground bases: easy, intermediate, hard and extreme. In the easy, intermediate, hard and extreme categories, the two ground bases are one to four, five to seven, eight to ten, and eleven to nineteens triangles apart, respectively. In Figure 10, we report the mean, minimum and maximum success rate of the central agent in constructing a spanning structure for each category. Using the A2C algorithm the central agent requires approximately 400'000 optimization steps before being able to successfully complete a spanning structure in the easy category (see Figure 10a). In the three other more challenging categories a central agent trained with A2C does not learn to build a stable spanning structure even after 600'000 optimization steps. On the other hand, when the central agent is trained with the SAC algorithm, the central agent learns to complete a spanning

structure in all four categories. As to be expected, the number of optimization steps needed for the central agent to learn to build a spanning structure increases as the task complexity increases. This can be seen in Figure 10a-10d, where the success rate is plotted as a function of the number of training steps for easy, intermediate, hard, and extreme tasks.


**(a) Widths of size 1, 2, 3 and 4 triangles.**




**(b) Widths of size 5, 6 and 7 triangles.**


**(c) Widths of size 8, 9 and 10 triangles.**


**(d) Widths of size 11 to 19 triangles.**

**Figure 10: Success rate for easy, intermediate, hard and extreme tasks for the SAC and A2C algorithm. The solid line represents the mean success rate and the shaded area shows the spread from its minimum to its maximum.**

*4.2.2 Discussion.* As seen in the results above, the A2C algorithm only succeeded in constructing a spanning structure on easy tasks. The central agent learned to connect the two ground bases by building a straight line, shown in Figure 11a. However, following this policy in harder tasks resulted in an unstable structure and the structure collapsed. The drawback of the A2C algorithm is that it uses an $\epsilon$-greedy policy which does not explore the policy space sufficiently and can get stuck in local optima. For example, in 90 percent of the episodes the central agent trained with A2C built the exact same structure.

The SAC algorithm, on the other hand, succeeded in constructing a spanning structure on tasks of all difficulty levels. This could be seen from the success rates shown in Figure 10. Since the SAC algorithm has an exploration budget (fixed level of entropy) the central agent learned a nearly deterministic policy in harder tasks, where in certain states only a single action would lead to the successful completion of a spanning structure. To compensate for such nearly deterministic policies in most of the states of the hard tasks and to meet the exploration budget, in easier tasks the central agents learned a more random policy since in many states several actions could lead to the successful completion of a spanning structure.

After training, a greedy policy was obtained for each task by selecting the action with the highest probability at each state in the episode given the policy learned by SAC. Successful implementations of such a greedy policy are demonstrated in Figure 13. Even when the central agent was forced to take a sub-optimal action with respect to the greedy policy at the initial state of the episode, the robot arms were still able to build a stable spanning structures. This is shown in Figure 14.

In conclusion, using easier tasks to help learn to successfully complete harder tasks is a major advantage in training the central agent with the SAC algorithm. Furthermore, the exploration budget of the SAC algorithm ensures the construction of more diverse structures which in turn results in more robust policies. Thus, the SAC algorithm seems suitable to bridge the gap between simulations and reality. If an action is infeasible in real life for reasons such as the robot kinematics or the friction dynamics, the agent could adapt and change its plan mid-way.

**(a) Success for gaps of width up to 4 triangle wide**

**(b) Failure for larger gaps: The friction force with the ground is too small to sustain the structure.**

**Figure 11: Structures produced by the A2C algorithm after 450'000 optimization steps.**

**(a) Example, where the SAC algorithm produced the same structure as the A2C algorithm.**

**(b) Example, where the SAC algorithm produced a different structure than the A2C algorithm.**

**Figure 12: Different structures produced by the SAC algorithm after 20'000 optimization steps.**

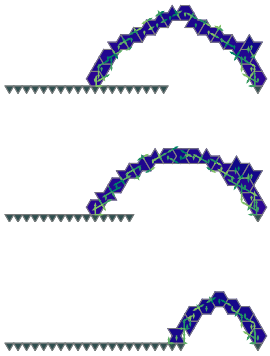


**Figure 13: Structures produced when using a greedy policy on a central agent trained with the SAC algorithm.**

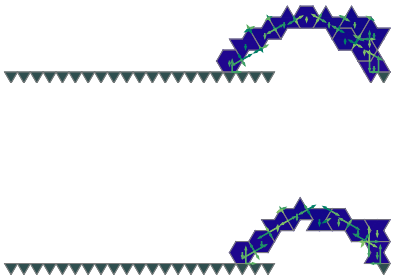


**Figure 14: Structures produced when using a greedy policy on a central agent trained with the SAC algorithm, when the rightmost block is forced to be a hourglass with different orientation.**

## 5 VALIDATION

We validated our learned models with physical robotic assembly tests. The robots used for the fabrication are two ABB GoFa CRB 15000 robot arms with suction grippers (as shown in Fig. 15). We 3D printed two types of blocks and attached sandpaper to the blocks to ensure that the friction between blocks matched the one used during the simulation. The bricks are placed at the pick station so the robot can pick them from the table and place them at the desired locations.

We planned the robotic path with Rapidly-exploring Random Tree (RRT) [LaValle and Kuffner 2001] algorithm, which accounts for both robotic feasibility and collision avoidance, including considerations for self-collisions, collisions between robots, collisions with the environment and existing structures. The RRT algorithm is implemented in compas_fab [Rust et al. 2018] with Open Motion Planning Library [Şucan et al. 2012] and MoveIt![Coleman et al. 2014]. Note that in our RL training simulation environment, we focused solely on collisions between the blocks and the robot arm's

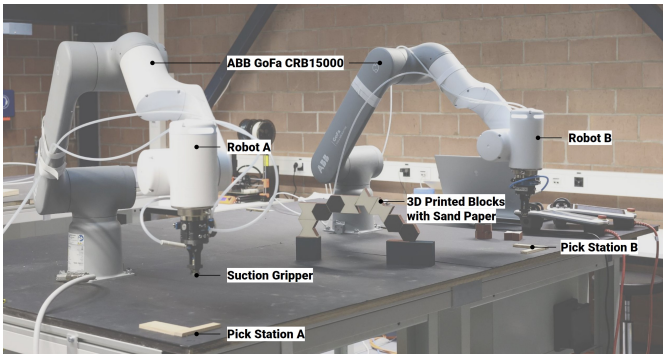



**Figure 15: Robotic fabrication setup**

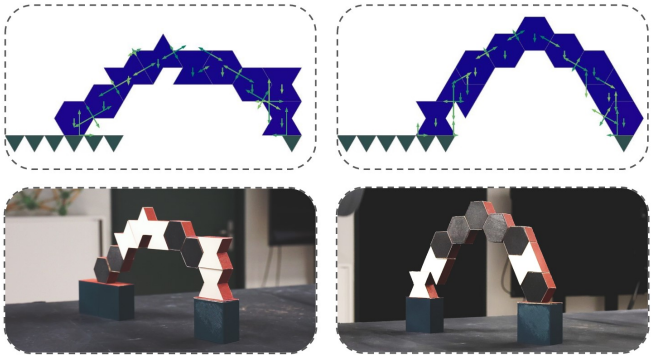


**Figure 16: Picked structures for fabrication**

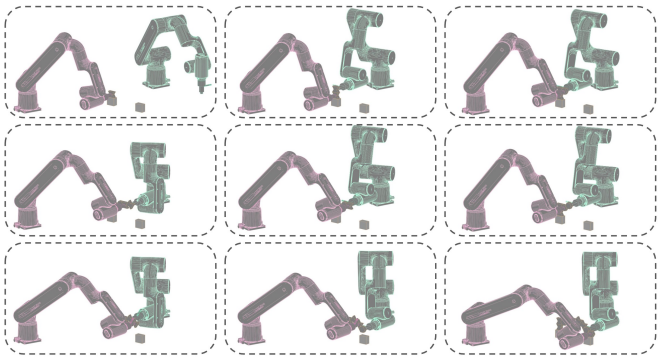


**Figure 17: Simulation of robotic fabrication process**

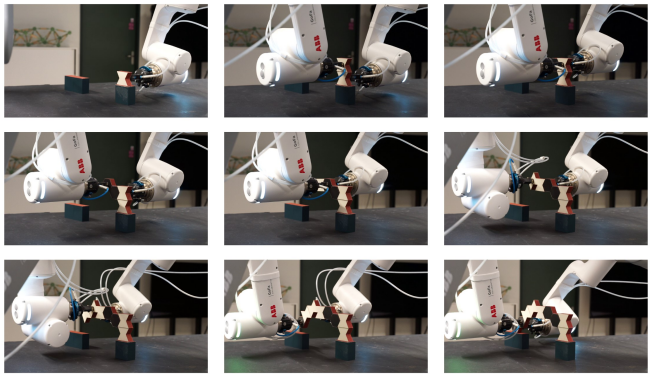


**Figure 18: Robotic fabrication of the structure**

functionality (holding or placing). We did not incorporate the complete pick-and-place path or consider potential collisions along that path. In this way, we could save the simulation time and reduce the complexity of the RL training. That is the reason we need to conduct additional path planning.

As shown in Figure 16, we tested two successful spanning structures generated from the SAC-trained model in Experiment 2 (the last structure in Figure 13 and Figure 14). Both structures are successfully built in the physical world. This validates the feasibility of our algorithm.

Here we pick the first structure to demonstrate the simulation (Figure 17) and fabrication process (Figure 18). The entire process of fabrication and simulation can be viewed in the video we submitted together with the paper.

## 6 CONTRIBUTION, LIMITATION AND FUTURE WORK

Our paper showcases the successful implementation of an RL algorithm for designing spanning structures that can be constructed by two robotic arm without scaffolding. Through experiments, we found that the centralized setup using the soft actor critic algorithm proved to be an effective approach. Importantly, we validated the practicality and effectiveness of our algorithm by successfully constructing structures learned by the RL algorithm in the real world using two robotic arms.

Our contributions can be summarized in the following points:

- We demonstrated that by working with design goals instead of specific design, the RL algorithm can break free from the current linear, top-down approach (design, geometry discretization, construction sequencing, robotic path planning, and final structure fabrication) and bring construction considerations into the design creation. Through its sequential decision-making process, the algorithm can generate spanning structures that can be constructed without scaffolding. This saves considerable time associated with design-related processes and minimizes the risk of rework. By adopting this approach, we aim to foster a novel, efficient, cost-effective robotic fabrication process.
- We showed that a decentralized MARL framework was unsuitable for the task with our current set-up, and single agent RL vastly outperforms it.

The code for our environment, algorithm, and trained model is available on GitHub[3], making it accessible to researchers and practitioners interested in exploring and utilizing our approach for their own projects.

Our results so far have some limitations:

- The reason why the MARL framework could not compete with the single agent one is not yet fully formalized. It is important to advance this understanding since the MARL framework could have potential specifically if more robotic

[3]https://github.com/Fllask/RLSFCSS

arms are needed during construction. A rigorous analysis of the convergence rate and exploring alternative algorithms for MARL can provide more valuable insights into this problem.

- The shapes of our blocks, hexagonal and hourglass-like blocks, are very specific, while the construction industry mostly works with more common and uniform shapes. In future work, we aim to enhance our algorithm to accommodate more general geometries commonly encountered in construction applications. By expanding the capabilities of our algorithm, we can address a broader range of construction scenarios and contribute to the industry's needs beyond the specific shapes we have focused on.
- Our current training simulator is fast but has limitations as it operates within a discrete, two-dimensional design space. Enhancing it to support continuous design would enable variations in block shapes and sizes, allowing agents to be trained with specific tolerances. This upgrade is crucial for future scalability and applying the algorithm in real-world scenarios. Additionally, expanding the simulator to accommodate 3D structures is a potential avenue, enabling agents to construct not just linear spans but also domes and other architectural forms. We intend to upgrade the simulator in these two ways.
- Although our algorithm manages to create interesting spanning structures and their construction sequence, it does not consider robotic kinematics and path planning. As a result, additional planning is required to ensure the robotic fabrication process can effectively execute the generated structure. While it is possible to include those factors in the stage of training, that also significantly increases the training time. An even further step would be to allow the robots to move parts of the structure composed of several blocks after they are placed. This would allow them to create some wiggle room and would be highly beneficial when finishing the structure. We would like to explore this problem in our future work.
- Our generated spanning structures are only evaluated for their stability under their self-weight. The spanning structures in the architectural context also need to sustain external loads, such as live loads (people walking on the bridge), wind loads, and earthquake loads. One of our future goals is to address those structural design requirements with the RL algorithm.
- During fabrication, many factors can introduce inaccuracies, such as robot calibration, table flatness, and precise block picking. Simulating these factors accurately is challenging. To ensure successful structure construction in the physical world, two approaches can be taken. The first approach is to enhance structure robustness by designing it to accommodate potential inaccuracies. This involves incorporating resilience into the structure's design, enabling it to tolerate and adapt to variations during fabrication. The second approach involves augmenting the fabrication process with increased intelligence. This can be achieved by integrating force sensors and cameras, which provide real-time feedback and enable more precise and informed fabrication. We would like to investigate these approaches to mitigate the impact of inaccuracies during fabrication.
- During the training, a structure is considered as a success as soon as it is stable. However, we did not investigate what would make a structure optimal. An optimal structure would not only use the least amount of blocks possible but also be stable and robust to unexpected forces during all steps of the construction. The balance between these two factors makes it hard to formally discuss this question, as it mostly relies on human expertise and regulations. A final next step would be to determine what an optimal structure is and to use this definition to train our algorithm in a more directed fashion.

While our work has certain limitations, it underscores the potential of using RL algorithms in architectural construction. By doing so, we pave the way for innovative designs, pushing the frontiers of architectural creativity and design methodologies. Therefore, pursuing the aforementioned future directions is both relevant and promising.

## REFERENCES


Victor Bapst, Alvaro Sanchez-Gonzalez, Carl Doersch, Kimberly L. Stachenfeld, Pushmeet Kohli, Peter W. Battaglia, and Jessica B. Hamrick. 2019. Structured agents for physical construction. In *ICML*.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. 2018. Relational inductive biases, deep learning, and graph networks. https://doi.org/10.48550/ARXIV.1806.01261

Boris Belousov, Bastian Wibranek, Jan Schneider, Tim Schneider, Georgia Chalvatzaki, Jan Peters, and Oliver Tessmann. 2022. Robotic architectural assembly with tactile skills: Simulation and optimization. *Automation in Construction* 133 (2022), 104006. https://doi.org/10.1016/j.autcon.2021.104006

Xavier Bresson and Thomas Laurent. 2017. Residual Gated Graph ConvNets. https://doi.org/10.48550/ARXIV.1711.07553

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. 2021. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. https://doi.org/10.48550/ARXIV.2104.13478

Edvard P.G. Bruun, Sigrid Adriaenssens, and Stefana Parascho. 2022. Structural rigidity theory applied to the scaffold-free (dis)assembly of space frames using cooperative robotics. *Automation in Construction* 141 (2022), 104405. https://doi.org/10.1016/j.autcon.2022.104405

Petros Christodoulou. 2019. Soft Actor-Critic for Discrete Action Settings. https://doi.org/10.48550/ARXIV.1910.07207

David Coleman, Ioan Alexandru Sucan, Sachin Chitta, and Nikolaus Correll. 2014. Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study. *CoRR* abs/1404.3785 (2014). http://arxiv.org/abs/1404.3785

Ursula Frick, Tom Mele, and Philippe Block. 2015. Decomposing Three-Dimensional Shapes into Self-supporting, Discrete-Element Assemblies. https://doi.org/10.1007/978-3-319-24208-8_16

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018a. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. https://doi.org/10.48550/ARXIV.1801.01290

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. 2018b. Soft Actor-Critic Algorithms and Applications. https://doi.org/10.48550/ARXIV.1812.05905

Yijiang Huang, Caelan Reed Garrett, Ian Ting, Stefana Parascho, and Caitlin Mueller. 2021. Robotic additive construction of bar structures: Unified sequence and motion planning. *CoRR* abs/2105.11438 (2021). arXiv:2105.11438 https://arxiv.org/abs/2105.11438

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and

Demis Hassabis. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature* 596, 7873 (July 2021), 583–589. https://doi.org/10.1038/s41586-021-03819-2

Steven M LaValle and Jr. James J. Kuffner. 2001. Randomized Kinodynamic Planning. *The International Journal of Robotics Research* 20 (2001), 378–400. Issue 5. https://doi.org/10.1177/02783640122067453

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Ré mi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. 2022. Competition-level code generation with AlphaCode. *Science* 378, 6624 (dec 2022), 1092–1097. https://doi.org/10.1126/science.abq1158

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2020. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. arXiv:1706.02275 [cs.LG]

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. (2016). https://doi.org/10.48550/ARXIV.1602.01783

Stefana Parascho, Isla Xi Han, Samantha Walker, Alessandro Beghini, Edvard P. G. Bruun, and Sigrid Adriaenssens. 2020. Robotic vault: a cooperative robotic assembly method for brick vault construction. *Construction Robotics* 4, 3-4 (Nov. 2020), 117–126. https://doi.org/10.1007/s41693-020-00041-w

Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st ed.). John Wiley and Sons, Inc., USA.

Matthias Rippmann. 2016. Funicular Shell Design: Geometric Approaches to Form Finding and Fabrication of Discrete Funicular Structures. https://doi.org/10.3929/ethz-a-010656780

Romana Rust, Gonzalo Casas, Stefana Parascho, David Jenny, Kathrin Dörfler, Matthias Helmreich, Augusto Gandia, Zhao Ma, Ines Ariza, Matteo Pacher, Beverly Lytle, Yijiang Huang, and Chen Kasirer. 2018. COMPAS-FAB: Robotic fabrication package for the COMPAS Framework. https://github.com/compas-dev/compas_fab/

L. S. Shapley. 1953. Stochastic Games*. *Proceedings of the National Academy of Sciences* 39, 10 (1953), 1095–1100. https://doi.org/10.1073/pnas.39.10.1095 arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.39.10.1095

Jingwen Wang, Wenjun Liu, Gene Ting-Chun Kao, Ioanna Mitropoulou, Francesco Ranaudo, Philippe Block, and Benjamin Dillenburger. 2023. Multi-robotic Assembly of Discrete Shell Structures. *Advances in Architectural Geometry*. https://doi.org/10.1515/9783111162683-020

Bastian Wibranek, Yuxi Liu, Niklas Funk, Boris Belousov, Jan Peters, and Oliver Tessmann. 2021. Reinforcement Learning for Sequential Assembly of SL-Blocks.

Kaicong Wu and Axel Kilian. 2020. Designing Compression-Only Arch Structures Using Robotic Equilibrium Assembly. *Impact: Design With All Senses* (2020), 608–622. https://doi.org/10.1007/978-3-030-29829-6_47

Ioan A Şucan, Mark Moll, and Lydia E Kavraki. 2012. The Open Motion Planning Library. *IEEE Robotics and Automation Magazine* 19 (12 2012), 72–82. Issue 4. https://doi.org/10.1109/MRA.2012.2205651 https://ompl.kavrakilab.org.