Toward Life-Long Creative Problem Solving: Using World Models for Increased Performance in Novelty Resolution

Evana Gizzi[1], **Wo Wei Lin**[2], **Mateo Guaman Castro**[3], **Ethan Harvey**[1], **Jivko Sinapov**[1]
[1] Department of Computer Science, Tufts University, MA, USA
[2] Department of Electrical and Computer Engineering, Tufts University, MA, USA
[2] The Robotics Institute, Carnegie Mellon University, PA, USA
{Evana.Gizzi,Wo_Wei.Lin,Ethan.Harvey,Jivko.Sinapov}@tufts.edu, mguamanc@andrew.cmu.edu

Abstract

Creative problem solving (CPS) is a skill which enables innovation, often times through repeated exploration of an agent's world. In this work, we investigate methods for life-long creative problem solving (LLCPS), with the goal of increasing CPS capability over time. We develop two world models to facilitate LLCPS which use sub-symbolic action and object information to predict symbolic meta-outcomes of actions. We experiment with three CPS scenarios run sequentially and in simulation. Results suggest that LLCPS is possible through the use of a world model, which can be trained on CPS exploration trials, and used to guide future CPS exploration.

Introduction

Creative problem solving (CPS) is a skill which enables adaptation to novel situations, often through innovating on-the-fly (Gizzi et al., 2020, 2022). A key process in creative problem solving work is an agent's exploration with its environment, which typically requires many interactions in order to find a solution to encountered novelty. To date, research in CPS within the field of artificial intelligence has focused predominantly on resolving a singular novel task-at-hand. For example, when a robot needs to figure out how to push a heavy object, it may explore alternative parameterizations of a `push` actions to discovery a `strike` action. In this circumstance, the agent will explore until a solution is found. In doing so, these exploration trials are often "disposed of" in CPS resolution. These interim exploration episodes typically contain a large number of agent-environment interactions, which provide a large and fruitful data sample of experience for the agent to otherwise learn from.

In this paper, we develop a method for enabling *life-long creative problem solving (LLCPS)*, which uses CPS exploration data to train a world model to increase CPS performance over time. The world model is continuously trained on both a) past CPS exploration trials and b) any past world interactions. We train two world models (a neural network and a naive Bayes model) with a combination of sub-symbolic and symbolic data as input, and symbolic data as the output. In doing so, we are able to direct the agent in its CPS exploration to avoid those trials which are not likely to resolve the CPS task, which decreases the total amount of exploration in CPS over time. We evaluated our approach in a 3D physics-based simulation environment, over three consecutive experimental scenarios to observe how CPS performance changes over time, and compared our approach to three alternative baseline world model choices.

Related Work

Although life-long creative problem solving has not been directly explored in research, similar lines of work investigate *life-long learning*, which develops methods to enable continual learning over time such that an agent is able to utilize *both* its past experiences and new knowledge (see Parisi et al. (2019) for a review). For example, Rao et al. (2019) develop a custom model for continual life-long learning which leverages a suit of artificial intelligence methods to learn representations of tasks on the fly without labels or human intervention. Within the mobile robotics navigation domain, Kahn et al. (2021) develop a method which gathers data for off-policy training of a retroactive self-supervised predictive model, centered around environment affordance learning. *Multi-task learning (MTL)* is an area within machine learning that aims to learn a general task model using samples from multiple tasks as a way to derive shared representation (Crawshaw, 2020). In doing so, MTL aims to address the data efficiency issues that are typical in machine learning for single task learning (STL), to increase performance in learning – but not necessarily to specifically be used for novel problem solving. For example, in Kalashnikov et al. (2021), a generalized multi-task deep reinforcement learning method called "MT-Opt" is trained off-line, simultaneously across multiple robot manipulation tasks. Similarly, *meta-reinforcement learning (MRL)* aims to increase performance in general task handling by optimizing adaptation to new tasks (Yu et al., 2020). For example, in Javed and White (2019), a meta-level objective is used in MRL to minimize catastrophic interference and promote future learning via naturally sparse representation learning. Unlike MTL, MRL assumes that all training and testing (novel task) data is drawn from the same task distribution.

Theoretical Framework

Consider an agent which is able to act in its world through *symbolic* planning to as a method for accomplishing tasks.

Additionally, the agent is able to use perceived *sub-symbolic* information about its world in order to learn a *world model* to resolve novelty in task failure.

Symbolic Planning

We assume that the robot has a symbolic knowledge base $\mathcal{K}$, defined as a classical planning problem, where $\mathcal{K} = \langle \mathcal{S}_{\mathcal{K}}, \mathcal{A}_{\mathcal{K}}, \mathcal{E}_{\mathcal{K}}, \mathcal{P}_{\mathcal{K}} \rangle$, with respective components denoted $\mathcal{S}, \mathcal{A}, \mathcal{E}, \mathcal{P}$ for brevity. The set $\mathcal{S}$ indicates possible world states, reachable by taking *actions* on *entities* (either manipulable, static, or parts of the agent) in the sets $\mathcal{A}$ and $\mathcal{E}$ respectively. Specifically, $\mathcal{S} = \{s_1 \dots s_n\}$, $\mathcal{E} = \{e_1 \dots e_p\}$, and $\mathcal{A} = \{a_1(\triangledown_1) \dots a_m(\triangledown_m)\}, \triangledown_i \subseteq \mathcal{E}$, where the elements in a ordered list $\triangledown_i$ are considered to be the *arguments* of its corresponding action a_i. Note that in general, entities can be physical objects in the environment or the end effectors of the robot, but in this work we only consider physical manipulable objects in the environment. We define a set of known predicate descriptors, or *fluents*, which can be used to describe entities in the world as $\mathcal{F} = \{f_1(\triangledown) \dots f_q(\triangledown)\}$ along with their negations $\hat{\mathcal{F}} = \{\hat{f_1(\triangledown)} \dots \hat{f_q(\triangledown)}\}$, where $\triangledown \subset \mathcal{E}$. Together, the predicate descriptors and their negations comprise an encompassing set of predicates $\mathcal{P} = \mathcal{F} \bigcup \hat{\mathcal{F}}$ which is used by the agent to describe states, entities, and information about the execution of actions, as is typical in planning domains. Thus, a given state $s_i \in \mathcal{S}$ is composed of a finite set of predicates $\mathcal{F}_i \subset \mathcal{F}$ which hold true in world state s_i. Note, this does not include negation predicates in $\mathcal{F}$, although these may be deduced by the planning agent. Moreover, we assume a *planning domain definition language (PDDL)* representation of actions, where each action has a set of *preconditions* and *effects*, denoted ρ_i, $p_i \in \mathcal{P}$, indicating the predicates which must hold true *before* executing an action (preconditions), and those which are assumed to hold true *after* executing an action (effects). Note that the preconditions and effects can include those negation predicates in $\hat{\mathcal{F}}$, described earlier.

The agent is able to use the aforementioned information to act in its world, through planning, to accomplish tasks. We define a task $\mathcal{T}$ in $\mathcal{K}$ as $\mathcal{T} = (\mathcal{K}, s_0, s_g)$, where s_0 is an initial state, s_g is a goal state, and $s_0, s_g \in \mathcal{S}$ (recall a state is composed of a set of fluents which hold true). A plan $\pi = [a_1, \dots a_{|\pi|}]$ is a solution to accomplishing task $\mathcal{T}$.

Sub-symbolic-based Learning

Next, we describe the sub-symbolic information known and perceivable to the agent. For a given symbolic knowledge base $\mathcal{K}$, we assume that the robot has a corresponding sub-symbolic knowledge base Ψ, containing low-level action executors and object feature information (collectively described as the tuple ($\mathcal{K}$, Ψ)). Specifically, $\Psi = \langle \mathcal{R}, X \rangle$, where $\mathcal{R} = \{r_1 \dots r_{|\mathcal{A}_{\mathcal{K}}|}\}$ denotes a set of action controllers for the actions in $\mathcal{A}_{\mathcal{K}}$, and $X = \{x_1 \dots x_{|\mathcal{E}_{\mathcal{K}}|}\}$ denotes a set of feature mappings $x_i : e_i \mapsto \mathbb{R}^n$ for the objects in $\mathcal{E}_{\mathcal{K}}$, where n is the size of the input vector (experimentally chosen), discussed in the next paragraph. For every action in $a_i \in A_{\mathcal{K}}$, there exists a corresponding action controller $r_i \in \mathcal{R}$ which is able to execute a_i with various sub-symbolic parameterizations. Thus $|A_{\mathcal{K}}| = |R|$. Additionally, for every entity $e_i \in \mathcal{E}_{\mathcal{K}}$, there exists a feature mapping $x_i \in X$ which contains sub-symbolic information about entity properties. For every entity list $\mathcal{E}_j$, there exists a list of feature mappings $\hat{X_j}$ which contains the mappings x_i of individual entities in $e_i \in \mathcal{E}_j$.

Value Description (type)	Value Possibilities
encoded action (int)	{1,2,3,4}
rate (float)	action specific
movementMagnitude (float)	action specific
encoded orientation (int)	{1,2}
encoded shape (int)	{1,2,3}
volume (float)	[0.0,∞)
encoded color (float)	{1,2,3}
entity vector magnitude (float)	[0.0,∞)
unit vector x (float)	[0.0,1.0]
unit vector y (float)	[0.0,1.0]
unit vector z (float)	[0.0,1.0]

Table 1: World model $\mathcal{W}$ input values. Data types and value possibilities of each feature in our proof-of-concept is shown. Values which are encoded into numeric values are as follows: action (1 = `push_together`, 2 = `remove_from_container`, 3 = `place_in_container`), orientation (1 = left, 2 = top), shape (1 = sphere, 2 = box, 3 = cylinder), color (1 = red, 2 = blue, 3 = green).

A given feature space X has a cardinality n (denoted $|X|_n$) such that every feature vector mapping $x_i \in X$ is represented as a feature vector containing n distinct object features (thus, $|x_i| = n$). Therefore, for a given knowledge base Ψ, entities can be described using exactly n feature values. Furthermore, we assume that the agent is able to perceive the values of a given feature space through visual or haptic feedback. We assume that the agent starts with all features abstracted already, and thus, in our proof of concept, we do not require the agent to discover these features.

Forward Model We define a world model for our hybrid tuple ($\mathcal{K}$, Ψ) as $\mathcal{W} : (a_i, r_i, \triangledown_i, X_i) \mapsto \Omega$ where Ω defines the static output vector of the world model, which numerically encodes fluent changes which incur after the mapping (See Table 2 for our proof-of-concept world model output choices. Note that the output can be changed to suit the domain). The input to the mapping is a given action a_i with parameter settings r_i, executed over arguments $\triangledown_i$ with corresponding feature vectors X_i (See Table 1 for our proof-of-concept world model input choices. Note that the input can be changed to suit the domain). Thus, for any action, parameter settings to that action, entity arguments to that action, and corresponding feature mappings or the entity arguments, $\mathcal{W}$ is able to predict what fluent states in the world may change as a result of executing a_i on e_i with low-level settings r_i and X_i.

Problem Formulation

Given a task $\mathcal{T}$, a planner generates a plan π to accomplish a goal state s_g. The planning agent, containing an accu-

Value Description	Value Possibilities
positive visibility change	$\{0,1\}$
negative visibility change	$\{0,1\}$
positive reachability change	$\{0,1\}$
negative reachability change	$\{0,1\}$
positive touching change	$\{0,1\}$
negative touching change	$\{0,1\}$

Table 2: World model $\mathcal{W}$ output values. Our proof-of-concept output vector Ω is defined by 6 output values, each characterizing meta-level symbolic changes in the world. A 0 value indicates none of the meta-level changes (in value description) occurred, whereas a 1 indicates 1 or more instances of the meta-level change occurred.

rate representation of the world in its symbolic knowledge base $\mathcal{K}$, is able to successfully execute π, thereby achieving its goal state s_g. We refer to this case as the *original scenario*. Now, suppose that in the case of novelty, something about the world changes such that $\mathcal{K}$ is no longer sufficient, but needs to be updated with new information such that $\mathcal{K}$ becomes $\mathcal{K}'$. The agent also must learn a new set of corresponding action controllers $\mathcal{R}_{\mathcal{K}'}$ (represented as trajectories relative to the arguments of the action). We refer to this scenario as the *novel scenario*. In this novel context, the planner initially uses $\mathcal{K}$ to plan for solving $\mathcal{T}$, once again generating π. Upon executing π, a plan failure occurs for some action $a_f \in \pi$. At this point, the agent must explore its world to learn a new knowledge base $\mathcal{K}'$, providing it with an updated and accurate representation of the new world, along with its corresponding set of action controllers $\mathcal{R}_{\mathcal{K}'}$. We define the learning process $\mathcal{L}$ as the process in which an agent can learn a new knowledge base $\mathcal{K}'$ using exploration method ω, such that $\mathcal{L}(\mathcal{K}, \omega) \mapsto \mathcal{K}'$.

The exploration method ω used by the agent for CPS is a method which can result in knowledge base expansion. For example, in previous work, we demonstrate knowledge base expansion through action discovery (Gizzi et al., 2021a). In preliminary work, we demonstrate knowledge base expansion via action discovery through trajectory segmentation (Gizzi et al., 2019). In another case, we demonstrate action discovery through behavior babbling across action parameter assignments (Gizzi et al., 2021b). In Gizzi et al. (2022), we provide a comprehensive review of work in CPS which provide methods for knowledge base expansion through various exploration methods ω.

Experiments

World Model

We experimented with 2 model types, with each model formulated as multi-label binary classifiers.

Inputs and Outputs The inputs and outputs to our models are listed in Table 1 and Table 2, respectively. Before training our models, we performed basic preprocessing on our data to render the data formats shown in the tables. We one-hot encoded the categorical features in both the input (actions, shapes, color, and orientation were encoded as described in Table 1), and output (world fluent changes were encoded to indicate whether they occurred or not, as described in Table 2). We also standardized continuous features by removing the mean and scaling to unit variance in a given feature space. Lastly, we split our data into a training and testing set to prevent over-fitting.

Model 1: Neural Network The first model we tested was a feed forward neural network (NN), which is a basic artificial neural networks, where connections between nodes do not form a cycle. Our NN had 3 hidden layers, 256 neurons in each hidden layer, a binary cross entropy loss function, and a decaying learning rate for the CPR scenarios. After examining multiple model choices, we determined that a shallow and narrow neural network was not complex enough to learn the data but still achieved high binary accuracy since few actions in the data set affected the agent's world. Conversely, a deep and wide neural network was able to learn the complexity of the data.

Model 2: Naïve Bayes The next model we tested was a naïve bayes model (NB). The NB model uses Bayes Theorem and the assumption that each input variable is independent to dramatically simplify calculations. We extended a binomial naïve bayes model to support multi-label classification by fitting one classifier per label. Recommending actions to the agent in CPS when performing exploration is well suited for a binomial naïve bayes model since the agent is training on a knowledge base of independent trials and each trial produces six binary labels.

Measures We developed four measures used to prioritize exploration trial recommendations by our world models. That is, given a list of possible exploration trials (where each trial describes an action to vary with corresponding parameter settings, and low level information about the entity argument to the action – thus describing a world model input choice), the agent uses its world model to first predict multi-label binary outputs described in Table 2, and then numerically quantifies each trial based on the world model output it render. By using the *least destructive* measure, the model orders the list of recommended exploration trials based on how much a given input changes in it negative reachability output. Exploration trials which minimize these changes are prioritized. The *most changes* measure ranked inputs based on how many fluent property changes they rendered through the world model. Thus, inputs that rendered the highest net value in the sum of the values of Ω were prioritized. The *most positive changes* measure prioritized inputs which resulted in the high rank for the sum of positive reachability, positive touching, and positive visibility outputs. And lastly, the *least negative changes* measure prioritized inputs which resulted in the low rank for the sum of negative reachability, negative touching, and negative visibility outputs.

Scenarios

We ran a proof-of-concept experiment of our methodology in PyBullet, which is a 3D physics simulation environment. The world model of the agent is first trained on input/output data points (described later), sampled from randomized actions on randomized entities. After initial training, the robot

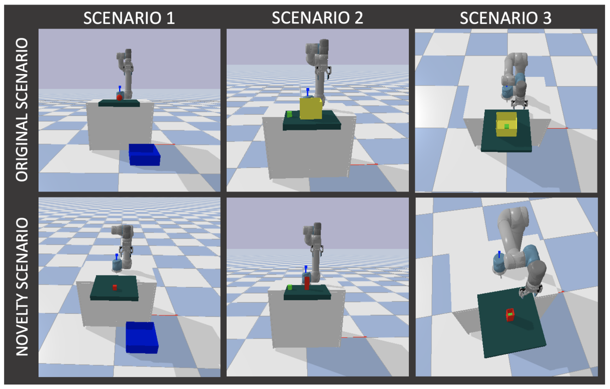


Figure 1: CPS Scenarios. In Scenario 1, the robot has a goal of pushing an object off the table, into a bucket on the ground. In Scenario 2, the robot has a goal of placing an item into a container. In Scenario 3, the robot has a goal of emptying the contents of a container, which has one object in it.

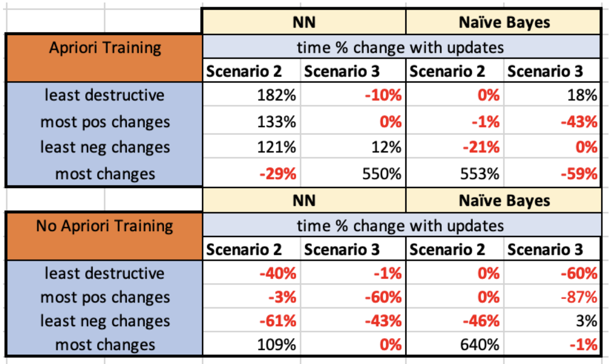
attempts to solve 3 CPS scenarios sequentially Each experimental scenario in shown in terms of its *original* and *novelty* condition, shown in Figure 1).

Results

In order to evaluated whether the use of a world model increases CPS ability (through decreasing exploration time) across longitudinal CPS trials, we ran two experiments, where the world model used in each trial was first trained on the same 200 data points of randomly generated world model interactions. In each experiment, we took the average of four trial runs to calculate average exploration time for each scenario in the described sequence, along with the total exploration time for the sequence. We performed this test for the NN model and the NB model. Additionally, we performed this test for each of the 4 measures.

In the first experiment, we allowed exploration trials during each scenario to be used to train the model over time, across scenarios. In the second experiment, we reset the training data back to the original set of 200 data points, and retrained the model before each scenario. In this way, we were able to observe whether training on CPS trials was helpful toward decreasing CPS exploration over time. Note that each scenario is characteristically different, regarding the amount of exploration needed to find a solution. For example, scenario 1 requires exploration of actions outside of the original failed plan, or defocused exploration. Therefore, comparisons were made relative to the corresponding scenarios of each experiment.

We did not find a significant difference between model updating versus not model updating in the first experiment. We believe this may be due to the fact that the data generated in the randomized trials may have not been a great representation of normal robot exploration (for example, in many trials, objects fell off of the table before exploration was able to begin). Moreover, even with accurate exploration, we believe the training apriori may have biased the agent toward "nominal problem solving," which uses different reasoning



	NN		Naïve Bayes	
Apriori Training	time % change with updates			
	Scenario 2	Scenario 3	Scenario 2	Scenario 3
least destructive	182%	-10%	0%	18%
most pos changes	133%	0%	-1%	-43%
least neg changes	121%	12%	-21%	0%
most changes	-29%	550%	553%	-59%
	NN		**Naïve Bayes**	
No Apriori Training	time % change with updates			
	Scenario 2	Scenario 3	Scenario 2	Scenario 3
least destructive	-40%	-1%	0%	-60%
most pos changes	-3%	-60%	0%	-87%
least neg changes	-61%	-43%	-46%	3%
most changes	109%	0%	640%	-1%

Figure 2: Percent change in time for scenario 2 and 3 execution. Red values show instances where model updating improve performance (by reducing exploration time against trials with no model updating). Thus, in the case where the world model was first trained a priori, there was a decrease in CPS exploration time in 50% of the measure-model combination choices. In the case where the world model was not trained a priori, there was a decrease in CPS exploration time in 81% of the measure-model combination choices.

than CPS. For this reason, we decided to test how the agent would perform if there was little aprioiri training.

We performed the same two experiments, where we instead only trained our models on 4 data points (one for each action, randomly sampled from the original 200 data points). Results are shown in Figure 2. In this case, we found that there was a reduction in CPS time between scenario 1 and 2, *and* between scenario 2 and 3, in 50% and 81% of the trial combinations for NN and NB, respectively (further described in the caption of Figure 2). **This shows that updating the NB world model using only CPS exploration trials is beneficial toward decreasing CPS exploration, as opposed to not.** When executing sequences of consecutive CPS exploration without model updating in-between scenarios, the agent was still updating its own world model within the exploration of an individual scenario. Therefore, its possible that there is still benefit in having a "miniature" world model for each scenario, not to be used in a long term sense.

Conclusion and Future Work

In this paper, we develop a method for enabling life-long creative problem solving by training a world model on creative problem solving exploration data to increase CPS exploration performance. It was shown that using a naive Bayes model is useful toward decreasing exploration time in CPS over time, when trained on CPS data alone. A limitation of our work is that it does not perform CPS over extensive/complex CPS operational runs. Future work should consider performing LLCPS over 100 seeds, or more. Similar limitations are addressed in Sun (2007). Additionally, future work should consider using alternative output vectors for capturing meta-level world changes, and different measures to rank those output values such that predictions can be more consistent across scenarios. Lastly, future work should compare alternative meta-level world models for LLCPS, including reinforcement learning-based methods.

Author Contributions

[EG] ideated/wrote the paper, and developed the framework.
[WWL] co-led development of the simulation environment.
[MGC] co-led development of the simulation environment.
[EH] developed the world models.
[JS] provided advise and edits to the paper.

Acknowledgements

The research presented in this paper was conducted with partial support from DARPA (contract W911NF-20-2-0006).

References

Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

Evana Gizzi, Mateo Guaman Castro, and Jivko Sinapov. Creative problem solving by robots using action primitive discovery. In *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 228–233. IEEE, 2019.

Evana Gizzi, Lakshmi Nair, Jivko Sinapov, and Sonia Chernova. From computational creativity to creative problem solving agents. In *International Conference on Computational Creativity (ICCC)*, pages 370–373, 2020.

Evana Gizzi, Mateo Guaman Castro, Wo-Wei Line, and Jivko Sinapov. A framework for creative problem solving through action discovery. In *2021 Robotics: Science and Systems (RSS 2021) Workshop on Declarative and Neurosymbolic Representations in Robot Learning and Control (DNR-ROB)*, 2021a.

Evana Gizzi, Amel Hassan, Wo Wei Lin, Keenan Rhea, and Jivko Sinapov. Toward creative problem solving agents: Action discovery through behavior babbling. In *2021 IEEE International Conference on Development and Learning (ICDL)*, pages 1–7. IEEE, 2021b.

Evana Gizzi, Lakshmi Nair, Sonia Chernova, and Jivko Sinapov. Creative problem solving in artificially intelligent agents: A survey and framework. *arXiv preprint arXiv:2204.10358*, 2022.

Khurram Javed and Martha White. Meta-learning representations for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Gregory Kahn, Pieter Abbeel, and Sergey Levine. Badgr: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters*, 6 (2):1312–1319, 2021.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113: 54–71, 2019.

Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Ron Sun. The importance of cognitive architectures: An analysis based on clarion. *Journal of Experimental & Theoretical Artificial Intelligence*, 19(2):159–193, 2007.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.