Appendices

A. Curiosity Metrics

A.1. State Estimation Curiosity

The objective of the state estimation curiosity module is to estimate the underlying state of the world through a set of visual observations. State configurations that are visually distinct from previously seen configurations will lead to high losses. The curiosity module and loss function are defined as follows.

$$H_\beta : Im \rightarrow \mathcal{S}$$

$$\mathcal{L}_i = \|H_\beta(Im_i) - s_i\|_2^2$$

Where Im is the space of $84 \times 84 \times 3 \cdot n_p$ matrices containing n_p images captured from various perspectives in the scene, $s \in S$ denotes the state, and i is an index into the batch. In our experiments, we used a single top-down perspective ($n_p = 1$).

A.2. Forward Dynamics Curiosity

The objective of the forward dynamics curiosity module is to estimate future states given current states and actions. Given deterministic physics, actions modify states deterministically. Therefore, it is theoretically possible to estimate a future state from an action and the current state. Rare or unpredictable object-object interaction will yield the highest loss in dynamics prediction and therefore will be sought out by the curiosity module. The curiosity module and loss function are described as follows.

$$H_\beta : S \rightarrow \mathcal{S}$$

$$f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$$

$$\mathcal{L}_i = \|H_\beta(s_i) - f(s_i, a_i)\|_2^2$$

Where f is the deterministic dynamics model which maps a state and action at t to a state at time $t + \tau$ for some fixed τ.

. Correspondence to: Aidan Curtis <curtisa@mit.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

A.3. Random Network Distillation Curiosity

Random Network Distillation (RND) is a recently proposed method for obtaining state-space coverage. This is achieved by training a model to fit some fixed random transformation of the input data. The RND curiosity module achieves low loss when it matches the transformation. This results in high curiosity module loss for unseen states.

$$H_\beta, \Phi_\beta : S \rightarrow \mathcal{S}$$

$$\mathcal{L}_i = \|H_\beta(s_i) - \Phi(s_i)\|_2^2$$

Where Φ is the fixed mapping that is initialized with random parameters. Among other mappings including random linear and non-linear mappings with uniformly and gaussian distributed weights, we found that the best performing mapping was a random permutation of the state vector.

B. Network Structures and Hyperparameters

B.1. Actor Critic Policy and Value Networks

The structure of the policy network is a multilayer perceptron model with three layers. The input size is the dimensionality of the state space and the output size is equivalent to the dimensionality of the action space. Action space and state space dimensionality are described in more detail below. The hidden layers each have 64 hidden units with $tanh$ activation functions. The the value network has an equivalent architecture but with an output size of one.

For PPO and A2C training, we used the same hyperparameters during action selection network training and baseline experimentation. For PPO we used a batch size of 128 with 1024 samples collected per update, $\gamma = 0.9$, $lr = 7e - 4$, $\varepsilon = 0.2$. We also selected value and entropy coefficients of 0.5 and 0.0 respectively and restricted the magnitude of the gradient to 0.5. For each batch, we did 4 PPO epoch updates. For A2C, we used the same learning rate and coefficients as PPO.

B.2. State Estimation Convolution Network

The state estimation network takes in visual input from perspectives in the scene and outputs an estimated state. We achieved this image to state mapping using a convolutional neural network with three convolutional layers and two fully

connected layers, all with $ReLU$ nonlinearities. Because we used a single perspective for all of our experiments, the input size is $84 \times 84 \times 3$ and the output size is the dimensionality of the state space. The convolutional layers have kernel sizes of 8, 4, 3 and strides of 4, 2, 1. The fully connected layers each have 128 hidden units.

B.3. Forward Dynamics Network

The forward dynamics network maps s_t, a_t to $s_{t+\tau}$. This is achieved using a multilayer perceptron model with 3 layers, 64 hidden units per layer and $tanh$ activation functions. The input size is the sum of the action and state dimensionality and the output size is the dimensionality of the state.

B.4. RND Network

The RND network maps states to transformed states. The size we used for the transformed state is equivalent to the size of the state. Therefore, the input and output sizes are both equivalent to the dimensionality of the state. The architecture consisted of a three layer multilayer perceptron model with $tanh$ activation functions and 64 hidden units for each hidden layer.

B.5. Universal Curiosity Module Hyperparameters

For training the curiosity networks, we used an Adam optimizer with a learning rate of 5×10^{-5}, a batch size of 128, and 1024 samples per update. We also found that performance was higher when we used an adaptive number of samples per update. The number of samples is increased until the loss is below a threshold after training. This improves CSP by giving the curiosity network time to train to baseline before adding nodes to the tree.

C. Task Specific State Space and Action Space

C.1. Stack State

This set of tasks consists of k cubes. Each cube has it's own $SE(3)$ configuration parameterized by three positional degrees of freedom and three euclidean rotational degrees of freedom. When the linking macro-action is enabled, the state also contains indicator values for each pairwise object combination.

C.2. Push-Away State

This task contains two larger cubes, one smaller cube, and one elongated, flat rectangular prism. Again, the dimensionality of the state space is $SE(3)$ for each object along with pairwise indicators when the linking macro-action is enabled.

C.3. Bookshelf State

This task contains two rods elongated rectangular prism rods and a rectangular prism book, each with a $SE(3)$ configuration space and pairwise linking. Linking is always enabled for this task because it is required to complete it.

C.4. Launch-Block State

The state space for this task contains five small cubes, one larger cube that sits at the end of the seesaw, and the angle of the seesaw itself. Each of the cubes has an $SE(3)$ configuration space and the seesaw has a bounded real configuration with a single degree of freedom.

C.5. Transfer States

When comparing between tasks, the action spaces and state spaces need matching dimensionality to ensure the networks could be substituted. We achieved this by making sure each problem instance had an equivalent number of objects, even if it was unnecessary to use all objects to complete the task. For the transfer results stated in this paper, this equated to using five objects for each task.

C.6. Action Spaces

When both linking and pick-place macro-actions are enabled, the action space needs to contain information for choosing which object to move (k total), the goal pose of the object (dof), which of the $\binom{k}{2}$ object pairs to link or unlink , and which macro-action to select (linking or pick-place). Therefore, the dimensionality of the action space is $\binom{k}{2} + (dof + 1) \cdot k + 2$. The macro-action, link, and object discrete variables are chosen by performing an argmax over the respective section of the action space. If a link macroaction is selected between two objects that are already linked, this is considered an unlink action and the constraint between the two objects is removed. If a link is chosen between two objects which aren't in contact, then the action is considered to be infeasible. For implementation simplicity, we use k^2 variables for the linking indicator rather than $\binom{k}{2}$ and ignore any redundant pairs.

D. Baselines

D.1. Statistical Analysis

In addition to the CSP variants, we compared the performance of CSP to a number of deep reinforcement learning and motion planning baselines. The implementation of these baselines is explained in detail below.

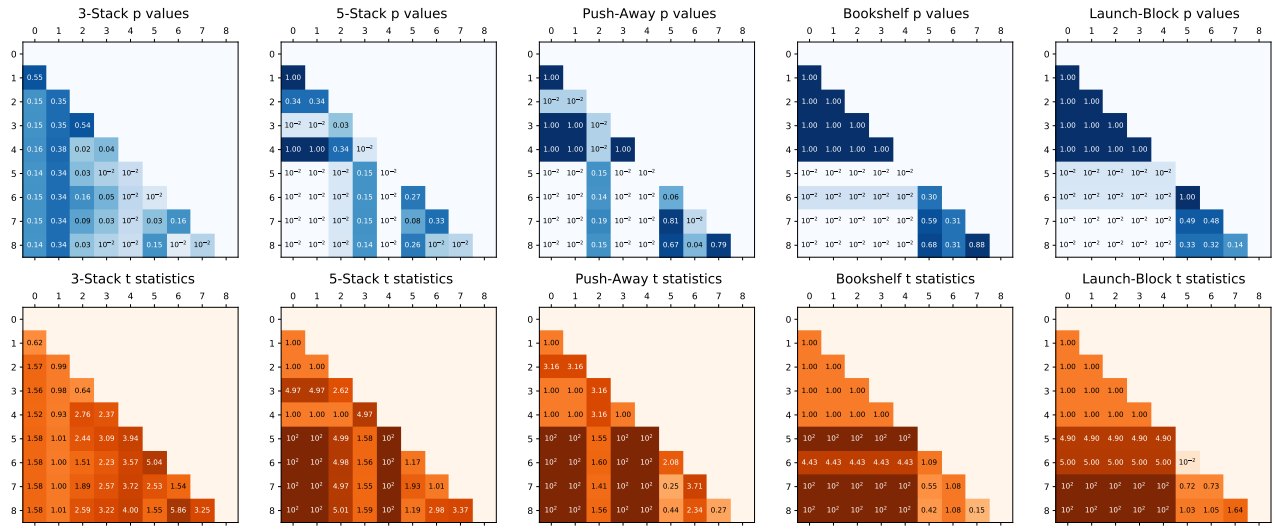

Figure 1. Results from statistical t-test between each pairwise combination of baseline algorithms on each task. The algorithms, in order, are Vanilla PPO, RND-PPO , RRT, HER, CSP-No Curiosity, CSP-RND No AC, CSP-FD, CSP-SE, CSP-RND. 10^2 is used for all values greater than 10^2 and 10^{-2} is used for all values less than 10^{-2}.

D.2. RRT

The overall objective of RRT is to explore the configuration space by sampling a feasible configuration from some distribution over the configuration space, selecting the node that is closest to that configuration in the search tree, executing an action primitive from that node, and adding the resulting node to the search tree (Lavalle, 1998). Because the problem setup is such that the effects of individual macro-actions are unknown, direct analytic control is impossible. Thus, the action is selected randomly from a uniform distribution over the space of parameterized macro-actions.

D.3. Vanilla DRL

We used vanilla implementations of PPO (Schulman et al., 2017) and A2C (Mnih et al., 2016) using the same hyperparameters described in Section B.1. A major shortcoming when using these algorithms for planning is that they are unable to revert to previously explored states and thus will often get stuck with one or multiple of the objects in the scene becoming unreachable. Therefore, we implemented a probabilistic resetting policy such that the environment resets to the initial problem state with probability 1e-4. While training, the environment was also reset to the starting state when a goal state was reached. The agent received a reward of 1 after reaching the goal state and a reward of 0 otherwise. The quantitative results reported were the number of steps taken by the trained policy in six unseen initial configurations after $5e5$ steps of training. The primary reason for the failure of vanilla DRL baselines was that the reward is far too sparse in each of the problems to generalize to unseen configurations. In many of the problems, the DRL algorithms failed to find even a single positive example for which it received reward.

In order to validate that our actor-critic implementation was functioning correctly, we tested it on the simpler 2-Stack problem in which the reward was not too sparse for policy learning. The 2-stack problem was solved by our actor critic implementation but the sparse-reward 3-stack problem could not be solved (see Fig 2).

D.4. Curious DRL

One potential method for overcoming the sparse reward problem in DRL is to "densify" the reward space by adding an intrinsic reward that incentivizes environmental exploration (Choi et al., 2019). In order to test this, we modified the vanilla PPO implementation to use the sum of extrinsic reward (R_e) and RND world model loss trained alongside the actor-critic networks (R_i) as the total reward ($R_t = \alpha R_e + (1-\alpha)R_i$) for the DRL agent. We experiment with several values of the alpha hyperparameter ($\alpha = 0.0, 0.1, 0.5, 0.9, 0.99, 1.0$), with vanilla PPO being a special case where $\alpha = 1$, and found no clear benefit over vanilla training in the quantitative results (see Fig 2).

D.5. HER

We used the DDPG implementation of hindsight experience replay (Andrychowicz et al., 2017) which modifies the experience replay buffer to be evenly split between no-reward trajectories and pseudo-reward trajectories. The pseudo-reward trajectories are generated from no-reward trajectories by giving a reward for whatever state was reached at a

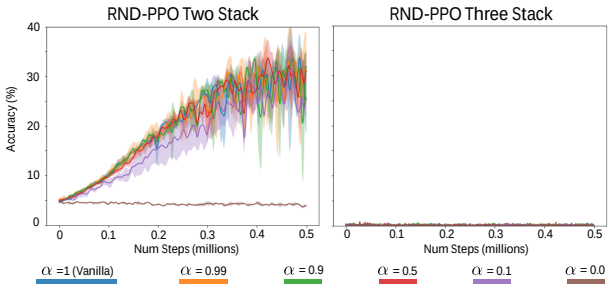

Figure 2. Reward curves during PPO training for (a) 2-Stack and (b) 3-Stack tasks using different intrinsic/extrinsic reward weighting α. These results demonstrate that our actor-critic DRL implementation is correct and functioning in environments where the reward it not sparse, but fails to learn in sparse-reward environments even when the reward includes intrinsic curiosity.

	WM Update	AC Update	Action
Vanilla PPO	N/A	271 ± 30	0.9 ± 0.1
RND-PPO	187 ± 17	251 ± 44	0.9 ± 0.1
RRT	N/A	N/A	44 ± 13
CSP-None	N/A	N/A	1.1 ± 0.4
8 **CSP-RND**	172 ± 6	243 ± 18	1.1 ± 0.2
CSP-FD	161 ± 15	224 ± 36	1.0 ± 0.2
CSP-SE	249 ± 10	219 ± 21	1.2 ± 0.2

Table 1. Time takes for batch updates and action selection in each of the baseline algorithms (In milliseconds). All testing was done on the same device with . CSP-SE is significantly slower when run on a CPU.

random point on the no-reward trajectory. Our implementation follows the DDPG architecture and hyperparameters used in the HER paper.

E. Runtime Complexity

In this section, we empirically analyze the complexity of CSP compared to other baselines using two metrics: batch update speed and action selection speed. While these are not the only time-consuming components of the planning process (e.g. environment simulation speed, geometric motion planning, graph data structure bookkeeping), they are the only algorithm-dependent components. All experiments were run on a 48 CPU machine using a single NVIDIA TITAN X GPU.

F. Code

All of the code for this project can be found here:
https://github.com/neuroailab/CuriousSamplePlanner

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. *CoRR*, abs/1707.01495, 2017. URL http://arxiv.org/abs/1707.01495.

Choi, J., Guo, Y., Moczulski, M., Oh, J., Wu, N., Norouzi, M., and Lee, H. Contingency-aware exploration in reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyxGB2AcY7.

Lavalle, S. M. Rapidly-exploring random trees: A new tool for path planning. 1998.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016. URL http://arxiv.org/abs/1602.01783.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.