Conditional Abstraction Trees for Sample-Efficient Reinforcement Learning (Supplementary Material)

Mehdi Dadvar[1] **Rashmeet Kaur Nayyar**[1] **Siddharth Srivastava**[1]

[1]Arizona State University, Tempe, Arizona, USA

A SCALABILITY STUDY

We conducted experiments (see Fig. 1) to test the scalability of CAT+RL, Q-learning, and PPO on Office World problems with increasing complexity. It shows that CAT+RL has greater scalability than the baselines. The results also indicate that (a) DRL methods do better on smaller problem instances that are less "complex" and are unable to handle increasing complexity and (b) methods designed for image-based RL do not directly scale in RL problems such as those used in this work. Thus, CAT+RL addresses the challenges with the scalability of RL to tasks whose states cannot be easily expressed as images or robot configuration states.

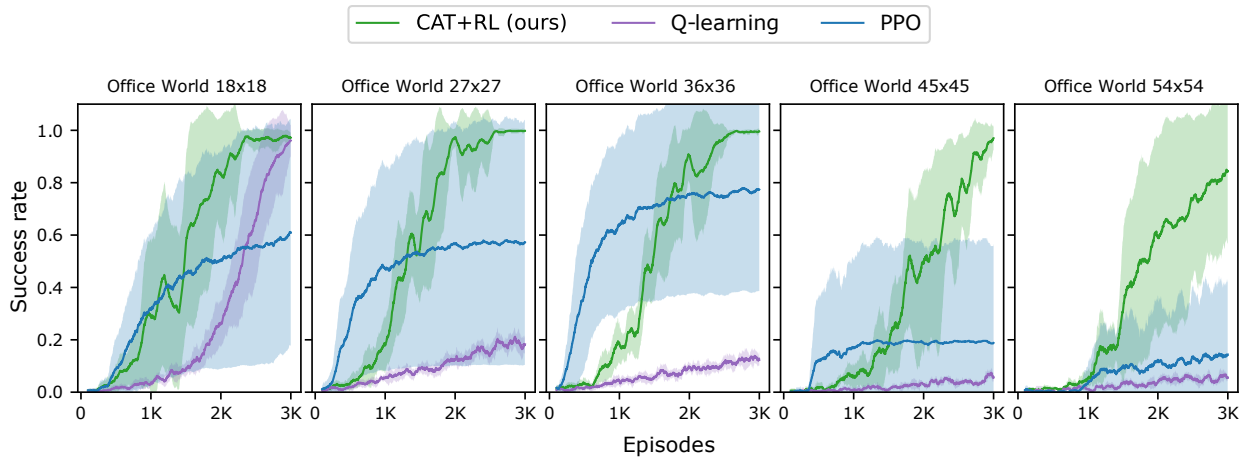

Figure 1: Scalability of CAT+RL (our method), Q-learning, and PPO on Office World problems with increasing complexity i.e. increasing ranges of state variables. The title refers to the problem size, y-axis shows average success rates and standard deviations for 10 independent runs averaged over last 100 training episodes, and x-axis shows episodes. The maximum episode lengths used for Office World problems with dimensions 18x18, 27x27, 36x36, 45x45, and 54x54 are 250, 500, 700, 1000, and 1500 respectively.

We replicated the scalability study with an exact condition compared to Fig. 1 except we altered the neural network architecture of PPO to study the effect of the neural network architecture of deep RL algorithms on their scalability, as shown in Fig. 2. To this end, we reduced the size of PPO's network architecture from 64 to 16 neurons per hidden layer, where two hidden layers were utilized in both cases. The results indicate that reducing the network size does not improve the performance of PPO and rejects the hypothesis that the original architecture used in the paper for deep RL baselines might be over parameterized or excessively large for the given test problems.

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

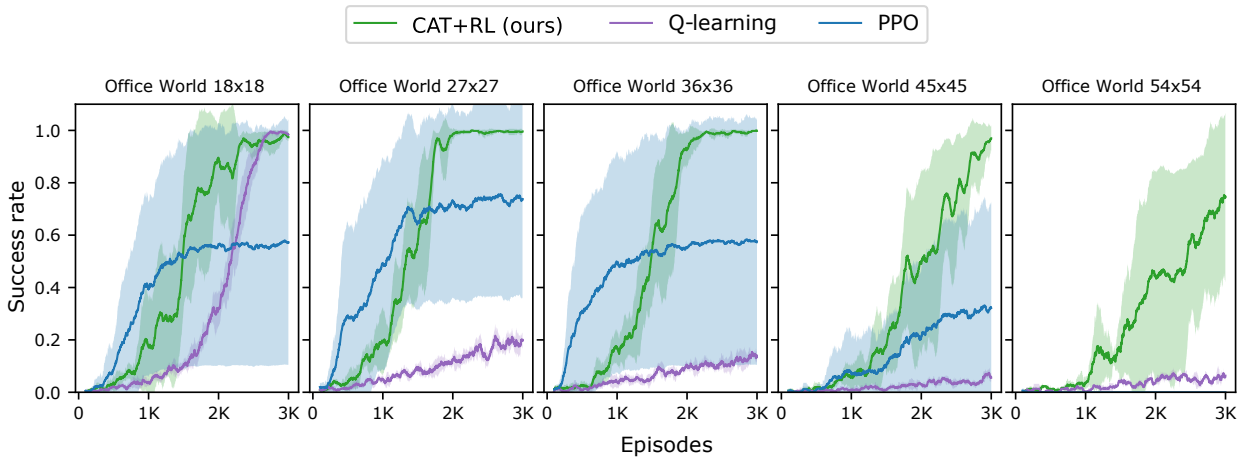

Figure 2: Scalability of CAT+RL (our method), Q-learning, and PPO with a small neural network. This is a replication of the scalability study reported in Fig. 1 except we ran PPO with a smaller neural network architecture (two hidden layers with 16 neurons per hidden layer).

B TIME COMPLEXITY ANALYSIS

The worst case of the computational complexity of the learning and evaluation phases of CAT+RL is similar to that of the underlying RL algorithm that it is used (Q-learning). The refinement phase consists of a CAT search for an unstable state with a time complexity of $O(n \log n)$ and a split operation which is linear in the number of state variables.

Tab. 1 shows the time (mean and standard deviation computed for 10 runs) taken for CAT+RL, Q-learning, and PPO for solving Office World problems with increasing problem complexity. We also compared the runtimes for CAT+RL, Q-learning, PPO, and DQN for all domains as shown in Tab. 1. CAT+RL takes significantly less times, especially compared to DRL baselines (atleast 10 times and atmost 50 times less than baselines) on three out of five domains. In the Office World domain, CAT+RL takes about 1.6 times less time than DQN. In the Water World domain, DRL baselines are faster with much lower runtimes for all the algorithms compared to other domains, and the reasons can be attributed to the low horizon used and high stochasticity in the problems due to random initialization of the agent and ball locations. All deep learning experiments were executed on two GeForce RTX 3070 GPUs with 8 GB memory running Ubuntu 18.04. All other experiments were executed on 5.0 GHz Intel i9 CPUs with 64 GB RAM running Ubuntu 18.04.

Office World problem size	Time (s) $\pm$ std dev by CAT+RL	Time (s) $\pm$ std dev by Q-learning	Time (s) $\pm$ std dev by PPO
18x18	302.75 ± 29.0	97.69 ± 4.7	2843.42 ± 959.17
27x27	391.36 ± 28.5	441.8 ± 13.85	4956.8 ± 2458.74
36x36	535.71 ± 54.6	1174.84 ± 46.23	8428.24 ± 2867.52
45x45	416.41 ± 58.94	1322.26 ± 45.87	11463.28 ± 3309.09
54x54	1010.52 ± 219.98	7750.53 ± 308.79	15293.57 ± 5815.63

Table 1: Total time taken (mean and standard deviation) by CAT+RL, Q-learning, and PPO to solve Office World problems with increasing complexity.

We found CAT+RL to be surprisingly efficient in terms of runtime. Although Q-learning was completed before our approach for small problems, CAT+RL is significantly faster than Q-learning when the problem size increases even when the time for abstraction refinement is taken into account. The reason for this performance boost is that, in practice, CAT+RL performs significantly fewer computations than it would require to solve the underlying MDP due to the abstraction that it builds on the fly. Although the abstract MDP becomes finer after each refinement phase, the state space size of this abstract MDP is still significantly smaller than the concrete MDP.

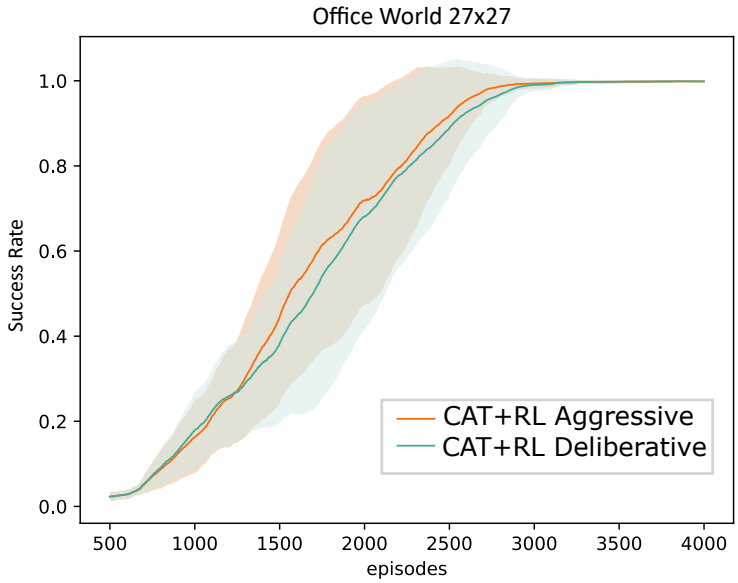

Figure 3: Comparing aggressive and deliberative variants of CAT+RL. We ran both variants 10 times and the shadows show the standard deviation of the measurements.

Problem (size)	Time (s) $\pm$ std dev by CAT+RL	Time (s) $\pm$ std dev by Q-learning	Time (s) $\pm$ std dev by PPO	Time (s) $\pm$ std dev by DQN
Wumpus (64x64)	137.86 ± 11.41	2184.56 ± 11.90	9956.56 ± 5123.24	6487.406 ± 134.55
Taxi (30x30)	744.18 ± 51.78	16835.07 ± 243.5	34642.69 ± 2720.662	8921.77 ± 1179.88
Office (36x36)	535.71 ± 54.6	1174.84 ± 46.23	8428.24 ± 2867.52	877.474 ± 291.595
Water World	804.86 ± 13.63	–	418.358 ± 30.02	271.75 ± 8.58
Mountain Car	36.84 ± 1.88	–	3851.592 ± 1560.44	2196.104 ± 347.024

Table 2: Total time taken (mean and standard deviation for 5 runs) by CAT+RL, Q-learning, PPO, and DQN to solve Wumpus World, Taxi World, Office World, Water World, and Mountain Car problems.

C HYPERPARAMETERS

We used standard architectures for A2C, PPO, DQN from StableBaselines3 [1] and Option-Critic [2]. We use the open-source code available for the state-of-the-art baseline JIRP [3].

CAT+RL's parameters are n_{check} and the threshold value t_{succ} for the refinement condition, the cap k for the maximum number of unstable states that can be refined in each refinement phase, and n_{eval} for the duration of the evaluation phase. The same parameter values were used across all our experiments except for cap k which was set proportionally to the size of the problem. In contrast, we had to conduct significant hyperparameter exploration for the baselines because the default settings led to insignificant learning.

For all of the domains, we use two layers each consisting of 64 neurons in all DRL architectures except for the Mountain Car domain where we found two layers with 128 neurons each as the best-performing architecture. Tab. 3, 4, 5, 7 show the important hyperparameters used for all the domains and methods.

[1] https://github.com/DLR-RM/stable-baselines3
[2] https://github.com/lweitkamp/option-critic-pytorch
[3] https://github.com/logic-and-learning/AdvisoRL

Hyperparameters	CAT+RL	Q-learning	Option-critic	JIRP	A2C	DQN	PPO
Threshold (t_{succ})	0.8	–	–	–	–	–	–
n_{check}	100	–	–	–	–	–	–
n_{eval}	100	–	–	–	–	–	–
Cap (k)	20	–	–	–	–	–	–
Exploration rate (ϵ)	1.0	1.0	1.0	0.4	–	1.0	–
Minimum exploration rate	0.05	0.05	0.05	0.05	–	0.05	–
Exploration decay	0.991	0.991	0.9991	0.9991	–	–	–
Exploration fraction	–	–	–	–	–	1.0	–
Learning rate (α)	0.05	0.05	0.05	1e-4	7e-4	2e-4	2e-4
Discount factor (γ)	0.95	0.95	0.95	0.95	0.95	0.95	0.95
Number of episodes	10000	10000	10000	10000	10000	10000	10000
Maximum episode length	1200	1200	1200	1200	1200	1200	1200
Options	-	-	8	-	-	-	-

Table 3: Parameters used in Wumpus World.

Hyperparameters	CAT+RL	Q-learning	Option-critic	JIRP	A2C	DQN	PPO
Threshold (t_{succ})	0.8	–	–	–	–	–	–
n_{check}	100	–	–	–	–	–	–
n_{eval}	100	–	–	–	–	–	–
Cap (k)	20	–	–	–	–	–	–
Exploration rate (ϵ)	1.0	1.0	1.0	0.4	–	1.0	–
Minimum exploration rate	0.05	0.05	0.05	0.05	–	0.05	–
Exploration decay	0.9992	0.9992	0.9992	0.9992	–	–	–
Exploration fraction	–	–	–	–	–	1.0	–
Learning rate (α)	0.05	0.05	0.05	1e-4	8e-4	1e-4	3e-4
Discount factor (γ)	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Number of episodes	3000	3000	3000	3000	3000	3000	3000
Maximum episode length	1000	1000	1000	1000	1000	1000	1000
Options	-	-	8	-	-	-	-

Table 4: Parameters used in Office World.

Hyperparameters	CAT+RL	Q-learning	Option-critic	JIRP	A2C	DQN	PPO
Threshold (t_{succ})	0.8	–	–	–	–	–	–
n_{check}	100	–	–	–	–	–	–
n_{eval}	100	–	–	–	–	–	–
Cap (k)	10	–	–	–	–	–	–
Exploration rate (ϵ)	1.0	1.0	1.0	0.4	–	1.0	–
Minimum exploration rate	0.05	0.05	0.05	0.05	–	0.05	–
Exploration decay	0.9992	0.9992	0.9992	0.9992	–	–	–
Exploration fraction	–	–	–	–	–	1.0	–
Learning rate (α)	0.05	0.05	0.05	5e-5	7e-4	1e-4	2e-4
Discount factor (γ)	0.999	0.999	0.999	0.999	0.999	0.999	0.999
Number of episodes	20000	20000	20000	20000	20000	20000	20000
Maximum episode length	1500	1500	1500	1500	1500	1500	1500
Options	-	-	8	-	-	-	-

Table 5: Parameters used in Taxi World.

Hyperparameters	CAT+RL	A2C	DQN	PPO
Threshold (t_{succ})	0.7	–	–	–
n_{check}	150	–	–	–
n_{eval}	150	–	–	–
Cap (k)	1	–	–	–
Exploration rate (ϵ)	1.0	–	1.0	–
Minimum exploration rate	0.05	–	0.05	–
Exploration decay	0.999	–	–	–
Exploration fraction	–	–	1.0	–
Learning rate (α)	0.05	7e-4	1e-4	3e-4
Discount factor (γ)	0.95	0.95	0.95	0.95
Number of episodes	5000	5000	5000	5000
Maximum episode length	100	100	100	100

Table 6: Parameters used in Water World.

Hyperparameters	CAT+RL	A2C	DQN	PPO
Threshold (t_{succ})	0.8	–	–	–
n_{check}	400	–	–	–
n_{eval}	400	–	–	–
Cap (k)	1	–	–	–
Exploration rate (ϵ)	1.0	–	1.0	–
Minimum exploration rate	0.01	–	0.01	–
Exploration decay	0.99	–	–	–
Exploration fraction	–	–	0.1	–
Learning rate (α)	0.05	1e-4	1e-4	1e-4
Discount factor (γ)	0.99	0.99	0.99	0.99
Number of episodes	2000	2000	2000	2000
Maximum episode length	200	200	200	200

Table 7: Parameters used in Mountain Car World.

D ALGORITHMIC DETAILS

In the paper, we explained how CAT+RL finds a state variable for each found unstable state. Here, we propose another approach for finding the state variable for the situations where the concrete Q-table is not available to CAT+RL. This approach aggressively blames all of the state variables for each unstable state and refines each unstable state w.r.t. all state variables. This method can be employed to avoid keeping the track of the concrete Q-table. The aggressive CAT+RL is essentially effective where the reward is frequent with high variation in an environment. We implemented the aggressive and the deliberative (the one discussed in the paper) variants of CAT+RL for blaming a state variable to do the refinement of an unstable state. We analyzed the performance of both variants of CAT+RL in Office World and demonstrated that the CAT+RL performs robustly regardless of the choice of this component, as shown in 3.

E DOMAIN DESCRIPTIONS

Office World We consider an office world scenario with dimensions 36×36 containing walls and four rooms A, B, C, and D. The task for the agent is to collect coffee and mail and deliver them to the office. The agent can execute any action from East, West, North, and South. On applying any action, the agent executes the action successfully with a probability of 0.8 and may slip to one of the two adjacent cells with a probability of 0.1 each. The agent receives a reward of 1000 on completing the task successfully and 0 otherwise.

Wumpus World We consider a Wumpus world with dimensions 64×64 containing obstacles and pits. The task for the agent is to reach the southeast corner location from the northwest corner location in the grid while avoiding pits. The four actions and the stochastic probabilities are the same as in the office world. If the agent's movement is obstructed due to an obstacle, it falls back to its location and receives a reward of -2. The agent receives -1 reward on every step and the episode ends

as soon as it enters a pit, receiving a negative reward of -1000. On reaching the correct destination location, it receives a positive reward of 500.

Taxi World We consider a taxi world scenario with dimensions 30×30 in which there are four pick-up and drop-off locations, one in each corner of the grid. The taxi agent starts at a random cell in the grid. The task of the taxi is to pick up a passenger from its pick-up location and deliver at its destination drop-off location, both selected randomly. It can execute actions: East, West, North, South, Pick-up, and Drop-off. Each move action has stochastic probabilities similar to Office world. It obtains a reward of -1 on applying a move action and -100 on illegal pick-up and drop-off actions. Upon dropping the passenger at the correct destination, it receives a positive reward of 500.

Water World We consider a continuous state space environment with dimensions 200×200 containing one green ball, one red ball, and one agent represented by a black ball. Each ball moves in one direction with constant speed and bounces back upon hitting the edges. The agent has control over its velocity via taking a move action in one of the east, west, north, and south directions. The task for the agent is to collide with the moving green ball while avoiding the red ball. The episode terminates when the agent collides with a ball. The agent receives a reward of 1000 and -1000 on colliding with the green ball and the red ball respectively.

Mountain Car Mountain Car is a continuous state discrete action environment from Open AI Gym [4]. The agent receives -1 reward on each step and 1000 reward on reaching the goal position. The maximum number of steps allowed in an episode is 200.

[4] https://www.gymlibrary.dev/environments/classic_control/mountain_car/