

1 SCALABILITY STUDY

We conducted experiments (see Fig. 1) to test the scalability of DAR+RL, Q-learning, and PPO on Office World problems with increasing complexity. It shows that DAR+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, DAR+RL addresses the challenges with scalability of RL to tasks whose states cannot be easily expressed as images or robot configuration states.

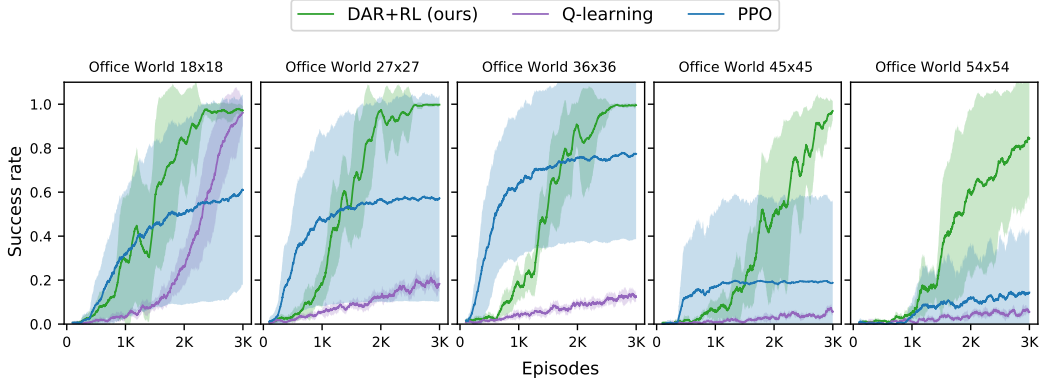


Figure 1: Scalability of DAR+RL (our method), Q-learning, and PPO on Office World problems with increasing complexity i.e. increasing ranges of state variables. The title refers to the problem size, y-axis shows average success rates and standard deviations for 10 independent runs averaged over last 100 training episodes, and x-axis shows episodes. The maximum episode lengths used for Office World problems with size 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.

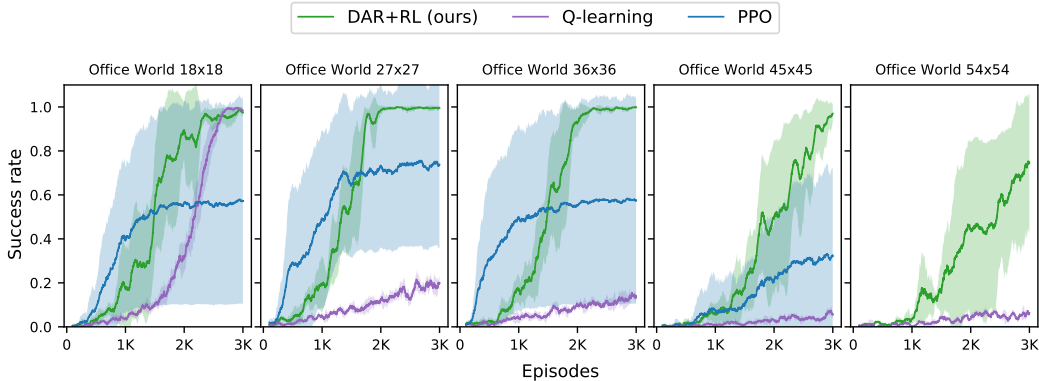


Figure 2: Scalability of DAR+RL (our method), Q-learning, and PPO with small neural network. This is a replication of the scalability study reported in Fig. 1 except we ran PPO with a smaller neural network architecture (two hidden layers with 16 neurons per hidden layer).

2 TIME COMPLEXITY ANALYSIS

The worst case of computational complexity of the learning and evaluation phases of DAR+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 DAR+RL, Q-learning, and PPO for solving Office World problems with increasing problem complexity. All 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 DAR+RL	Time (s) \pm std dev by Q-learning	Time (s) \pm std dev by PPO
18x18	302.75 \pm 29.0	97.69 \pm 4.7	2843.42 \pm 959.17
27x27	391.36 \pm 28.5	441.8 \pm 13.85	4956.8 \pm 2458.74
36x36	535.71 \pm 54.6	1174.84 \pm 46.23	8428.24 \pm 2867.52
45x45	416.41 \pm 58.94	1322.26 \pm 45.87	11463.28 \pm 3309.09
54x54	1010.52 \pm 219.98	7750.53 \pm 308.79	15293.57 \pm 5815.63

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

We found DAR+RL to be surprisingly efficient in terms of runtime. Although Q-learning completed before our approach for small problems, DAR+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, DAR+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.

3 HYPERPARAMETERS

We used standard architectures for A2C, PPO, DQN from Stable-Baselines3 (<https://github.com/DLR-RM/stable-baselines3>) and Option-Critic (<https://github.com/lweitkamp/option-critic-pytorch>). We use the open-source code available for the state-of-the-art baseline JIRP (<https://github.com/logic-and-learning/AdvisoRL>).

DAR+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 hyper-parameter exploration for the baselines because the default settings led to insignificant learning.

Tab. 2, 3, 4, 5 show the important hyperparameters used for all the domains and methods.

4 ALGORITHMIC DETAILS

4.1 FINDING UNSTABLE STATES

Starting with an initial coarse abstraction, DAR+RL proceeds the learning phase by implementing a vanilla Q-learning routine over abstract states to learn an abstract policy $\bar{\pi}(\bar{s})$. Since this coarse initial abstraction is likely to be too coarse, DAR+RL checks a refinement condition every e_{check} episodes. This refinement condition evaluates to true if the recent success rate of the learning agent is below some threshold t_{succ} . When the refinement condition evaluates to true, it implies that the current abstraction is not effective and requires further refinement. However, DAR+RL doesn’t know which abstract states are too coarse. Therefore, DAR+RL starts the evaluation phase for n_{eval} episodes to collect some samples of Q-values while the RL agent is interacting with the environment with a fixed policy. Thus, the evaluation phase is simply a Q-learning routine over abstract

Hyperparameters	DAR+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 2: Parameters used in Wumpus World.

Hyperparameters	DAR+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 3: Parameters used in Office World.

Hyperparameters	DAR+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 4: Parameters used in Taxi World.

states with a fixed policy. Since the policy is fixed in this phase, high variation in Q-values for the same state-action pair (\bar{s}, a) implies an instability in the abstract state. When the evaluation phase is terminated, all samples are stored in Γ from which the refinement phase can be executed. In the refinement phase, $UnstableState(\Gamma)$ takes the collected samples Γ and returns the top k unstable

Hyperparameters	DAR+RL	A2C	DQN	PPO
Threshold (t_{succ})	0.8	—	—	—
n_{check}	100	—	—	—
n_{eval}	100	—	—	—
Cap (k)	1	—	—	—
Exploration rate (ϵ)	1.0	—	1.0	—
Minimum exploration rate	0.5	—	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 5: Parameters used in Water World.

states. To this end, $\text{UnstableState}(\Gamma)$ calculates the normalized standard deviation of Q-values for all samples of (\bar{s}, a) . In other words, there should be multiple samples in Γ for the same abstract state and action pair. When the normalized standard deviation is calculated for all these samples, $\text{UnstableState}(\Gamma)$ clusters the abstract states (that have been sampled in Γ) into stable and unstable states using k-means clustering technique. Finally, $\text{UnstableState}(\Gamma)$ returns the top k unstable states from the unstable cluster.

4.2 FINDING UNSTABLE STATE VARIABLES

When $\text{UnstableState}(\Gamma)$ finds the top k unstable states from the samples collected throughout the evaluation phase, the unstable states can be split into f new states w.r.t a state variable i following the definition of f-split refinement in Definition. 2. Then the question is: what state variable should DAR+RL blame for the observed instability in an unstable state?

Deliberative Approach. We know that DAR+RL learns an abstract policy $\bar{\pi}$ over abstract states so it maintains and updates the Q-table for abstract states to find the optimal abstract policy. However, DAR+RL also maintains and updates the Q-table for concrete states. This concrete Q-table can be further used for various application such as finding contributing state variables for an unstable state. The notion of deliberative refinement is to refine an unstable state w.r.t a state variable that results in the most consistent new abstract states. Basically, splitting an abstract state over a state variable results in f new abstract states. Now, for each newly created abstract state, DAR+RL goes through the underlying concrete states and calculates the normalized standard deviation of the Q-values. Intuitively, if all concrete states under any of the newly created abstract states have Q-values with small standard deviation for the same action a , then splitting over that state variable would be the near-optimal refinement and can potentially decrease/resolve the instability in the abstract MDP. DAR+RL repeats this process for all state variables and chooses the one that minimizes the standard deviation of the underlying Q-values on the concrete level.

Aggressive Approach. In this approach, DAR+RL splits an unstable state w.r.t all state variables. In other word, DAR+RL blames all state variables for the instability in an abstract state. This method can be employed to avoid keeping the track of the concrete Q-table. The aggressive DAR+RL is essentially effective where the reward is frequent with high variation in an environment.

We implemented the aggressive and deliberative DAR+RL for blaming a state variable to do the refinement of an unstable state. We analyzed the performance of both variants of DAR+RL in Office World and demonstrated that the DAR+RL performs robustly regardless of the choice of this component, as shown in 3.

5 DOMAIN DESCRIPTIONS

Office World. We consider a 36×36 office world scenario with 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

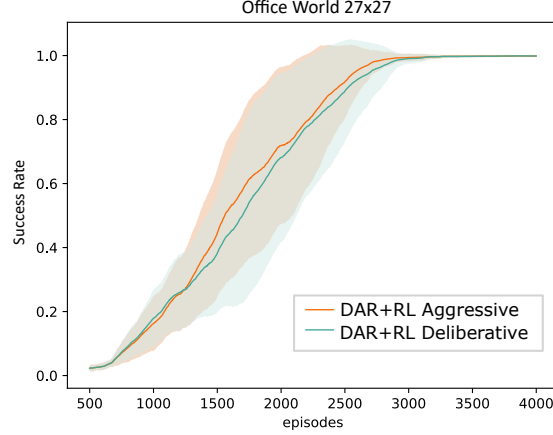


Figure 3: Comparing aggressive and deliberative variants of DAR+RL. We ran both variants 10 times and the shadows show the standard deviation of the measurements.

the action successfully with probability 0.8 and may slip to one of the two adjacent cells with the 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 64×64 Wumpus world with obstacles and pits. The task for the agent is to reach the south-east corner location from the north-west corner location in the grid while avoiding pits. The four actions and the stochastic probabilities are 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 30×30 taxi world scenario 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 for 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 300×300 two dimensional box with 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.