

A APPENDIX

A.1 GOAL-CONDITIONED ARL

This framework can be readily extended to goal-conditioned reinforcement learning (Kaelbling 1993; Schaul et al., 2015), which is also an area of study covered in some prior works on reset-free reinforcement learning (Sharma et al., 2021). Assuming a goal space \mathcal{G} and task-distribution $p_g : \mathcal{G} \mapsto \mathbb{R}_{\geq 0}$, assume that the algorithm $\mathbb{A} : \{s_i, a_i, s_{i+1}\}_{i=0}^{t-1} \mapsto (a_t, \pi_t)$, where $a_t \in \mathcal{A}$ and $\pi_t : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \mapsto \mathbb{R}_{\geq 0}$. Equation 1 can be redefined as follows:

$$J_D(\pi) = \mathbb{E}_{g \sim p_g, s_0 \sim \rho, a_t \sim \pi(\cdot | s_t, g), s_{t+1} \sim p(\cdot | s_t, a_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, g) \right] \quad (3)$$

Additionally, we will assume that the algorithm has access to a set of samples $g_i \sim p(g)$ from the goal distribution. The definitions for deployed policy evaluation and continuing policy evaluation remains the same.

A.2 RELATING PRIOR WORKS TO ARL

In this section, we connect the settings in prior works to our proposed autonomous reinforcement learning formalism. First, consider the typical reinforcement learning approach: Algorithm \mathbb{A} assigns the rewards to transitions using $r(s, a)$, learns a policy π and outputs $\pi_t = \pi$ and $a_t \sim \pi$ (possibly adding noise to the action). The algorithm exclusively optimizes the reward function r throughout training.

Reconsider the door closing example: the agent needs to practice closing the door repeatedly, which requires opening the door repeatedly. However, if we optimize π to maximize r over its lifetime, it will never be incentivized to open the door to practice closing it again. In theory, assuming an ergodic MDP and that the exploratory actions have support over all actions, the agent will open the door given enough time, and the agent will practice closing it again. However, in practice, this can be quite an inefficient strategy to rely on and thus, prior reset-free reinforcement learning algorithms consider other strategies for exploration. To understand current work, we introduce the notion of a surrogate reward function $\tilde{r}_t : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. At every time step t , \mathbb{A} outputs $a_t \sim \pi_e$ and π_t for evaluation, where π_e optimizes \tilde{r}_t over the transitions seen till time $t - 1$ ³. Prior works on reset-free reinforcement learning can be encapsulated within this framework as different choices for surrogate reward function \tilde{r}_t . Some pertinent examples: Assuming r_ρ is some reward function designed that shifts the agent’s state distribution towards initial state distribution ρ , alternating $\tilde{r}_t = r$ and $\tilde{r}_t = r_\rho$ for a fixed number of environment steps recovers the forward-backward reinforcement learning algorithms proposed by Han et al. (2015); Eysenbach et al. (2017). Similarly, R3L (Zhu et al., 2020) can be understood as alternating between a perturbation controller optimizing a state novelty reward and the forward controller optimizing the task reward r . Recent work on using multi-task learning for reset-free reinforcement learning (Gupta et al., 2021) can be understood as choosing $\tilde{r}_t(s_t, a_t) = \sum_{k=1}^K r_k(s_t, a_t) \mathbb{I}[s_t \in \mathcal{S}_k]$ such that $\mathcal{S}_1, \dots, \mathcal{S}_K$ is a partition of the state space \mathcal{S} and only reward function r_k is active in the subset \mathcal{S}_k . Assuming the goal-conditioned autonomous reinforcement learning framework, the recently proposed algorithm VaPRL (Sharma et al., 2021) can be understood as creating a curriculum of goals g_t such that at every step, the action $a_t \sim \pi(\cdot | s_t, g_t)$. The curriculum simplifies the task for the agent, bootstrapping on the success of easier tasks to efficiently improve $J_D(\pi)$.

A.3 ENVIRONMENT DESCRIPTIONS AND REWARD FUNCTIONS

Tabletop-Organization. The Tabletop-Organization task is a diagnostics object manipulation task proposed by (Sharma et al., 2021). The observation space is a 12 dimensional vector consisting of the object position, gripper position, gripper state, and the current goal. The action space is 3-D action that consists of a 2-D position delta and gripper torque. The reward function is a sparse indicator function, $r(s, g) = \mathbb{I}(\|s - g\|_2 \leq 0.2)$. The agent is provided with 12 forward and 12 backward demonstrations.

Sawyer-Door. The Sawyer Door environment consists of a Sawyer robot with a 14 dimension observation space consisting of 3-D effector position, 3-D door position, gripper state and desired

³This can also be captured in a multi-task reinforcement learning framework, where π_e, π_t are the same policy with different task variables as input.

goal. The action space is a 4 dimension action space that consisting of a 3-D end effector control and normalized gripper torque. Let s_d be dimensions of the observation corresponding to the door state, and g be the corresponding goal. The reward function is a sparse indicator function $r(s, g) = \mathbb{1}(\|s_d - g\|_2 \leq 0.08)$. The agent is provided with a 5 forward and 5 backward demos.

Sawyer-Peg. The Sawyer-Peg environment shares observation and action space as the Sawyer-Door environment. Let s_p be the state of the peg and g be the corresponding goal. The reward function is a sparse indicator function $r(s, g) = \mathbb{1}(\|s - g\|_2 \leq 0.05)$. The agent is provided with 10 forward and 10 backward demonstrations for this task.

Franka-Kitchen. The Franka-Kitchen environment consists of a 9-DoF Franka robot with an array of objects (microwave, two distinct burner, door) represented by a 14 dimensional vector. The reward function is composed of a dense reward that is a sum of the euclidean distance between the goal position of the arm and the current state plus shaped reward per object as described in [Gupta et al. \(2019\)](#). No demonstrations are provided for this task.

DHand-LightBulb. The DHand is a 4 fingered robot (16-DoF) mounted on a 6-DoF Sawyer Arm. Let x, y be the current position of the object, x_{goal}, y_{goal} be the corresponding goal, and z by the objects euler angle. The reward function for this task is the following.

$$\begin{aligned}
 R_{bulb} = & - \left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{goal} \\ y_{goal} \end{bmatrix} \right\| - 2 \cdot (|z - z_{goal}|) + \\
 & \mathbb{I} \left\{ \left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{goal} \\ y_{goal} \end{bmatrix} \right\| < 0.1 \right\} \\
 & + \\
 & 10(\mathbb{I} \left\{ \left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{goal} \\ y_{goal} \end{bmatrix} \right\| < 0.1 \text{ AND } |z - z_{goal}| < 0.1 \right\}) - \\
 & \mathbb{I} \left\{ z < \text{threshold} \right\}
 \end{aligned}$$

The observation space of the DHand consists of a 30 dimensional observation and corresponding goal state. The observation is composed of a 16 dimensional hand position, 7 dimensional arm position, 3 dimension object position, 3 dimensional euler angle and a 6 dimensional vector representing the dimensional wise distance to between objects in the environment. The action space is a position delta over the combined 22 DoF of the robot.

Minitaur-Pen. The Minitaur pen’s observation space is the joint positions of its 8 links, their corresponding velocity, current torque, quaternion of its base position, and goal location in the pen. The action space is a 8 dimensional PD target. Let s_b be the 2-D position of the agent, and g be the corresponding goal. Let s_t be the current torques on the agent, and s_v be their velocities. The reward for the agent is a dense reward $r(s, a) = -2.0 \cdot \|s_b - g\| + 0.02 \cdot \|s_v \cdot s_t\|$. We do not provide any demonstrations for this task.

A.4 ALGORITHMS

We use soft actor-critic [Haarnoja et al. \(2018b\)](#) as the base algorithm for our experiments in this paper. When available, the demonstrations are added to the replay buffer at the beginning of training.

Hyperparameter	Value
Actor-critic architecture	fully connected(256, 256)
Nonlinearity	ReLU
RND architecture	fully connected(256, 256, 512)
RND Gamma	0.99
Optimizer	Adam
Learning rate	3e-4
γ	0.99
Target update τ	0.005
Target update period	1
Batch size	256
Classifier batch size	128
Initial collect steps	10^3
Collect steps per iteration	1
Reward scale	1
Min log std	-20
Max log std	2

Table 3: Shared algorithm parameters.

Environment	Training Horizon (H_T)	Evaluation Horizon (H_E)	Replay Buffer Capacity
Tabletop-Organization	200,000	200	20,000,000
Sawyer-Door	200,000	300	20,000,000
Sawyer-Peg	100,000	200	20,000,000
Franka-Kitchen	100,000	400	10,000,000
DHand-Lightbulb	400,000	400	10,000,000
Minitaur-Pen	100,000	1000	10,000,000

Table 4: Environment specific parameters.

A.5 EVALUATION CURVES

In this section, we plot the detailed deployment policy evaluation and continuing policy evaluation curves for different algorithms and different environments:

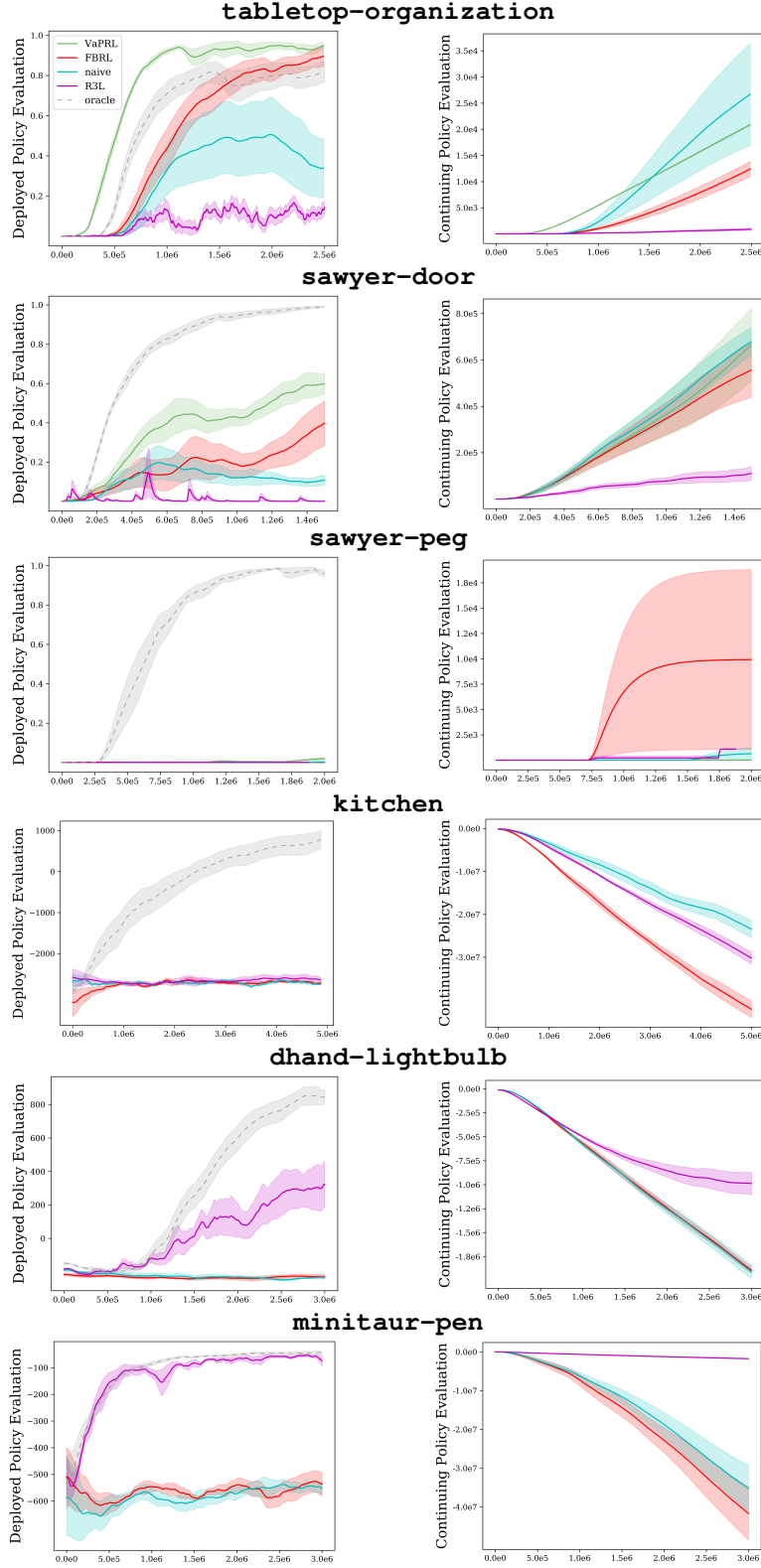


Figure 5: Deployed Policy Evaluation and Continuing Policy Evaluation per environment. Results and averaged over 5 seeds. Shaded regions denote 95% confidence bounds.