A Architecture and Pseudo Code of BPQL

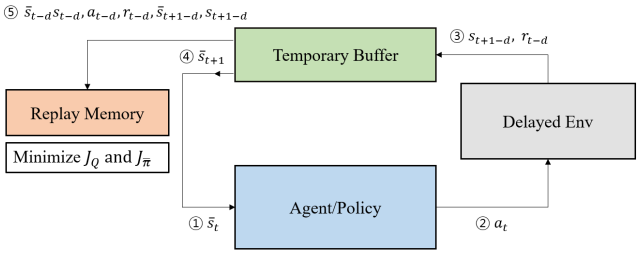

Figure 4: BPQL architecture: The agent selects an action using an augmented state, which is obtained from the temporary buffer, as an input of the policy. At every time $t > 2d$, a transition tuple $\bar{s}_{t-d}, s_{t-d}, a_{t-d}, r_{t-d}, \bar{s}_{t+1-d}, s_{t+1-d}$ is stored in the replay memory, and the policy and beta critic are trained by minimizing $J_{\bar{\pi}}$ and J_{Q_β}, respectively.

Algorithm 1 Belief-Projection-Based Q-Learning (BPQL)

initialize a policy $\bar{\pi}_\phi(a|\bar{s})$, beta critics $Q_{\theta,\beta}(s,a)$, target beta critics $Q^{\text{tar}}_{\theta,\beta}(s,a)$, an empty replay memory $\mathcal{D}$, a temporary buffer $\mathcal{B}$, delayed timesteps d, learning rate for the beta critic λ_{Q_β}, learning rate for the policy $\lambda_{\bar{\pi}}$, and soft update ratio τ
for $episode = 1$ to E **do**
  for $t = 1$ to H **do**
    if $t < d$ **then**
      executes action a_t randomly or 'No-Op' action
      put a_t to $\mathcal{B}$
    else if $t = d$ **then**
      choose action a_d randomly or 'No-Op' action
      $s_1 \leftarrow$Env(a_d)
      put s_1 and a_d to $\mathcal{B}$
    else
      get $s_{t-d}, a_{t-d}, \ldots, a_{t-1}$ from $\mathcal{B}$
      $\bar{s}_t \leftarrow (s_{t-d}, a_{t-d}, \ldots, a_{t-1})$
      $a_t \leftarrow \pi_\phi(\bar{s}_t)$
      $s_{t+1-d}, r_{t-d} \leftarrow$Env$(a_t)$
      put s_{t+1-d} and a_t to $\mathcal{B}$
      if $t > 2d$ **then**
        get $s_{t-2d}, s_{t-2d+1}, s_{t-d}, a_{t-2d}, \ldots, a_{t-d}$ from $\mathcal{B}$
        $\bar{s}_{t-d} \leftarrow (s_{t-2d}, a_{t-2d}, \ldots, a_{t-d-1})$
        $\bar{s}_{t-d+1} \leftarrow (s_{t-2d+1}, a_{t-2d+1}, \ldots, a_{t-d})$
        store $(\bar{s}_{t-d}, s_{t-d}, a_{t-d}, r_{t-d}, \bar{s}_{t+1-d}, s_{t+1-d})$ in $\mathcal{D}$
        pop s_{t-2d}, a_{t-2d} from $\mathcal{B}$
      end if
    end if
  end for
  for each gradient step **do**
    $\omega \leftarrow \omega - \lambda_{Q_\beta} \nabla_\theta J_{Q_\beta}(\omega)$
    $\phi \leftarrow \phi - \lambda_{\bar{\pi}} \nabla_\phi J_{\bar{\pi}}(\phi)$
    $\bar{\omega} \leftarrow \tau\omega + (1-\tau)\bar{\omega}$
  end for
end for

B Proof of Proposition 3.2

Proposition 3.2 *Let the projection operator on the column-space of the belief matrix* $\boldsymbol{B}$ *with respect to the weighted Euclidean norm* $||\cdot||_{\boldsymbol{W}}$ *be* $\Pi_{\boldsymbol{W}}$*, where* $\boldsymbol{W} = [\rho^{\bar{\pi}}(\bar{s}_1), \rho^{\bar{\pi}}(\bar{s}_2), ..., \rho^{\bar{\pi}}(\bar{s}_j)]$ *is a steady-augmented state probability vector, and the Markov chain be irreducible i,e,* $\rho^{\bar{\pi}}(\bar{s}_k) > 0$ *for all* $k \in \{1, 2, ..., j\}$*. Then the combined operator* $\Pi_{\boldsymbol{W}}\bar{T}^{\bar{\pi}}$ *is* γ*-contraction with respect to* $||\cdot||_{\boldsymbol{W}}$*.*

Proof. Let $\mathbf{V}^1_{\text{proj.}}$ and $\mathbf{V}^2_{\text{proj.}}$ be an arbitrary vector in $\mathbb{R}^j$, $\bar{P}^{\bar{\pi}}(\bar{s}'|\bar{s})$ be a transition probability of $\bar{s} \to \bar{s}'$ when the agent follow the policy $\bar{\pi}$, and $V^{1,\text{proj.}}_k, V^{2,\text{proj.}}_k$ be the k-th element of the $\mathbf{V}^1_{\text{proj.}}$ and $\mathbf{V}^2_{\text{proj.}}$ respectively.

$$
\begin{aligned}
&||\Pi_{\mathbf{W}}\bar{T}^{\bar{\pi}}\mathbf{V}^1_{\text{proj.}} - \Pi_{\mathbf{W}}\bar{T}^{\bar{\pi}}\mathbf{V}^2_{\text{proj.}}||^2_{\mathbf{W}} && (28)\\
&= ||\Pi_{\mathbf{W}}(\bar{T}^{\bar{\pi}}\mathbf{V}^1_{\text{proj.}} - \bar{T}^{\bar{\pi}}\mathbf{V}^2_{\text{proj.}})||^2_{\mathbf{W}} && (29)\\
&\leq ||\Pi_{\mathbf{W}}(\bar{T}^{\bar{\pi}}\mathbf{V}^1_{\text{proj.}} - \bar{T}^{\bar{\pi}}\mathbf{V}^2_{\text{proj.}})||^2_{\mathbf{W}} + ||(\mathbf{I} - \Pi_{\mathbf{W}})(\bar{T}^{\bar{\pi}}\mathbf{V}^1_{\text{proj.}} - \bar{T}^{\bar{\pi}}\mathbf{V}^2_{\text{proj.}})||^2_{\mathbf{W}} && (30)\\
&= ||\bar{T}^{\bar{\pi}}\mathbf{V}^1_{\text{proj.}} - \bar{T}^{\bar{\pi}}\mathbf{V}^2_{\text{proj.}}||^2_{\mathbf{W}} && (31)\\
&= ||\bar{\mathbf{R}} + \gamma\bar{\mathbf{P}}\mathbf{V}^1_{\text{proj.}} - (\bar{\mathbf{R}} + \gamma\bar{\mathbf{P}}\mathbf{V}^2_{\text{proj.}})||^2_{\mathbf{W}} && (32)\\
&= \gamma\Sigma^j_{k=1}\rho^{\bar{\pi}}(\bar{s}_k)\left(\Sigma^j_{l=1}\bar{P}^{\bar{\pi}}(\bar{s}_l|\bar{s}_k)\left(V^{1,\text{proj.}}_l - V^{2,\text{proj.}}_l\right)\right)^2 && (33)\\
&\leq \gamma\Sigma^j_{k=1}\rho^{\bar{\pi}}(\bar{s}_k)\Sigma^j_{l=1}\bar{P}^{\bar{\pi}}(\bar{s}_l|\bar{s}_k)\left(V^{1,\text{proj.}}_l - V^{2,\text{proj.}}_l\right)^2 && (34)\\
&= \gamma\Sigma^j_{l=1}\Sigma^j_{k=1}\rho^{\bar{\pi}}(\bar{s}_k)\bar{P}^{\bar{\pi}}(\bar{s}_l|\bar{s}_k)\left(V^{1,\text{proj.}}_l - V^{2,\text{proj.}}_l\right)^2 && (35)\\
&= \gamma\Sigma^j_{l=1}\rho^{\bar{\pi}}(\bar{s}_l)\left(V^{1,\text{proj.}}_l - V^{2,\text{proj.}}_l\right)^2 && (36)\\
&\qquad\qquad \because \Sigma^j_{k=1}\rho^{\bar{\pi}}(\bar{s}_k)\bar{P}^{\bar{\pi}}(\bar{s}_l|\bar{s}_k) = \rho^{\bar{\pi}}(\bar{s}_l) \text{ by the definition of } \rho^{\bar{\pi}}\\
&= \gamma||\mathbf{V}^1_{\text{proj.}} - \mathbf{V}^2_{\text{proj.}}||^2_{\mathbf{W}}, && (37)
\end{aligned}
$$

where Equality (31) holds by the Pythagorean Theorem, and Inequality (34) follows from the Jensen's inequality. □

C Environment Details

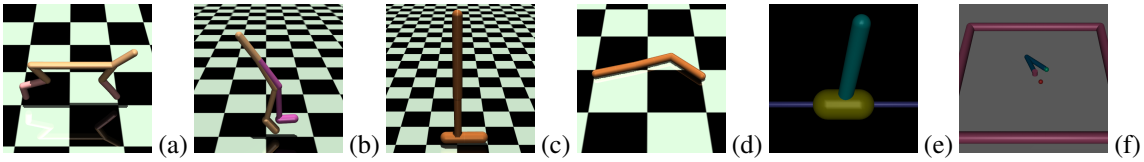

Figure 5: MuJoCo continuous control environments in the experiment: (a) HalfCheetah-v3, (b) Walker2d-v3, (c) Hopper-v3, (d) Swimmer-v3, (e) InvertedPendulum-v2, and (f) Reacher-v2.

Table 2: Details of the MuJoCo environment

	Action dimension	State dimension	Timestep (s)
HalfCheetah-v3	6	17	0.05
Walker2d-v3	6	17	0.008
Hopper-v3 (s)	3	11	0.008
Swimmer-v3	2	6	0.04
InvertedPendulum-v2	1	4	0.04
Reacher-v2	2	11	0.02

D Implementation Details

In this section, we provide the implementation details of the algorithms used in this study. All methods presented in the experiment used an SAC as their base learning algorithm with the following characteristics:

- Stochastic Gaussian policy approaches.
- Automated entropy adjustment [Haarnoja et al., 2018].
- Clipped-double Q-learning, which was introduced in the TD3 algorithm to prevent overestimating the Q-value [Fujimoto et al., 2018].
- Soft target update, which changes the target values slowly and improves the learning stability[Lillicrap et al., 2015].
- Adam optimizer, a variant of the stochastic gradient descent method [Kingma and Ba, 2014].

The details of the hyperparameters are presented in Table 3.

Table 3: Hyperparameters for BPQL and the baselines.

Hyperparameters	**Values**
Critic network	256, 256
Policy network	256, 256
Discount factor	0.99
Replay memory size	1 M
Minibatch size	256
Learning rate	0.0003
Target entropy	-dim$\|\mathcal{A}\|$
Target smoothing coefficient	0.995
Optimizer	Adam

E Plots of Performance Comparison

In this section, we present several plots of the performance curves of BPQL and other baselines in the environments with delayed timesteps of 3, 6 and 9.

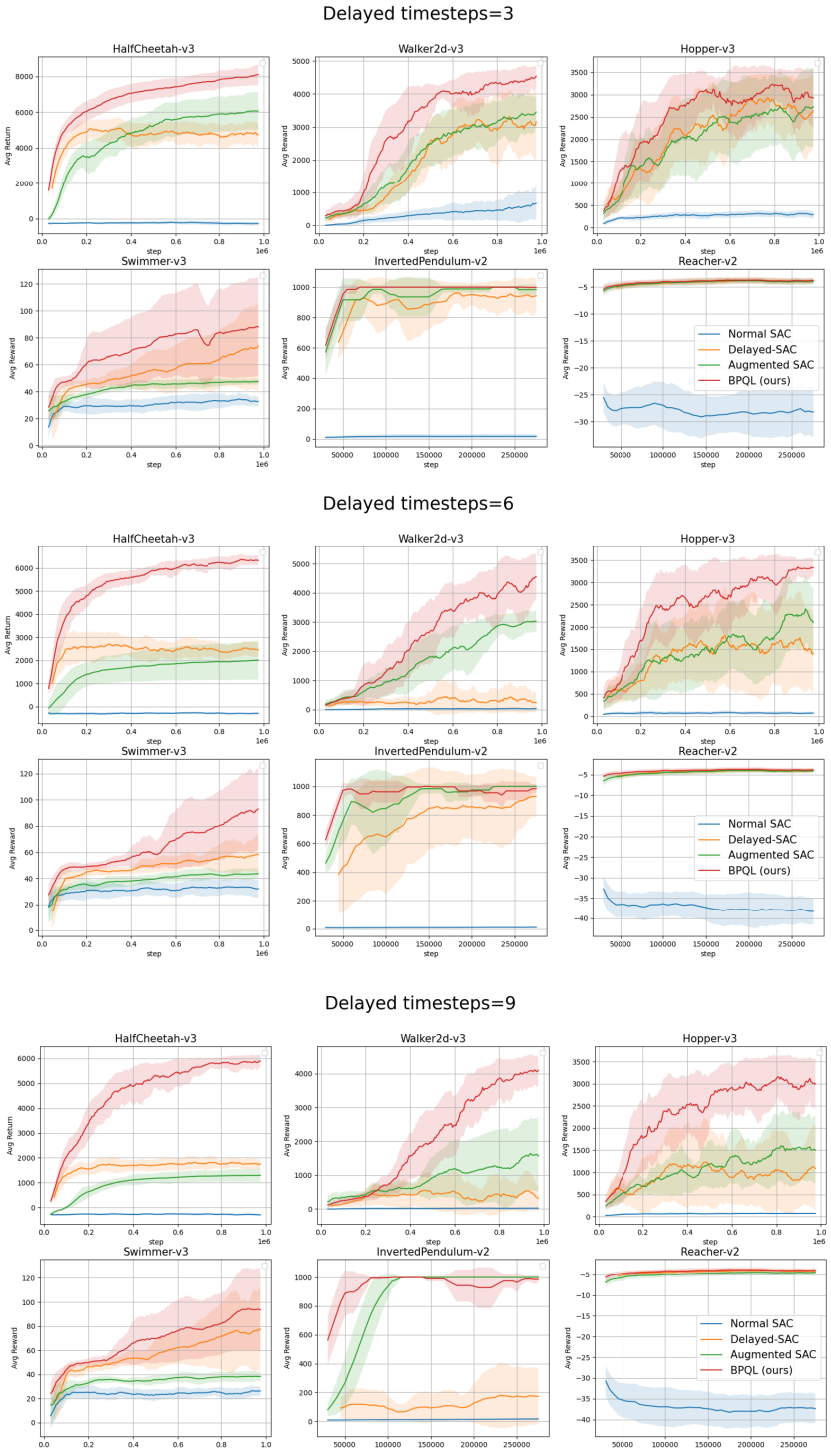

Figure 6: Performance curves for each algorithm in continuous tasks with three, six and nine delayed timesteps.

F Additional Experiments

F.1 Extremely Long Delayed Environment

We conducted additional experiments to determine how well BPQL could solve the control problem even in very long delayed environments. Figure 7 shows that in BPQL, the policy is improved through interaction with the environment in a very long delayed environment, but the conventional methods find it difficult improve their policy.

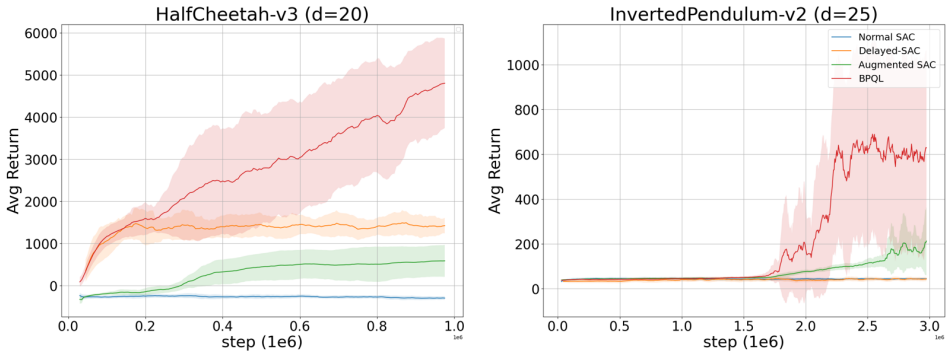

Figure 7: HalfCheetah-v3 environment with 20 delayed timesteps (*left*) and InvertedPendulum-v2 environment with 25 delayed timesteps (*right*). Each timestep is 0.05 s and 0.04 s for HalfCheetah-v3 and InvertedPendulum-v2 environments, respectively. In the InvertedPendulum-v2 environment, to prevent early failure, the agent uses a pretrained policy to determine the actions of the first 25 (=number of delayed timesteps) timesteps. We repeated the test for the tasks five times with different random seeds.

F.2 Action delay

We also evaluated BPQL and other baselines in *action delayed* environments. Figure 8 illustrates the interaction between an agent and action delayed environment.

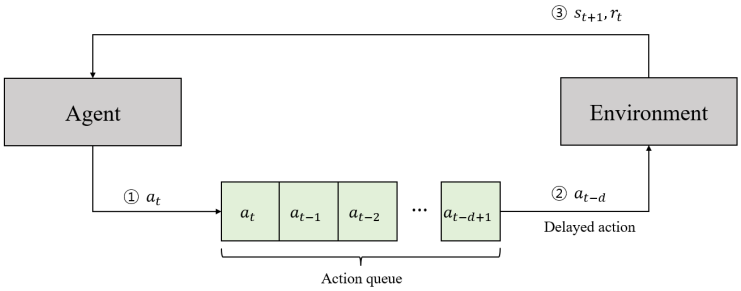

Figure 8: In an action delayed environment, the environment takes the delayed action a_{t-d} instead of the current action a_t, where d is the number of delayed timesteps.

F.2.1 Augmented State in an Action Delayed Environment

In an action delayed environment, the augmented state at time $t+d$ is defined as:

$$\bar{s}_{t+d} = (s_t, a_{t-d}, a_{t-d+1}, \cdots, a_{t-1}), \tag{38}$$

where the d is the number of delayed timesteps for an action. The CDMDP for an action delayed environment is defined as a tuple $(\mathcal{X}, \mathcal{A}, R, P, \gamma, d_a)$, where $\mathcal{X}$ is the original state space, $\mathcal{A}$ is the action space, $R : \mathcal{X} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $P : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \mapsto [0, 1]$ is the transition kernel, $\gamma \in (0, 1)$ is a discount factor, and d_a is the number of delayed timesteps for an action.

This CDMDP can be reduced to a MDP $(\mathcal{S}, \mathcal{A}, \bar{\mathcal{R}}, \bar{P}, \gamma)$, where $\mathcal{S} = \mathcal{X} \times \mathcal{A}^{d_a}$ is an augmented state space, $\bar{\mathcal{R}} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward mapping, and $\bar{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition kernel as in the case of observation delayed environments [Katsikopoulos and Engelbrecht, 2003]. BPQL for action delayed environments uses the augmented state defined in Equation (38) instead of the one defined in Equation (6).

F.2.2 Results

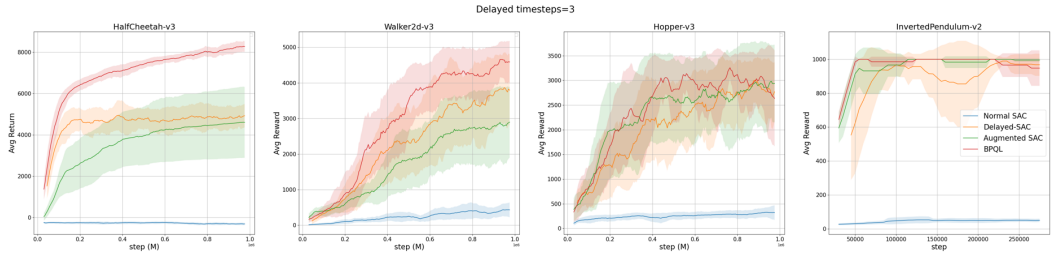

Figure 9: Performance curves for each algorithm in continuous tasks with 3 action delayed timesteps. We repeated the test for the tasks five times with different random seeds.

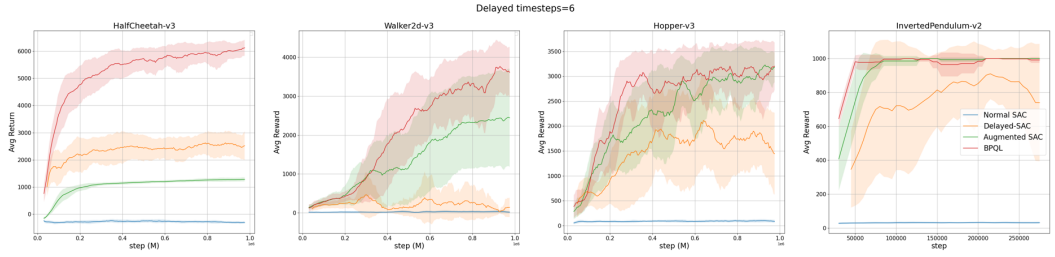

Figure 10: Performance curves for each algorithm in continuous tasks with 6 action delayed timesteps. We repeated the test for the tasks five times with different random seeds.

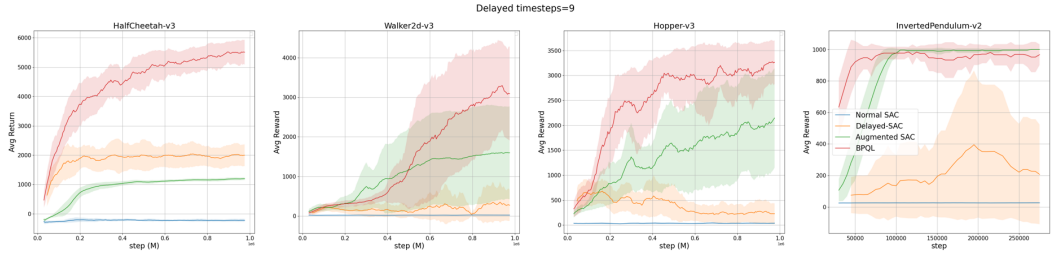

Figure 11: Performance curves for each algorithm in continuous tasks with 9 action delayed timesteps. We repeated the test for the tasks five times with different random seeds.