Asymmetric DQN for Partially Observable Reinforcement Learning (Supplementary Material)

Andrea Baisero[1] **Brett Daley**[1] **Christopher Amato**[1]

[1]Khoury College of Computer Sciences, Northeastern University, Boston, Massachusetts, USA

A PROOFS

A.1 PROOF OF LEMMA 3.1

We first note that B_π may also be expressed as $B_\pi \colon Q \mapsto R + \gamma P_\pi Q$, where $P_\pi \colon \mathcal{Q} \to \mathcal{Q}$ is the linear operator defined by $P_\pi Q(h,a) \doteq \mathbb{E}_{o|h,a}\left[Q(hao, \pi(hao))\right]$, and has operator ∞-norm $\|P_\pi\| = 1$. Next, we show that B_π is a γ-contraction in ∞-norm, i.e., for any two $Q, Q' \in \mathcal{Q}$, the following inequality holds:

$$\begin{aligned}
\|B_\pi Q - B_\pi Q'\| &= \|R + \gamma P_\pi Q - R - \gamma P_\pi Q'\| \\
&= \gamma \|P_\pi \left(Q - Q'\right)\| \\
&\leq \gamma \|P_\pi\| \|Q - Q'\| \\
&= \gamma \|Q - Q'\| \,. \qquad (1)
\end{aligned}$$

Therefore, we conclude that B_π has a unique fixed point which is Q^π by definition.

A.2 PROOF OF LEMMA 3.2

Proof. We show that B is a γ-contraction in ∞-norm, i.e., for any two $Q, Q' \in \mathcal{Q}$, the following inequality holds:

$$\begin{aligned}
\|BQ - BQ'\| &= \max_{h,a} \left| R(h,a) + \gamma\, \mathbb{E}_{o|h,a} \max_{a'} Q(hao, a') - R(h,a) - \gamma\, \mathbb{E}_{o|h,a} \max_{a'} Q'(hao, a') \right| \\
&= \gamma \max_{h,a} \left| \mathbb{E}_{o|h,a} \left[\max_{a'} Q(hao, a') \right] - \mathbb{E}_{o|h,a} \left[\max_{a'} Q'(hao, a') \right] \right| \\
&= \gamma \max_{h,a} \left| \mathbb{E}_{o|h,a} \left[\max_{a'} Q(hao, a') - \max_{a'} Q'(hao, a') \right] \right| \\
&\leq \gamma \max_{h,a} \mathbb{E}_{o|h,a} \left[\left| \max_{a'} Q(hao, a') - \max_{a'} Q'(hao, a') \right| \right] \\
&\leq \gamma \max_{h,a,o} \left| \max_{a'} Q(hao, a') - \max_{a'} Q'(hao, a') \right| \\
&\leq \gamma \max_{h,a,o,a'} \left| Q(hao, a') - Q'(hao, a') \right| \\
&= \gamma \|Q - Q'\| \,. \qquad (2)
\end{aligned}$$

Therefore, we conclude that B has a unique fixed point which is Q^* by definition. □

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

A.3 PROOF OF LEMMA 3.3

Proof. We first note that B_π may also be expressed as $B_\pi : U \mapsto R + \gamma P_\pi U$, where $P_\pi : \mathcal{U} \to \mathcal{U}$ is the linear operator defined by $P_\pi U(h,s,a) \doteq \mathbb{E}_{s',o|s,a}\left[U(hao, s', \pi(hao))\right]$, and has operator ∞-norm $\|P_\pi\| = 1$. Next, we show that B_π is a γ-contraction in ∞-norm, i.e., for any two $U, U' \in \mathcal{U}$, the following inequality holds:

$$\begin{aligned}
\|B_\pi U - B_\pi U'\| &= \|R + \gamma P_\pi U - R - \gamma P_\pi U'\| \\
&= \gamma \|P_\pi \left(U - U'\right)\| \\
&\leq \gamma \|P_\pi\| \|U - U'\| \\
&= \gamma \|U - U'\| \,. \qquad (3)
\end{aligned}$$

Therefore, we conclude that B_π has a unique fixed point which is U^π by definition. □

A.4 PROOF OF LEMMA 4.2 (ASYMMETRIC BELLMAN EQUIVALENCE)

Proof. We assume $Q = EU$, and show that the following elementwise identity holds,

$$\begin{aligned}
(EB_{g(Q)}U)(h,a) &= \mathbb{E}_{s|h}\left[R(s,a) + \gamma\, \mathbb{E}_{s',o|s,a}\left[U(hao, s', \operatorname*{argmax}_{a'} Q(hao,a'))\right]\right] \\
&= R(h,a) + \gamma\, \mathbb{E}_{s|h}\left[\mathbb{E}_{s',o|s,a}\left[U(hao, s', \operatorname*{argmax}_{a'} Q(hao,a'))\right]\right] \\
&= R(h,a) + \gamma\, \mathbb{E}_{s',o|h,a}\left[U(hao, s', \operatorname*{argmax}_{a'} Q(hao,a'))\right] \\
&= R(h,a) + \gamma\, \mathbb{E}_{o|h,a}\left[\mathbb{E}_{s'|hao}\left[U(hao, s', \operatorname*{argmax}_{a'} Q(hao,a'))\right]\right] \\
&= R(h,a) + \gamma\, \mathbb{E}_{o|h,a}\left[Q(hao, \operatorname*{argmax}_{a'} Q(hao,a'))\right] \\
&= R(h,a) + \gamma\, \mathbb{E}_{o|h,a}\left[\max_{a'} Q(hao,a')\right] \\
&= BQ(h,a) \,. \qquad (4)
\end{aligned}$$

Therefore, $EB_{g(Q)}U = BQ$. □

A.5 PROOF OF THEOREM 4.4 (AQL OPTIMALITY)

Proof. Let (h_k, s_k, a_k) denote the history, state, and action visited at the k-th iteration of the AQL algorithm. To facilitate our analysis, we would like to remove the explicit conditional updates in Equations (13) and (14). We define the binary random process $\chi_k \in \mathcal{U}$ such that

$$\chi_k(h,s,a) = \begin{cases} 1 & \text{if } (h,s,a) = (h_k, s_k, a_k) \\ 0 & \text{otherwise} \end{cases} \,. \qquad (5)$$

Using elementwise multiplication and division, the AQL updates in Equations (13) and (14) can be written as

$$U_{k+1} \leftarrow U_k + \alpha_k \chi_k (B_{g(Q_k)} U_k + w_k - U_k) \,, \qquad (6)$$

$$Q_{k+1} \leftarrow Q_k + \alpha_k (E\chi_k)(EB_{g(Q_k)} U_k + v_k - Q_k) \,. \qquad (7)$$

We note that the noise processes w_k and v_k are not statistically independent because they are computed using a shared transition, which guarantees that $v_k = Ew_k$. This allows us to prove that U_k and Q_k remain mutually consistent after

each update, i.e., $Q_k = EU_k$, which we show by induction. From the assumed initialization in the theorem, the base case $Q_0 = EU_0$ is satisfied. Assume that $Q_k = EU_k$ holds for the inductive hypothesis. It follows that

$$\begin{aligned} EU_{k+1} &= E(U_k + \alpha_k \chi_k (B_{g(Q_k)} U_k + w_k - U_k)) \\ &= EU_k + \alpha_k (E\chi_k)(EB_{g(Q_k)} U_k + Ew_k - EU_k)) \\ &= Q_k + \alpha_k (E\chi_k)(EB_{g(Q_k)} U_k + v_k - Q_k) \\ &= Q_{k+1}\,, \end{aligned} \tag{8}$$

and therefore U_k and Q_k are mutually consistent for all $k \geq 0$. By Lemma 4.2, Equation (7) reduces to

$$Q_{k+1} = Q_k + \alpha_k (E\chi_k)(BQ_k + v_k - Q_k)\,. \tag{9}$$

Now let $p_k(h, s, a) \in [0, 1]$ be the probability that $(h, s, a) = (h_k, s_k, a_k)$ conditioned on iteration $k - 1$. Additionally, let $\mathbf{1}$ denote vectors whose components are all equal to one. We can equivalently express Equations (13) and (14) in the form

$$U_{k+1} = (\mathbf{1} - \alpha_k p_k) U_k + \alpha_k p_k (B_{g(Q_k)} U_k + w'_k)\,, \tag{10}$$

$$Q_{k+1} = (\mathbf{1} - \alpha_k E p_k) Q_k + \alpha_k E p_k (BQ_k + v'_k)\,, \tag{11}$$

where

$$w'_k \doteq w_k + \left(\frac{\chi_k}{p_k} - \mathbf{1}\right)(B_{g(Q_k)} U_k + w_k - U_k)\,, \tag{12}$$

$$v'_k \doteq v_k + \left(\frac{E\chi_k}{Ep_k} - \mathbf{1}\right)(BQ_k + v_k - Q_k)\,. \tag{13}$$

It can be verified that $\mathbb{E}[w'_k] = \mathbb{E}[v'_k] = 0$ and that the conditional variances of w'_k and v'_k are bounded such that Proposition 4.4 of Bertsekas and Tsitsiklis [1995] applies given the conditions on α_k. It follows that Q_k converges with probability 1 to Q^*, the unique fixed point of the contraction mapping B (Lemma 3.2).

The fact that $Q_k \to Q^*$ guarantees the existence of some iteration k^*, with probability 1, such that $g(Q_k) = \pi^*$, for $k \geq k^*$. Therefore,

$$U_{k+1} = (\mathbf{1} - \alpha_k p_k) U_k + \alpha_k p_k (B_{\pi^*} U_k + w'_k)\,, \quad \forall k \geq k^*\,. \tag{14}$$

B_{π^*} is a contraction mapping that admits U^* as its unique fixed point (Lemma 3.3). Once again, Proposition 4.4 of Bertsekas and Tsitsiklis [1995] applies, and we conclude that $U_k \to U^*$ with probability 1.

□

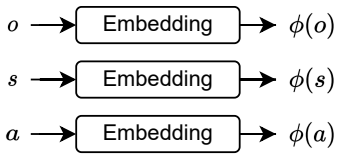
B MODEL ARCHITECTURES

This section contains the model architectures used by each method when run on each environment of our evaluation (see Figure 1). Some components of the architectures are the same for all methods and environments, while some components are domain-specific, e.g. to accommodate the different structures of states and observations in the different environments. In this section, we refer to the **Heaven-Hell-3** and **Heaven-Hell-4** environments as *categorical* environments; to the **Car-Flag** and **Cleaner** environments as *feature-vector* environments; and to the **GV-MemoryFourRooms-7x7** environment as a *gridverse* environment. For a thorough description of each environment, refer to Appendix C of Baisero and Amato [2022].

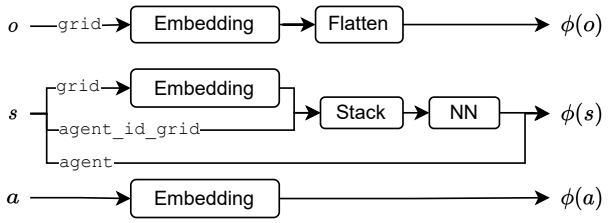
General Architecture The architecture components are shown in Figure 1. Action and observation features are concatenated to form the input to a 128-dimensional single-layer *gated recurrent unit* (GRU) Cho et al. [2014], which acts as the history model $\phi(h)$. The value NN is a feedforward model which varies in each type of environment.

Categorical POMDPs The action, observation, and state feature components are shown in Figure 1a. Categorical environments provide actions, observations, and states, as categorical indices, which we convert to parametric feature vectors using 64-dimensional embedding models. The value NN model is a 2-layer feedforward network with 512 and 256 nodes, and ReLU non-linearities.

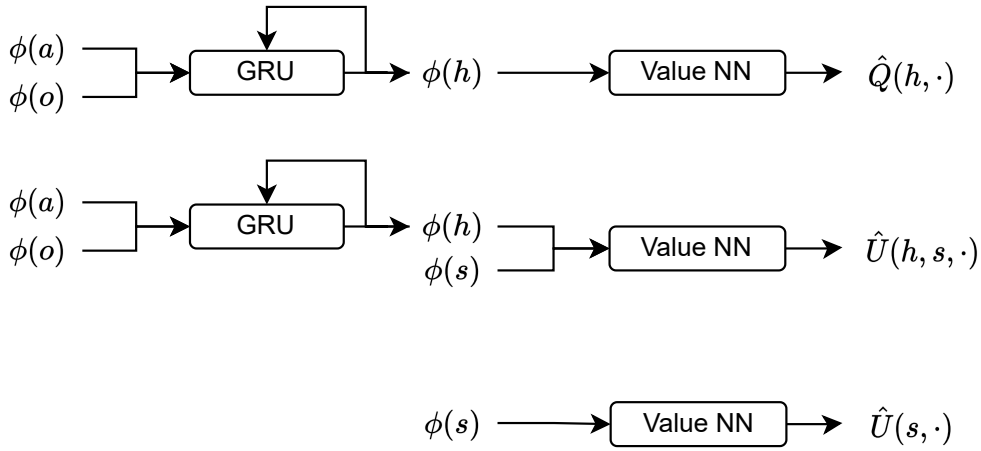
Feature-Vector POMDPs The observations and states provided by the feature-vector environments already come in a feature-vector form, which we do not process further. Actions are modeles as one-hot encodings. The value NN model is a 2-layer feedforward network with 512 and 256, and ReLU non-linearities.



(a) Observation, state, and action models for *categorical* POMDPs.



(b) Observation, state, and action models for *gridverse* POMDPs.



(c) DQN and ADQN architectures. Separate components are used for $\hat{Q}(h,a)$, $\hat{U}(h,s,a)$, and $\hat{U}(s,a)$. In each case, the final layer returns an array of values, one for each action $a \in \mathcal{A}$.

Figure 1: For *categorical* and *gridverse* environments, the observation, state, and action models $\phi(o)$, $\phi(s)$, and $\phi(a)$, are those respectively depicted in Figure 1a and Figure 1b. For the *feature-vector* environments, observation and state models $\phi(o)$ and $\phi(s)$ are directly provided as feature-vectors by the environment itself, while action models $\phi(a)$ are implemented as one-hot encodings. Figure 1c shows the architecture for the DQN models $\hat{Q}(h,a)$, $\hat{U}(h,s,a)$, and $\hat{U}(s,a)$.

Gridverse POMDPs The gridverse environments provide observations and states in a dictionary format containing different fields representing different aspects of the environment, see Appendix C of Baisero and Amato [2022]. Because observations and states already contain a lot of relevant information about the past, we use scalar 1-dimensional embedding models for the actions. The $3 \times 2 \times 3$ observations are first processed using an 8-dimensional embedding layer, and then flattened, which produces a 144-dimensional observation feature $\phi(o)$. The states contain relevant information in different forms, and require a more complex model. The `grid` component is processed using an embedding layer, which is then stacked with the `agent_id_grid` component, and processed by a 1- or $2-$ layer feedforward network (this is hyper-parameter L in Appendix D) with ReLU non-linearities. All outputs of the `grid` and `agent_id_grid` components are then concated to form the overall state feature $\phi(s)$.

C TRAINING DETAILS

We perform *fully episodic* training, by which we mean that various aspects of the training involve and are measured based on complete episodes:

- The replay buffer is more specifically an *episode buffer* which contains full episodes.
- The episode buffer is pre-populated using episodes sampled from a random policy. Episodes are sampled and inserted into the episode buffer until the episode buffer contains a total of $50k$ timesteps.
- The main training loop iterates an environment interaction phase with an optimization phase.
- In the environment interaction phase, a full episode is sampled from the environment using an ϵ-greedy policy based on the current $\hat{Q}$. The sampled episode is then inserted into the episode buffer.
- Any time the episode buffer exceeds a total of $1M$ timesteps, old episodes are removed until that is no longer the case.
- In the optimization phase, a variable number of optimization steps are performed. Optimization steps are performed until the total number of timesteps used for training exceeds the number of total timesteps sampled from the real environment by a given factor. This factor is determined by hyper-parameter F (usually 8 or 16). In each optimization step, a number of full episodes are sampled from the episode buffer, and used fully to form the minibatch of transitions used for the optimization step. Because each episode may contain a variable number of transitions, that means that the size of the minibatch of transitions used for optimization is variable. The number of episodes which is sampled is determined by hyper-parameter B (usually 1).
- The whole process is repeated until a given total number of timesteps have been sampled from the environment.

Although this form of training introduces correlations between the transitions in a minibatch $\{(h, s, a, r, s', o)_i\}_{i=1}^N$, it is also significantly more efficient than if transitions were completely i.i.d, due to the shared computation between the features of consecutive histories.

D HYPERPARAMETERS AND GRID SEARCH

For each combination of control problem and method, we perform a separate grid-search over hyper-parameters, and find the combination of hyper-parameters which results in the best performance, in each case using statistics aggregated over 20 independent runs. The hyper-parameter grid is domain dependent. For **Heaven-Hell-3**, **Heaven-Hell-4**, **Car-Flag**, and **Cleaner**, we search over the following:

- $\alpha \in \{0.0001, 0.0003\}$, the learning rate.
- $N_\epsilon \in \{1M, 2M\}$, the number of timesteps it takes for ϵ to decay linearly from its initial value of 1.0 to its final value of 0.1.
- $F \in \{8, 16\}$, the ratio between number of training timesteps and number of simulation timesteps, used to determine the frequency of optimization steps.
- $B \in \{1, 2\}$, the number of episodes sampled from the episode buffer for each optimization step.

For **GV-MemoryFourRooms-7x7**, we set $\alpha = 0.0001$, and search over the following:

- $N_\epsilon \in \{1M, 2M, 4M\}$, the number of timesteps it takes for ϵ to decay linearly from its initial value of 1.0 to its final value of 0.1.

Table 1: Hyper-parameter grid search results.

Domain	Method	α	N_ϵ	F	B	L
Heaven-Hell-3	**DQN**	0.0003	$2M$	16	2	–
	ADQN	0.0001	$2M$	16	2	–
	ADQN-VR	0.0001	$2M$	16	2	–
	ADQN-State	0.0001	$2M$	16	2	–
	ADQN-State-VR	0.0001	$2M$	16	2	–
Heaven-Hell-4	**DQN**	0.0003	$2M$	16	1	–
	ADQN	0.0001	$2M$	16	2	–
	ADQN-VR	0.0001	$2M$	16	1	–
	ADQN-State	0.0001	$2M$	16	2	–
	ADQN-State-VR	0.0001	$2M$	16	2	–
Car-Flag	**DQN**	0.0003	$1M$	16	2	–
	ADQN	0.0003	$1M$	16	1	–
	ADQN-VR	0.0003	$1M$	16	2	–
	ADQN-State	0.0003	$1M$	16	2	–
	ADQN-State-VR	0.0003	$1M$	16	1	–
Cleaner	**DQN**	0.0001	$2M$	8	2	–
	ADQN	0.0001	$1M$	16	2	–
	ADQN-VR	0.0001	$2M$	16	2	–
	ADQN-State	0.0001	$1M$	8	2	–
	ADQN-State-VR	0.0001	$2M$	8	2	–
GV-MemoryFourRooms-7x7	**DQN**	0.0001	$4M$	16	2	1
	ADQN	0.0001	$1M$	16	2	1
	ADQN-VR	0.0001	$1M$	16	2	1
	ADQN-State	0.0001	$4M$	16	2	1
	ADQN-State-VR	0.0001	$4M$	16	2	1

- $F \in \{8, 16\}$, the ratio between number of training timesteps and number of simulation timesteps, used to determine the frequency of optimization steps.
- $B \in \{2, 4\}$, the number of episodes sampled from the episode buffer for each optimization step.
- $L \in \{1, 2\}$, the number of final linear layers in the state and observation representation models.

Factoring in the 4 control problems with 2^4 combinations of hyper-parameters, and 1 control problem with $3 \cdot 2^3$ combinations of hyper-parameters, 5 methods for each, and 20 independent runs for each combination, we obtain a total number of 8800 independent runs necessary to present all the results of this work. Table 1 shows the hyper-parameters chosen from each grid search, which are the ones used for the results depicted in Figure 1.

References

Andrea Baisero and Christopher Amato. Unbiased Asymmetric Reinforcement Learning under Partial Observability. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 44–52, 2022.

Dimitri P. Bertsekas and John N. Tsitsiklis. Neuro-dynamic programming: an overview. In *Proceedings of 1995 34th IEEE conference on decision and control*, volume 1, pages 560–564. IEEE, 1995.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.