A PROOF OF THEOREM

A.1 PROOF OF THEOREM 3.1

In this section, we analyze the property of $\widehat{Q}$ in finite state-action space $\mathcal{S} \times \mathcal{A}$. The proof of $\lim_{t\to\infty} Q_t = Q^*$ has been well-established in previous work (Robbins & Monro, 1951; Jaakkola et al., 1993; Melo, 2001). Then the proof of $\lim_{t\to\infty} \widehat{Q}_t = \widehat{Q}^*$ is similar. We first prove the empirical Bellman operator Eq. (9) is a γ-contraction operator under the supremum norm. Then when updating in a sampling manner as Eq. (10), it can be considered as a random process. Borrowing an auxiliary result from stochastic approximation, we prove it satisfies the conditions that guarantee convergence. Finally, to prove $\widehat{Q}^*$ lower-bounds Q^*, we rewrite $\widehat{Q}^*(s,a) - Q^*(s,a)$ based on the standard and empirical Bellman operators. When the data covers the whole state-action space, we naturally have $\widehat{Q}^* = Q^*$.

For proof simplicity, we use β denotes policies that interact with the environment and form the current replay memory. We first show existing results for Bellman learning in Eq. (8), and then prove Theorem 3.1 in three steps. The Bellman (optimality) operator $\mathcal{B}$ is defined as:

$$(\mathcal{B}Q)(s,a) = \sum_{s'\in\mathcal{S}} P(s'|s,a)[r + \gamma \max_{a'} Q(s',a')]. \tag{7}$$

Previous works have shown the operator $\mathcal{B}$ is a γ-contraction with respect to supremum norm:

$$\|\mathcal{B}Q_1 - \mathcal{B}Q_2\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty,$$

the supremum norm $\|v\|_\infty = \max_{1\leq i\leq d} |v_i|$, d is the dimension of vector v. Following Banach's fixed-point theorem, Q converges to optimal action value Q^* if we consecutively apply operator $\mathcal{B}$ to Q, $\lim_{n\to\infty} (\mathcal{B})^n Q = Q^*$.

Further, the update rule in Eq. (8), i.e. Q-learning, is a sampling version that applies the γ-contraction operator $\mathcal{B}$ to Q.

$$Q(s,a) \leftarrow r(s,a) + \gamma \max_{a'} Q(s',a'). \tag{8}$$

It can be considered as a random process and will converge to Q^*, $\lim_{t\to\infty} Q_t = Q^*$, with some mild conditions (Szepesvári, 2010; Robbins & Monro, 1951; Jaakkola et al., 1993; Melo, 2001).

Similarly, we define the empirical Bellman (optimality) operator $\hat{\mathcal{B}}$ as:

$$(\hat{\mathcal{B}}\widehat{Q})(s,a) = \sum_{s'\in\mathcal{S}} P(s'|s,a)[r + \gamma \max_{a':\beta(a'|s')>0} \widehat{Q}(s',a')]. \tag{9}$$

And the sampling version we used on the graph is:

$$\widehat{Q}(s,a) \leftarrow r + \gamma \max_{a':\beta(a'|s')>0} \widehat{Q}(s',a'), \tag{10}$$

We split Theorem 3.1 into three lemmas. We first show $\hat{\mathcal{B}}$ is a γ-contraction operator under supremum norm, thus converges to optimal action value $\widehat{Q}^*$, $\lim_{n\to\infty} (\mathcal{B})^n \widehat{Q} = \widehat{Q}^*$. Then we show the sampling-based update rule in Eq. (10) converges to $\widehat{Q}^*$, $\lim_{t\to\infty} \widehat{Q}_t = \widehat{Q}^*$. Finally, we show $\widehat{Q}^*$ lower-bounds Q^*, $\widehat{Q}^*(s,a) - Q^*(s,a) \leq 0, \forall (s,a) \in \mathcal{S}\times\mathcal{A}$. And when the data covers the whole state-action space, i.e. $\beta(a|s) > 0$ for all state-action pairs, we naturally have $\widehat{Q}^*(s,a) = Q^*(s,a)$.

Lemma A.1. *The operator $\hat{\mathcal{B}}$ defined in Eq. (9) is a γ-contraction operator under supremum norm,*

$$\|\hat{\mathcal{B}}\widehat{Q}_1 - \hat{\mathcal{B}}\widehat{Q}_2\|_\infty \leq \gamma \|\widehat{Q}_1 - \widehat{Q}_2\|_\infty.$$

Proof. We can rewrite $\|\hat{\mathcal{B}}\widehat{Q}_1 - \hat{\mathcal{B}}\widehat{Q}_2\|_\infty$ as

$$
\begin{aligned}
&\|\hat{\mathcal{B}}\widehat{Q}_1 - \hat{\mathcal{B}}\widehat{Q}_2\|_\infty \\
&= \max_{s,a} \Big| \sum_{s' \in \mathcal{S}} P(s'|s,a)[r + \gamma \max_{a'_1:\beta(a'_1|s')>0} \widehat{Q}_1(s', a'_1)] - P(s'|s,a)[r + \gamma \max_{a'_2:\beta(a'_2|s')>0} \widehat{Q}_2(s', a'_2)] \Big| \\
&= \max_{s,a} \gamma \Big| \sum_{s' \in \mathcal{S}} P(s'|s,a)[\max_{a'_1:\beta(a'_1|s')>0} \widehat{Q}_1(s', a'_1) - \max_{a'_2:\beta(a'_2|s')>0} \widehat{Q}_2(s', a'_2)] \Big| \\
&\leq \max_{s,a} \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \Big| \max_{a'_1:\beta(a'_1|s')>0} \widehat{Q}_1(s', a'_1) - \max_{a'_2:\beta(a'_2|s')>0} \widehat{Q}_2(s', a'_2) \Big| \\
&\leq \max_{s,a} \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \max_{\tilde{a}:\beta(\tilde{a}|s')>0} \Big| \widehat{Q}_1(s', \tilde{a}) - \widehat{Q}_2(s', \tilde{a}) \Big| \\
&\leq \max_{s,a} \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \max_{\tilde{s},\tilde{a}:\beta(\tilde{a}|\tilde{s})>0} \Big| \widehat{Q}_1(\tilde{s}, \tilde{a}) - \widehat{Q}_2(\tilde{s}, \tilde{a}) \Big| \\
&= \max_{s,a} \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \|\widehat{Q}_1 - \widehat{Q}_2\|_\infty \\
&= \gamma \|\widehat{Q}_1 - \widehat{Q}_2\|_\infty,
\end{aligned}
$$

where the last line follows from $\sum_{s' \in \mathcal{S}} P(s'|s,a) = 1$. □

To show the sampling-based update rule in Eq. (10) converges to $\widehat{Q}^*$, we borrow an auxiliary result from stochastic approximation (Robbins & Monro, 1951; Jaakkola et al., 1993).

Theorem A.2. *The random process $\{\Delta_t\}$ taking values in $\mathbb{R}^n$ and defined as*

$$\Delta_{t+1}(x) = (1 - \alpha_t(x))\Delta_t(x) + \alpha_t(x)F_t(x) \tag{11}$$

converges to zero w.p.1 under the following assumptions:

(1) $0 \leq \alpha_t \leq 1, \sum_t \alpha_t(x) = \infty$ *and* $\sum_t \alpha_t^2(x) < \infty$*;*

(2) $\|\mathbb{E}[F_t(x)|\mathcal{F}_t]\|_W \leq \gamma \|\Delta_t\|_W$*, with* $\gamma < 1$*;*

(3) $Var[F_t(x)|\mathcal{F}_t] \leq C(1 + \|\Delta_t\|_W^2)$*, for* $C > 0$*.*

W is a norm. In our proof it is supremum norm.

Proof. See Robbins & Monro (1951); Jaakkola et al. (1993). □

Lemma A.3. *Given any initial estimation $\widehat{Q}_0$, the following update rule:*

$$\widehat{Q}_{t+1}(s_t, a_t) = \widehat{Q}_t(s_t, a_t) + \alpha_t(x_t, a_t)[r_t + \gamma \max_{a:\beta(a|s_{t+1})>0} \widehat{Q}_t(s_{t+1}, a) - \widehat{Q}_t(s_t, a_t)], \tag{12}$$

converges w.p.1 to the optimal action-value function $\widehat{Q}^*$ if

$$0 \leq \alpha_t(s,a) \leq 1, \quad \sum_t \alpha_t(s,a) = \infty \quad \textit{and} \quad \sum_t \alpha_t^2(s,a) < \infty,$$

for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.

Proof. Based on Theorem A.2, we prove the update rule in Eq. (12) converges.

Rewrite Eq. (12) as

$$\widehat{Q}_{t+1}(s_t, a_t) = (1 - \alpha_t(s_t, a_t))\widehat{Q}_t(s_t, a_t) + \alpha_t(x_t, a_t)[r_t + \gamma \max_{a:\beta(a|s_{t+1})>0} \widehat{Q}_t(s_{t+1}, a)]$$

Subtract $\widehat{Q}^*(s_t, a_t)$ from both sides:

$$\begin{aligned}&\widehat{Q}_{t+1}(s_t, a_t) - \widehat{Q}^*(s_t, a_t)\\ &= (1 - \alpha_t(s_t, a_t))(\widehat{Q}_t(s_t, a_t) - \widehat{Q}^*(s_t, a_t)) + \alpha_t(x_t, a_t)[r_t + \gamma \max_{a:\beta(a|s_{t+1})>0} \widehat{Q}_t(s_{t+1}, a) - \widehat{Q}^*(s_t, a_t)]\end{aligned}$$

Let

$$\Delta_t(s, a) = \widehat{Q}(s, a) - \widehat{Q}^*(s, a) \tag{13}$$

and

$$F_t(s, a) = r + \gamma \max_{a':\beta(a'|s')>0} \widehat{Q}_t(s', a') - \widehat{Q}^*(s, a). \tag{14}$$

We get the same random process shown in Theorem A.2 Eq. (11). Then, proving $\lim_{t\to\infty} \widehat{Q}_t = \widehat{Q}^*$ is the same as proving $\Delta_t(s, a)$ converges to zero with probability 1. We only need to show the assumptions in Theorem A.2 are satisfied under the definitions of Eqs. (13) and (14).

Theorem A.2 (1) is the same as the condition in Lemma A.3. It is easy to achieve, for example, we can choose $\alpha_t(s, a) = 1/t$.

For Theorem A.2 (2), we have

$$\begin{aligned}\mathbb{E}[F_t(s, a)|\mathcal{F}_t] &= \sum_{s'\in\mathcal{S}} P(s'|s, a)[r + \gamma \max_{a':\beta(a'|s')} \widehat{Q}_t(s', a') - \widehat{Q}^*(s, a)]\\ &= (\hat{\mathcal{B}}\widehat{Q}_t)(s, a) - \widehat{Q}^*(s, a)\\ &= (\hat{\mathcal{B}}\widehat{Q}_t)(s, a) - (\hat{\mathcal{B}}\widehat{Q}^*)(s, a)\end{aligned}$$

Thus,

$$\begin{aligned}\|\mathbb{E}[F_t(s, a)|\mathcal{F}_t]\|_\infty &= \|(\hat{\mathcal{B}}\widehat{Q}_t) - (\hat{\mathcal{B}}\widehat{Q}^*)\|_\infty\\ &\le \gamma\|\widehat{Q}_t - \widehat{Q}^*\|_\infty\\ &= \gamma\|\Delta_t\|_\infty,\end{aligned}$$

with $\gamma < 1$.

For Theorem A.2 (3), we have

$$\begin{aligned}Var[F_t(s)|\mathcal{F}_t] &= \mathbb{E}[F_t(s) - \mathbb{E}[F_t(s)|\mathcal{F}_t]|\mathcal{F}_t]^2\\ &= \mathbb{E}[F_t(s) - ((\hat{\mathcal{B}}\widehat{Q}_t)(s, a) - (\hat{\mathcal{B}}\widehat{Q}^*)(s, a))]^2\\ &= \mathbb{E}[r + \gamma \max_{a':\beta(a'|s')>0} \widehat{Q}_t(s', a') - \widehat{Q}^*(s, a) - ((\hat{\mathcal{B}}\widehat{Q}_t)(s, a) - (\hat{\mathcal{B}}\widehat{Q}^*)(s, a))]^2\\ &= \mathbb{E}[r + \gamma \max_{a':\beta(a'|s')>0} \widehat{Q}_t(s', a') - (\hat{\mathcal{B}}\widehat{Q}_t)(s, a)]^2\\ &= Var[r + \gamma \max_{a':\beta(a'|s')>0} \widehat{Q}_t(s', a')|\mathcal{F}_t]\end{aligned}$$

We add and minus a $\widehat{Q}^*$ term to make it close to the RHS in Theorem A.2 (3):

$$Var[r + \gamma \max_{a':\beta(a'|s')>0} \widehat{Q}^*(s', a') + \gamma \max_{a':\beta(a'|s')>0} \widehat{Q}_t(s', a') - \gamma \max_{a':\beta(a'|s')>0} \widehat{Q}^*(s', a')|\mathcal{F}_t]$$

Since r is bounded, thus $r + \gamma \max_{a':\beta(a'|s')>0} \widehat{Q}^*(s', a')$ is bounded. And clearly the second part $\max_{a':\beta(a'|s')>0} \widehat{Q}_t(s', a') - \max_{a':\beta(a'|s')>0} \widehat{Q}^*(s', a')$ can be bounded by $\|\Delta_t\|_\infty$ with some constant. Thus, we have

$$Var[F_t(s)|\mathcal{F}_t] \le C(1 + \|\Delta_t\|_\infty^2),$$

for some constant $C > 0$ under supremum norm. Thus, by Theorem A.2, Δ_t converges to zero w.p.1, i.e., $\widehat{Q}_t$ converges to $\widehat{Q}^*$ w.p.1. □

Lemma A.4. *The value estimation obtained by Eq. (9) lower-bounds the value estimation obtained by Eq. (7):*

$$\widehat{Q}^*(s, a) - Q^*(s, a) \le 0 \tag{15}$$

for all state-action pairs.

Proof. Following the definition of Eqs. (7) and (9), we can rewrite as

$$
\begin{aligned}
&\max_{s,a}(\widehat{Q}^*(s,a) - Q^*(s,a)) \\
&= \max_{s,a}(\hat{\mathcal{B}}\widehat{Q}^*(s,a) - \mathcal{B}Q^*(s,a)) \\
&= \max_{s,a}(\sum_{s'\in\mathcal{S}} P(s'|s,a)[r + \gamma \max_{\hat{a}':\beta(\hat{a}'|s')>0} \widehat{Q}^*(s',\hat{a}')] - \sum_{s'\in\mathcal{S}} P(s'|s,a)[r + \gamma \max_{a'} Q^*(s',a')]) \\
&= \max_{s,a} \sum_{s'\in\mathcal{S}} P(s'|s,a)\gamma(\max_{\hat{a}':\beta(\hat{a}'|s')>0} \widehat{Q}^*(s',\hat{a}') - \max_{a'} Q^*(s',a')) \\
&\leq \max_{s,a} \sum_{s'\in\mathcal{S}} P(s'|s,a)\gamma(\max_{\hat{a}'} \widehat{Q}^*(s',\hat{a}') - \max_{a'} Q^*(s',a')) \\
&\leq \max_{s,a} \sum_{s'\in\mathcal{S}} P(s'|s,a)\gamma \max_{\tilde{a}}(\widehat{Q}^*(s',\tilde{a}) - Q^*(s',\tilde{a})) \\
&\leq \max_{s,a} \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a) \max_{\tilde{s},\tilde{a}}(\widehat{Q}^*(\tilde{s},\tilde{a}) - Q^*(\tilde{s},\tilde{a})) \\
&= \gamma \max_{\tilde{s},\tilde{a}}(\widehat{Q}^*(\tilde{s},\tilde{a}) - Q^*(\tilde{s},\tilde{a})) = \gamma \max_{s,a}(\widehat{Q}^*(s,a) - Q^*(s,a))
\end{aligned}
$$

where the last line follows from $\sum_{s'\in\mathcal{S}} P(s'|s,a) = 1$. Then we have

$$
\begin{aligned}
\max_{s,a}(\widehat{Q}^*(s,a) - Q^*(s,a)) &\leq \gamma \max_{s,a}(\widehat{Q}^*(s,a) - Q^*(s,a)) \\
&\leq \gamma^2 \max_{s,a}(\widehat{Q}^*(s,a) - Q^*(s,a)) \\
&\leq \cdots \\
&\leq \gamma^n \max_{s,a}(\widehat{Q}^*(s,a) - Q^*(s,a))
\end{aligned}
$$

Take limit for both sides and since $0 < \gamma < 1$, we have $\max_{s,a}(\widehat{Q}^*(s,a) - Q^*(s,a)) \leq 0$.

When $\beta(a|s) > 0$ for all state-action pairs, the two contraction operators $\hat{\mathcal{B}}$ and $\mathcal{B}$ are the same. And based on Banach's fixed-point theorem, there is a unique fixed point. Thus $\widehat{Q}^*(s,a) = Q^*(s,a)$ for all state-action pairs., i.e., $\widehat{Q}^*(s,a) - Q^*(s,a) = 0, (s,a) \in \mathcal{S} \times \mathcal{A}$ holds when $\beta(a|s) > 0$ for all state-action pairs. □

Then, we get Theorem 3.1 proved with Lemmas A.1, A.3 and A.4.

B Environment Specifications

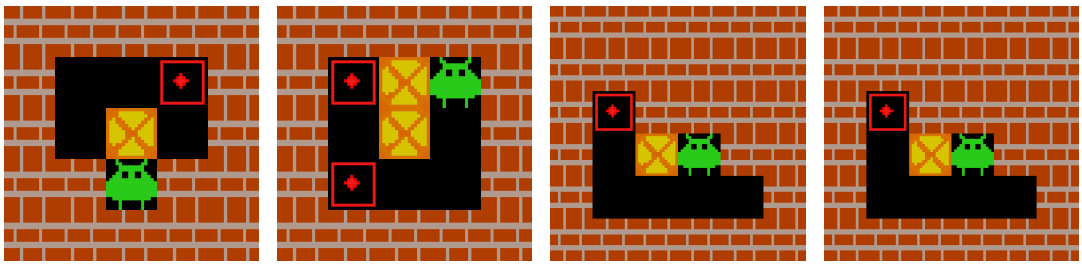
B.1 Sokoban

Sokoban (Schrader, 2018), the Japanese word for 'a warehouse keeper', is a puzzle video game, which is analogous to the problem of having an agent in a warehouse push some specified boxes from their initial locations to target locations. Target locations have the same number of boxes. The goal of the game is to manipulate the agent to move all boxes to the target locations. Specifically, the game is played on a rectangular grid called a room, and each cell of the room is either a floor or a wall. At each new episode, the environment will be reset, which means the layout of the room is randomly generated, including the floors, the walls, the target locations, the boxes' initial locations, and the location of the agent. We choose four tasks with different complexities from Push-5×5-1 to Push-6×6-2, which is shown in Figure 5. The numbers in the task name denote respectively the size of the grid and the number of boxes.

State Space. The state space consists of all possible images displayed on the screen. Each image has the same size as the map, and using the way of dividing each pixel of the image by 255 to normalize into [0,1], we preprocess the image to the inputting state.

Action Space. The action space of Sokoban has a total of eight actions, composed of moving and pushing the box in four directions, which are *left*, *right*, *up*, *down*, *push-left*, *push-right*, *push-up*, *push-down* in detail.

Reward Setting. The agent gets a punishment with a -0.1 reward after each time step. Successfully pushing a box to the target location, can get a +1 reward, and if all boxes are laid in the right locations, the agent can obtain an extra +10 reward. We set the max episode steps to 120, which means the cumulative reward during one episode ranges from -12 to 10 plus the number of boxes.



(a) Push-5x5-1 (b) Push-5x5-2 (c) Push-6x6-1 (d) Push-6x6-2

Figure 5: Visualization of puzzle tasks from Sokoban, which focuses on evaluating the capabilities of agents in spatial reasoning, logical deduction, and long-term planning.

B.2 CraftEnv

Craftenv (Zhao et al., 2023), A Flexible Robotic Construction Environment, is a collection of construction tasks. The agent needs to learn to manipulate the elements, including smartcar, blocks, and slopes, to achieve a target structure through efficient and effective policy. Each construction task is a simulation of the corresponding complex real-world task, which is challenging enough for reinforcement learning algorithms. Meanwhile, the CraftEnv is highly malleable, enabling researchers to design their own tasks for specific requirements. The environment is simple to use since it is implemented by Python and can be rendered using PyBullet. We choose three different designs of the building tasks, shown in Figure 6, to evaluate our algorithm in CraftEnv.

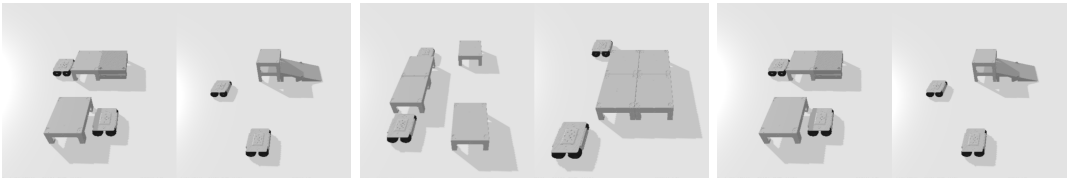
State Space. We assume that the agent can obtain all the information in the map. Therefore, the state consists of all knowledge of smartcar, blocks, folded slopes, unfolded slopes' body, and unfolded slopes' foot, including the position and the yaw angle.

Action Space. The available actions of an agent are designed based on real-world smartcar models, including a total of fifteen actions. Besides all eight directions moving actions, i.e. *forward*, *backward*, *left*, *right*, *left-forward*, *left-backward*, *right-forward*, and *right-backward*, there are interaction-related actions, designed to simulate the building process in the real world. Specifically, the agent can act *lift* and *drop* actions to decide whether or not to carry the surrounding basic element, and can *flod* or *unflod* slopes to build the complex buildings. In addition, the actions of *rotate-left* and *rotate-right* control the agent to rotate the main body to the left and right, and *stop* action is just a non-action.

Reward Setting. CraftEnv is a flexible environment as mentioned above. We can specify our own reward function for different construction tasks. For the relatively simple tasks of building with specified shape requirement, we can use discrete reward, where some reward is given when part of the blueprint is built. While, for building tasks with high complexity, various reward patterns should be designed to encourage the agent to build with different intentions.

C Experimental Details

In this section, we provide the implementation details including basic settings for preference-based RL, architecture of neural network, hyper-parameters and other training detail.



(a) The Strip-shaped Building (b) The Block-shaped Building (c) The Simple Two-Story Building

Figure 6: Visualization of building tasks from CraftEnv. From left to right are The Strip-shaped Building, The Block-shaped Building, and The Simple Two-Story Building task respectively.

C.1 Basic Settings

In the following section, we provide more details of the unsupervised exploration and the uncertainty-based sampling scheme, both of which are mentioned in Section 4.1. These are pivotal techniques in enhancing the feedback efficiency of algorithms, as referenced in Lee et al. (2021b). To ensure an fair comparison, all preference-based RL algorithms in our experiments incorporate both unsupervised exploration and uncertainty-based sampling.

Unsupervised Exploration. The technique of unsupervised exploration in preference-based RL is proposed by Lee et al. (2021b). Designing an intrinsic reward based on the entropy of the state efficiently encourages the agent to visit more diverse states and generate more various behaviors. More specifically, it uses a variant of particle-based entropy (Misra et al., 2003) as the estimation of entropy for the convenience of computation.

Uncertainty-based Sampling. There are some different sampling schemes, including but not limited to uniform sampling, disagreement sampling, and entropy sampling. The latter two sampling schemes are classified as uncertainty-based sampling, which has a better performance compared to uniform sampling intuitively and empirically.

C.2 Architecture and Hyperparameters.

In this section, we describe the architecture of neural networks of the SAC algorithm, which is used as the underlying model. Then we present the full list of hyperparameters of SAC, PEBBLE, and the proposed SEER. The actor of SAC has three layers, specifically, the first layer is the convolutional layer, composed of 16 kernels with a size of 3. Then we squeeze the output into one dimension as the input for the last two fully connected layers. The two Q networks of SAC have the same architecture as that of the actor, one convolutional layer and two fully connected layers. The detailed parameters of the neural network and hyperparameters during learning are shown in table 2. The hyperparameters of PEBBLE and SEER, which are different from those of SAC, are presented in table 3.

Table 2: Hyperparameters of SAC.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Number of layers	3 layers: 1 Conv2d, 2 Linear	Discount	0.99
Number of kernels of Conv2d	16	Batch size	256
Size of Kernel of Conv2d	3	Initial temperature	0.2
Stride of Conv2d	1	(β_1, β_2)	(0.9,0.999)
Padding of Conv2d	0	Update freq	4
Hidden units of hidden layer	128	Critic target update freq	8000
Activation Function	ReLU	Critic τ	1
Actor optimizer	Adam	Exploration	1
Critic optimizer	Adam	Graph τ (Graph-based)	1.0
Learning rate	1e-4	Policy weight (Graph-based)	1.0

Table 3: Hyperparameters of PEBBLE and SEER.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Length of segment	50	Numbers of reward functions / Ensemble size	3
Learning rate	0.0003	Top-k	5
Reward batch size	128	Length of segment (SEER)	20
Reward update	200	Beta β (SEER)	0.5
Frequency of feedback	2000	Graph update batch size (SEER)	32
Number of train steps	1e6	Critic update batch size (SEER)	64
Replay buffer capacity	1e6		