A Dataset Perspective on Offline Reinforcement Learning

Kajetan Schweighofer [*,§] **Andreas Radler** [*,§] **Marius-Constantin Dinu**[*,§,‡]
Markus Hofmarcher [§] **Vihang Patil** [§] **Angela Bitto-Nemling** [§,†]
Hamid Eghbal-zadeh [§] **Sepp Hochreiter** [§,†]

[§] ELLIS Unit Linz and LIT AI Lab,
Institute for Machine Learning,
Johannes Kepler University Linz, Austria
[†] Institute of Advanced Research in Artificial Intelligence (IARAI), Vienna, Austria
[‡] Dynatrace Research, Linz, Austria

Abstract

The application of Reinforcement Learning (RL) in real world environments can be expensive or risky due to sub-optimal policies during training. In Offline RL, this problem is avoided since interactions with an environment are prohibited. Policies are learned from a given dataset, which solely determines their performance. Despite this fact, how dataset characteristics influence Offline RL algorithms is still hardly investigated. The dataset characteristics are determined by the behavioral policy that samples this dataset. Therefore, we define characteristics of behavioral policies as exploratory for yielding high expected information in their interaction with the Markov Decision Process (MDP) and as exploitative for having high expected return. We implement two corresponding empirical measures for the datasets sampled by the behavioral policy in deterministic MDPs. The first empirical measure SACo is defined by the normalized unique state-action pairs and captures exploration. The second empirical measure TQ is defined by the normalized average trajectory return and captures exploitation. Empirical evaluations show the effectiveness of TQ and SACo. In large-scale experiments using our proposed measures, we show that the unconstrained off-policy Deep Q-Network family requires datasets with high SACo to find a good policy. Furthermore, experiments show that policy constraint algorithms perform well on datasets with high TQ and SACo. Finally, the experiments show, that purely dataset-constrained Behavioral Cloning performs competitively to the best Offline RL algorithms for datasets with high TQ.

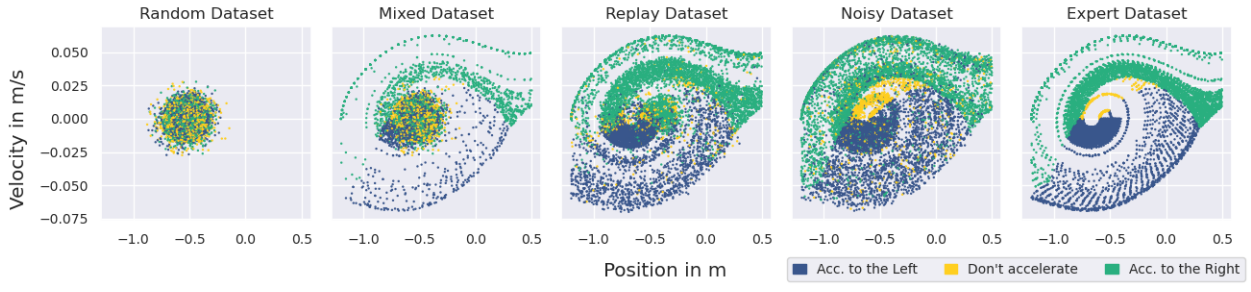


Figure 1: We illustrate the effect of the behavioral policy on the distribution of the sampled dataset. State-action pairs of different datasets were sampled from the `MountainCar` environment using different behavioral policies. One can visually perceive differences between these datasets, which we aim to quantify by our introduced measures (see Sec. 2.4).

[*] Authors contributed equally. The code is available at github.com/ml-jku/OfflineRL

1 INTRODUCTION

Central problems in RL are credit assignment (Sutton, 1984; Arjona-Medina et al., 2019; Holzleitner et al., 2020; Patil et al., 2020; Widrich et al., 2021; Dinu et al., 2022) and efficient exploration of the environment (Wiering and Schmidhuber, 1998; McFarlane, 2003; Schmidhuber, 2010). Exploration can be costly due to high computational complexity, violation of physical constraints, risk of physical damage, interaction with human experts, etc. (Dulac-Arnold et al., 2019). Furthermore, exploration may endanger humans through accidents inflicted by self-driving cars, crashes of production machines when optimizing production processes, or high financial losses when applied in trading or pricing. In limiting cases, simulations may alleviate these factors. However, designing robust and high quality simulators is a challenging, time consuming and resource intensive task, and introduces problems related to distributional shift and domain gap between the real world environment and the simulation (Rao et al., 2020).

Confronted with these tasks, one can utilize the framework of Offline RL (Levine et al., 2020), also referred to as Batch RL (Lange et al., 2012), which offers to learn policies from pre-collected or logged datasets, without interacting with an environment (Agarwal et al., 2020; Fujimoto et al., 2019a;b; Kumar et al., 2020). Many such Offline RL datasets already exist for various real world problems (Cabi et al., 2019; Dasari et al., 2020; Yu et al., 2020). Offline RL shares numerous traits with supervised deep learning, including, but not limited to leveraging large datasets. A core obstacle is generalization to unseen data, as stored samples may not cover the entire state-action space. In Offline RL, the generalization problem takes the form of domain shift (Adler et al., 2020) during inference. Apart from the non-stationarity of an environment or agent to environment interactions, the domain shift may be caused by the data collection process itself (Khetarpal et al., 2020). We illustrate this by collecting datasets using different behavioral policies in Fig. 1.

Multiple Offline RL algorithms (Agarwal et al., 2020; Fujimoto et al., 2019a;b; Gulcehre et al., 2021; Kumar et al., 2020; Wang et al., 2020) have been proposed to address these problems and have shown good results. Well known off-policy algorithms such as Deep Q-Networks (DQN) (Mnih et al., 2013) can readily be used in Offline RL, by initializing the replay-buffer with a pre-collected dataset. In practice, however, those algorithms often fail or lag far behind the performance they attain when trained in an Online RL setting. The reduced performance is attributed to the extrapolation errors for unseen state-action pairs and the resulting domain shift between the fixed given dataset and the states visited by the learned policy (Fujimoto et al., 2019a; Gulcehre et al., 2021). Several algorithmic improvements tackle these problems, including policy constraints (Fujimoto et al., 2019a;b; Wang et al., 2020), regularization of learned action-values (Kumar et al., 2020), and off-policy algorithms with more robust action-value estimates (Agarwal et al., 2020). While unified datasets have been released (Gulcehre et al., 2020; Fu et al., 2021) for comparisons of Offline RL algorithms, grounded work in understanding how the dataset characteristics influence the performance of algorithms is still lacking (Riedmiller et al., 2021; Monier et al., 2020).

We therefore study core dataset characteristics, and derive from first-principles theoretical measures related to exploration and exploitation which are well established policy properties. We derive a measure of exploration based on the *expected information* of the interaction of the behavioral policy in the MDP, the transition-entropy. Furthermore, we show that for deterministic MDPs, the transition-entropy equals the occupancy-entropy. Our measure of exploitation, the expected trajectory return, is a generalization of the expected return of a policy. We show that these measures have theoretical guarantees under MDP homomorphisms (van der Pol et al., 2020), thus exhibit certain stability traits under such transformations.

To characterize datasets and compare them across environments and generating policies, we implement two empirical measures that correspond to the theoretical measures: (1) SACo, corresponding to the occupancy-entropy, defined by the normalized unique state-action pairs, capturing exploration. (2) TQ, corresponding to the expected trajectory return, defined by the normalized average return of trajectories in the dataset, capturing exploitation.

We conducted experiments on six different environments from three different environment suites (Brockman et al., 2016; Chevalier-Boisvert et al., 2018; Young and Tian, 2019), to create datasets with different characteristics (see Sec. 3.1). On these datasets, 6750 RL learning trials were conducted, which cover a selection of popular algorithms in the Offline RL setting (Agarwal et al., 2020; Dabney et al., 2017; Fujimoto et al., 2019b; Gulcehre et al., 2021; Kumar et al., 2020; Mnih et al., 2013; Pomerleau, 1991; Wang et al., 2020). We evaluated their performance on datasets with different TQ and SACo. Variants of the off-policy DQN family (Mnih et al., 2013; Agarwal et al., 2020; Dabney et al., 2017) were found to require datasets with high SACo to perform well. Algorithms that constrain the learned policy towards the distribution of the behavioral policy perform well for datasets with high TQ or SACo or intermediate variations thereof. For datasets with high TQ, Behavioral Cloning (BC) (Pomerleau, 1991) outperforms variants of the DQN family and is competitive to the best performing Offline RL algorithms.

In summary, our contributions are:

(a) we derive theoretical measures that capture exploration and exploitation,

(b) we prove theoretical guarantees for the stability of these measures under MDP homomorphisms,

(c) we provide an effective method to characterize datasets through the empirical measures TQ and SACo,

(d) we conduct an extensive empirical evaluation of how dataset characteristics affect algorithms in Offline RL.

2 Characterizing RL Datasets

The selection of suitable measures for evaluating RL datasets and enabling their comparison with respect to algorithmic performance is a challenging and open research question (Riedmiller et al., 2021; Monier et al., 2020). As the distribution of the dataset is governed by the behavioral policy used to sample it, we aim to find measures for the characteristics of the behavioral policy. We define the characteristics of the behavioral policy as how exploitative and explorative it acts, and analyze the corresponding measures. These are the expected trajectory return for exploitation, and the transition-entropy for exploration in stochastic MDPs, which simplifies to the occupancy-entropy for exploration in deterministic MDPs. Furthermore, we study these theoretical measures under MDP homomorphisms (Ravindran and Barto, 2001; Givan et al., 2003; van der Pol et al., 2020; Abel, 2022) to analyze their dependence on such transformations. Finally, we implement empirical measures that correspond to the expected trajectory return and the occupancy-entropy, TQ and SACo, which can be calculated for a given set of datasets.

We define our problem setting as a finite MDP to be a 5-tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$ of finite set $\mathcal{S}$ with states s (random variable S_t at time t), $\mathcal{A}$ with actions a (random variable A_t), $\mathcal{R}$ with rewards r (random variable R_{t+1}), dynamics $p(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$, and $\gamma \in [0, 1)$ as a discount factor. The agent selects actions $a \sim \pi(S_t = s)$ based on the policy π, which depends on the current state s. Our objective is to find the policy π that maximizes the expected return $G_t = \sum_{k=0}^{T} \gamma^k R_{t+k+1}$. In Offline RL, we assume that a dataset $\mathcal{D} = \{\tau_i\}|_{i=1}^{B}$, consisting of B trajectories, is provided. A single trajectory τ consists of a sequence of (s, a, r, s') tuples.

2.1 Transition Entropy as Measure for Exploration

We define *explorativeness* of a policy π as the expected information of the interaction between the policy and a certain MDP. While the policy actively selects its next action given its current state, it is transitioned into a next state and receives a reward signal according to the dynamics of the MDP $p(r, s' \mid s, a)$, which cannot be influenced by the policy. Therefore, the policy interacting in the MDP can be seen as a single stochastic process that generates transitions (s, a, r, s'). A policy is explorative, if it is able to generate many different transitions with high probability in an MDP. As transitions can only be observed through the interaction between policy and MDP, explorativeness of a policy can only be defined in conjunction with a specific MDP. Policies that act very explorative in one MDP, could act much less explorative in another MDP and vice versa. For instance, a policy that does not open doors might explore multiple rooms very thoroughly if all doors are already open, but will get stuck in a single room if they are closed initially.

We measure explorativeness of a policy by the Shannon entropy (Shannon, 1948) of the transition probabilities $p_\pi(s, a, r, s')$ under the policy interacting with the MDP. We can rewrite the transition probabilities as $p(s, a, r, s') = p(s', r \mid s, a)\, p(s, a)$. The dynamics $p(r, s' \mid s, a)$ are solely MDP-dependent, while the state-action probability $p_\pi(s, a)$ is policy- and MDP-dependent. The state-action probability $p_\pi(s, a)$ is often referred to as occupancy measure[1] $\rho_\pi(s, a)$ (Neu and Pike-Burke, 2020; Ho and Ermon, 2016), as it describes how the policy occupies the state-action space.

We start with the Shannon entropy of the transition probabilities

$$H(p_\pi(s, a, r, s')) \coloneqq - \sum_{\substack{s,a,r,s' \\ p_\pi(s,a,r,s')>0}} p_\pi(s, a, r, s') \log(p_\pi(s, a, r, s')), \tag{1}$$

which we will further refer to as transition-entropy. The transition-entropy can be factored into the occupancy weighted sum of entropies of the dynamics and the occupancy (for a detailed derivation see Eq. 9 in Sec. A.1 in the Appendix):

$$H(p_\pi(s, a, r, s')) = \sum_{s,a} \rho_\pi(s, a)\; H(p(r, s' \mid s, a)) + H(\rho_\pi(s, a)). \tag{2}$$

An explorative policy should therefore aim to find a good balance between visiting all possible state-actions similarly likely, while visiting state-actions pairs with more stochastic dynamics $p(r, s' \mid s, a)$ more often. This link between

[1] To avoid additional assumptions on the MDP, we focus this analysis on a single stage, see Neu and Pike-Burke (2020).

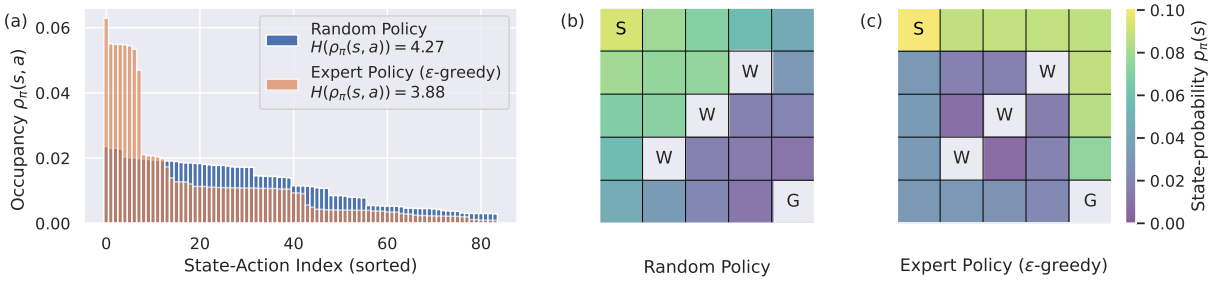


Figure 2: (a) Distribution of occupancies of two policies, a random policy and an expert policy that acts ϵ-greedy with $\epsilon = 0.5$. We show the occupancy-entropies, where the random policy attains a higher value than the ϵ-greedy expert policy. Both policies are evaluated in a five by five deterministic gridworld with four eligible actions (up, down, left, right). Episodes start at the starting position **S** and end upon reaching the goal state **G**, which yields a positive return. Walls **W** cannot be passed through. (b) & (c) Illustrate the state probabilities $p_\pi(s)$ under the random (b) and the ϵ-greedy expert (c) policy, which is the sum over actions of the underlying occupancies $\rho_\pi(s, a)$.

good exploration and visiting more stochastic dynamics is also found in optimal experiment design (Storck et al., 1995; Cohn, 1993). Note that there may be other ways to define exploration. A reasonable option is to define a *utility* measure for transitions to reach a goal or learn a certain property of an MDP. Higher exploration would then mean that more util transitions are more likely. Similarly, another option is to define a *distance* measure for transitions and define exploration as increasing the likelihood to sample transitions which have a higher distance to each other. A priori, it is hard to define the notions of utility or distance, hence we turned to the Shannon entropy.

Deterministic MDPs In this class of MDPs, we can simplify the exploration measure from Eq. 2. Since $p(r, s' \mid s, a)$ is deterministic, the dynamics-entropy $H(p(r, s' \mid s, a))$ is zero and the left term in Eq. 2 vanishes as shown in Eq. 11 in Sec. A.1 in the Appendix. Therefore, for deterministic MDPs the transition-entropy simplifies to the occupancy-entropy:

$$H(\rho_\pi(s,a)) \coloneqq - \sum_{\substack{s,a \\ \rho_\pi(s,a)>0}} \rho_\pi(s,a) \log(\rho_\pi(s,a)). \tag{3}$$

In this special case, a policy explores maximally if all possible state-action pairs are visited equally likely and thus the occupancy-entropy is maximal. Fig. 2 gives an intuition about the occupancy distribution and the resulting occupancy-entropy of different policies.

2.2 Expected Trajectory Return as Measure for Exploitation

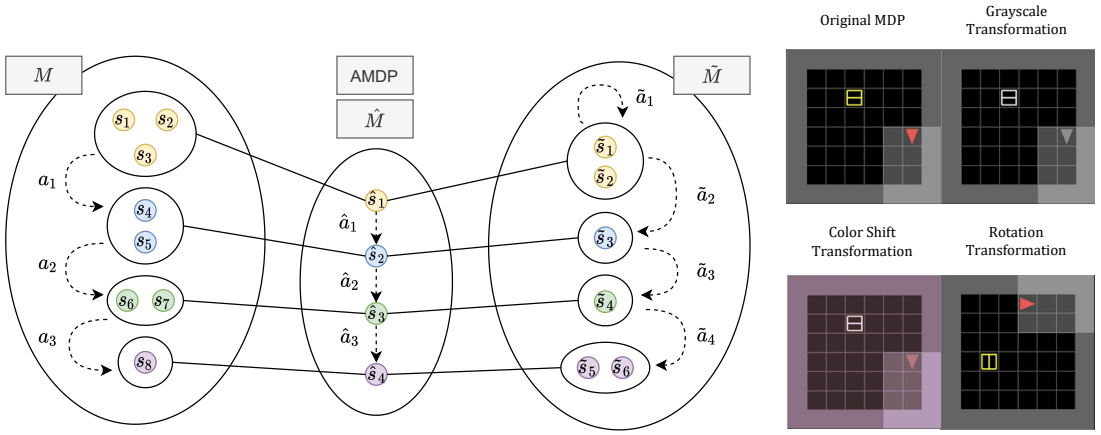
We define *exploitativeness* of a policy as how performant in terms of expected return it interacts with the MDP. We measure how exploitative a policy acts by its expected return $g_\pi = \mathbb{E}_\pi [G_t]$. Furthermore, we generalize the expected return to arbitrary policies (e.g. non-representable or non-Markovian policies as discussed in Fu et al. (2021)), as the Offline RL setting makes no assumptions on the behavioral policy used for dataset collection. Therefore, we define exploitativeness on a distribution of trajectories $\mathcal{T}$ observed from arbitrary policies as expected trajectory return $g_\mathcal{T}$, which is given by:

$$g_\mathcal{T} \coloneqq \mathbb{E}_{\tau \sim \mathcal{T}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid r_t \in \tau \right]. \tag{4}$$

In this turn, we can thus also evaluate policies for which we can not represent the dataset generating behavior but can generate data indefinitely. This is of relevance when considering human-generated datasets.

2.3 Common Structures of MDPs

We are interested in the *stability* of the proposed measures. We regard stability of our measures as the degree to which the measures change under slight modifications of the input. Modifications to the input are small changes in the state space, action space or dynamics of the MDP. To formalize changes of the MDP, we introduce the concept of a common abstract MDP (AMDP) (see Fig. 3). We base the definition of the AMDP on prior work from Sutton et al. (1999); Jong and Stone (2005); Li et al. (2006); Abel (2019); van der Pol et al. (2020); Abel (2022) by considering state and action abstractions.



(a) (b)

Figure 3: (a) Illustration of two homomorphic images of an abstract MDP (AMDP), (b) example of homomorphic transformations on `MiniGrid` (Chevalier-Boisvert et al., 2018). For further details see Sec. A.5 in the Appendix.

Definition 1. *Given two MDPs $M = (\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$ and $\tilde{M} = (\tilde{\mathcal{S}}, \tilde{\mathcal{A}}, \tilde{\mathcal{R}}, \tilde{p}, \gamma)$ with finite and discrete state-action spaces. We assume there exists a common abstract MDP (AMDP) $\hat{M} = (\hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{R}}, \hat{p}, \gamma)$, whereas M and $\tilde{M}$ are homomorphic images of $\hat{M}$. We define an MDP homomorphism by the surjective abstraction functions as $\phi : \mathcal{S} \to \hat{\mathcal{S}}$ and $\tilde{\phi} : \tilde{\mathcal{S}} \to \hat{\mathcal{S}}$, with $\phi(s), \tilde{\phi}(\tilde{s}) \in \hat{\mathcal{S}}$ for the state abstractions and $\{\psi_s : \mathcal{A} \to \hat{\mathcal{A}} \mid s \in \mathcal{S}\}$ and $\{\tilde{\psi}_{\tilde{s}} : \tilde{\mathcal{A}} \to \hat{\mathcal{A}} \mid \tilde{s} \in \tilde{\mathcal{S}}\}$, with $\psi_s(a), \tilde{\psi}_{\tilde{s}}(\tilde{a}) \in \hat{\mathcal{A}}$ for the action abstractions (see Appendix Sec. A.3 for more details regarding the assumptions on these abstraction functions and their implications on the MDP dynamics). Let $\pi(a \mid s)$ and $\tilde{\pi}(\tilde{a} \mid \tilde{s})$ be corresponding policies of M, $\tilde{M}$ such that they map via ϕ, ψ_s and $\tilde{\phi}, \tilde{\psi}_{\tilde{s}}$ to the same abstract policy $\hat{\pi}(\hat{a} \mid \hat{s})$ of the common AMDP $\hat{M}$. Let $\mathcal{T}$ and $\tilde{\mathcal{T}}$ be corresponding trajectory distributions of M, $\tilde{M}$, such that they map via ϕ, ψ_s and $\tilde{\phi}, \tilde{\psi}_{\tilde{s}}$ to the same abstract trajectory distribution $\hat{\mathcal{T}}$ of the common AMDP.*

We use Def. 1 to derive an upper bound of the difference in transition-entropy for two homomorphic images of the same common AMDP. For brevity we write $H(p)$, $H(\tilde{p})$ and $H(\hat{p})$ for the transition-entropies induced by π, $\tilde{\pi}$ and $\hat{\pi}$ respectively.

Theorem 1. *Given two homomorphic images M and $\tilde{M}$ of a common AMDP $\hat{M}$ and their respective transition probabilities $p(s, a, r, s')$, $\tilde{p}(\tilde{s}, \tilde{a}, r, \tilde{s}')$ and $\hat{p}(\hat{s}, \hat{a}, r, \hat{s}')$ induced by corresponding policies π and $\tilde{\pi}$ of M and $\tilde{M}$ and the abstract policy $\hat{\pi}$ of $\hat{M}$ they map to. The maximum absolute difference in transition-entropy is upper bounded by:*

$$|H(p) - H(\tilde{p})| \leqslant \max \Big[H(p), H(\tilde{p}) \Big] - H(\hat{p}) \tag{5}$$

Proof. We use the fact that homomorphic images have a transition-entropy greater or equal to the transition-entropy of the common AMDP. For a proof of this statement, see Sec. A.4 in the Appendix. Therefore, we use $H(p) \geq H(\hat{p})$ and $H(\tilde{p}) \geq H(\hat{p})$. Combining these inequalities leads to Eq. 5. □

Proposition 1. *Any corresponding policies $\pi(a \mid s)$ and $\tilde{\pi}(\tilde{a} \mid \tilde{s})$ of homomorphic images M and $\tilde{M}$ of a common AMDP $\hat{M}$, that map to the same abstract policy $\hat{\pi}(\hat{a} \mid \hat{s})$, exhibit the same expected return $g_\pi = g_{\tilde{\pi}}$. Furthermore, any corresponding trajectory distributions $\mathcal{T}$ and $\tilde{\mathcal{T}}$ of homomorphic images M and $\tilde{M}$ of a common AMDP $\hat{M}$, that map to the same abstract trajectory distribution $\hat{\mathcal{T}}$, exhibit the same expected trajectory return $g_{\mathcal{T}} = g_{\tilde{\mathcal{T}}}$.*

Prop. 1 follows from the conditions in Eq. 18 and Eq. 19 for M and $\tilde{M}$ respectively. For completeness we mention that for *isomorphic* transformations of MDPs (i.e. under an MDP homomorphism with bijective abstraction functions), it follows directly that the measures discussed in Sec. 2.1 and Sec. 2.2 are preserved.

We conclude: (1) Policies, defined on homomorphic images of a common AMDP which map to the same abstract policy, yield the same expected return. Trajectory distributions on homomorphic images that have a corresponding abstract trajectory distribution, yield the same expected trajectory return. (2) From Eq. 5 we can formally confirm

the intuition about the stability of our exploration measures between homomorphic images of a common AMDP: If the transition-entropy of the *greatest* common AMDP of two corresponding MDPs increases, the difference in their transition-entropies decreases.

2.4 Empirical Measures

We aim to implement empirical measures that are applicable to a set of given datasets and correspond to the theoretical measures of exploration and exploitation of a policy introduced in Sec. 2.1 and Sec. 2.2. Furthermore, these measures are normalized to references, to allow for a comparison of datasets sampled from different MDPs.

SACo. First, we implement an empirical measure that corresponds to the theoretical measure of exploration of a policy in a deterministic MDP, given by Eq. 3. One way to implement such a measure is the naïve entropy estimator of a dataset $\hat{H}(\mathcal{D})$ or its corresponding perplexity estimator $\hat{PP}(\mathcal{D}) = e^{\hat{H}(\mathcal{D})}$. The entropy estimator is upper bounded by the logarithm of *unique* state-action pairs in the dataset $\log(u_{s,a}(\mathcal{D}))$, thus $\hat{H}(\mathcal{D}) \leqslant \log(u_{s,a}(\mathcal{D}))$. It follows, that the perplexity estimator is upper-bounded by the number of unique state-action pairs in the dataset $\hat{PP}(\mathcal{D}) \leqslant u_{s,a}(\mathcal{D})$. We therefore base our empirical measure SACo on the unique state-action pairs $u_{s,a}(\mathcal{D})$, which has the benefit of being easy to interpret while corresponding to the exploration of the policy. Additionally, $u_{s,a}(\mathcal{D})$ captures the notion of *coverage of the state-action space* given a dataset, that is invoked in the Offline RL literature (Fujimoto et al., 2019a; Agarwal et al., 2020; Kumar et al., 2020; Monier et al., 2020). A further theoretical analysis on the relation between the naïve entropy estimator and unique state-action count is given in Sec.A.2 in the Appendix. The unique state-action pairs are normalized with the unique state-action pairs of a *reference dataset* $\mathcal{D}_{\text{ref}}$ of the same MDP and the same dataset size. Hence our empirical measure SACo is defined as

$$SACo(\mathcal{D}) \coloneqq \frac{u_{s,a}(\mathcal{D})}{u_{s,a}(\mathcal{D}_{\text{ref}})}. \tag{6}$$

We use the replay dataset as the reference dataset $\mathcal{D}_{\text{ref}}$ in our experiments, since it was collected throughout training of the online policy and is assumed to have the most diverse set of state-action pairs. Counting unique state-action pairs of large datasets is often infeasible due to time and memory restrictions, or if datasets are distributed across different machines. We use HyperLogLog (Flajolet et al., 2007) as a probabilistic counting method to facilitate the applicability of SACo to large-scale benchmarks (see Sec. A.9.4 in the Appendix for details).

TQ. Second, we implement an empirical measure that estimates the expected trajectory return $g_{\mathcal{T}}$ given by Eq. 4 , which is a measure of how exploitative the behavioral policy is. The expected trajectory return $g_{\mathcal{T}}$ is estimated by the average return of the trajectories in the dataset $\bar{g}(\mathcal{D}) = \frac{1}{B}\sum_{b=0}^{B}\sum_{t=0}^{T_b}\gamma^t r_{b,t}$ for B trajectories with their respective lengths T_b. To allow comparisons across MDPs, we normalize the average trajectory return with the best and the worst behavior observed in the same MDP, which therefore corresponds to the quality of the trajectories relative to those. We thus define our empirical measure TQ as the normalized average trajectory return:

$$TQ(\mathcal{D}) \coloneqq \frac{\bar{g}(\mathcal{D}) - \bar{g}(\mathcal{D}_{\text{min}})}{\bar{g}(\mathcal{D}_{\text{expert}}) - \bar{g}(\mathcal{D}_{\text{min}})}, \tag{7}$$

where $\mathcal{D}_{\text{min}}$ is a dataset collected by a minimal performant policy π_{min} and $\mathcal{D}_{\text{expert}}$ is a dataset collected by an expert policy π_{expert}. Throughout our experiments, we chose the dataset sampled from the best policy found during online training as $\mathcal{D}_{\text{expert}}$ and the dataset sampled from a random policy as $\mathcal{D}_{\text{min}}$.

3 Dataset Generation

In Offline RL, dataset generation is neither harmonized, nor thoroughly investigated (Riedmiller et al., 2021). It has been shown in Kumar et al. (2020) that the performance of Conservative Q-learning (CQL) improved by changing the behavioral policy of the underlying dataset. Similarly, in Gulcehre et al. (2021) exchanging an expert with a noisy expert behavioral policy changed the performance of the compared algorithms. Furthermore, there is no consensus of which behavioral policy is the most representative for generating datasets to compare the performance of algorithms. Kumar et al. (2020) claims that datasets generated by multiple different behavioral policies fit a real world setting best. Contrary to that, Fujimoto et al. (2019a), claim that a single behavioral policy is a better fit. Thus, there is an ambiguity in the Offline RL literature on what may be the correct data generation scheme to test Offline RL algorithms. We review which datasets are utilized in the literature to compare among algorithms, with the aim to represent the most prominent dataset creation schemes in our empirical evaluation.

Agarwal et al. (2020) test on a dataset which consists of all training samples seen during online training of a DQN agent. Fujimoto et al. (2019a) generate data using a trained policy with an exploration factor. Fujimoto et al. (2019b) evaluates on multiple datasets, which include a dataset consisting of all transitions a RL algorithm samples during online training and data generated using a trained policy. Gulcehre et al. (2021) uses the RL Unplugged dataset (Gulcehre et al., 2020), which consists of different datasets collected by various policies. Kumar et al. (2020) uses three datasets generated by using a random, expert and a mixture of expert and random policy, generated from multiple different policies. Wang et al. (2020) use datasets generated from a pre-trained model based on transfer learning from human experts.

3.1 Dataset Generation Schemes

We generate data in five different settings: 1) *random*, 2) *expert*, 3) *mixed* 4) *noisy* and 5) *replay*. These generation schemes are designed to systematically cover and extend prior settings from literature. For each of the datasets, we have collected a predefined number of samples by interacting with the respective environment (see Sec. 4). The number of samples in a dataset is determined by the number of environment interactions that are necessary to obtain expert policies through an Online RL algorithm. We use DQN (Mnih et al., 2013) as a baseline for the Online RL algorithm, which serves as an expert behavioral policy to create and collect samples, as described below. Details on how the online policy was trained are given in Sec. A.9.2 in the Appendix.

- **Random Dataset.** This dataset is sampled by a random behavioral policy. Such a dataset was used for evaluation in Kumar et al. (2020). It serves as a naïve baseline for data collection.
- **Expert Dataset.** We trained an online policy until convergence and sampled with the final greedy expert policy. Such datasets are used in Fu et al. (2021); Gulcehre et al. (2021); Kumar et al. (2020).
- **Mixed Dataset.** The mixed dataset is generated using a mixture of the random dataset (80%) and the expert dataset (20%). This is similar to Fu et al. (2021); Gulcehre et al. (2021), where they refer to such a dataset as *medium-expert*.
- **Noisy Dataset.** The noisy dataset is generated with an expert policy that selects the actions ϵ-greedy with $\epsilon = 0.2$. Creating a dataset from a fixed noisy policy is similar to the dataset creation process in Fujimoto et al. (2019a;b); Kumar et al. (2020); Gulcehre et al. (2021).
- **Replay Dataset.** This dataset is the collection of all samples generated by the online policy during training, thus, consists of data generated by multiple policies. Such a dataset was used in Agarwal et al. (2020); Fujimoto et al. (2019b).

3.2 Dataset Generation from a domain shift perspective

Generating multiple datasets in the same MDP from different policies can also be interpreted as a domain shift between the distributions of the policies. Therefore, we want to further analyze which domain shifts are possible when generating RL datasets. The interaction of a policy π in a certain MDP generates transitions (s, a, r, s'), with a probability $p_\pi(s, a, r, s)$. We define a domain shift as a change of the distribution $p_\pi(s, a, r, s)$ to $\breve{p}_\pi(s, a, r, s)$, analogous to the definition of domain shift in the supervised learning setting (Widmer and Kubat, 1996; Gama et al., 2014; Webb et al., 2018; Wouter, 2018; Kouw and Loog, 2019; Khetarpal et al., 2020; Adler et al., 2020). Four distinct sources of domain shift can be disentangled by applying the chain rule of conditional probability on the transition probability $p_\pi(s, a, r, s) = p(r \mid s, a, s')\, p(s' \mid s, a)\, p_\pi(a \mid s)\, \rho_\pi(s)$, where $\rho_\pi(s) = \sum_a \rho_\pi(s, a)$ is the state-occupancy. This separation is not unique, but yields the policy and the conditional probabilities used in the definition of the MDP. Note that we separated the dynamics $p(r, s' \mid s, a)$ into the reward-dynamics $p(r \mid s, a, s')$ and the state-dynamics $p(s' \mid s, a)$ to further disentangle possible reasons for a domain shift. We consider the following types of domain shifts in RL:

- **Reward-Dynamics shift:** $p(r \mid s, a, s')$ is changed to $\breve{p}(r \mid s, a, s')$, while the state-dynamics $p(s' \mid s, a)$, the policy $p_\pi(a \mid s)$ and the state-occupancy $\rho_\pi(s)$ stay the same. This changes the expected return under the policy, but its occupancy stays the same.
- **State-Dynamics shift:** $p(s' \mid s, a)$ is changed to $\breve{p}(s' \mid s, a)$, while the reward-dynamics $p(r \mid s, a, s')$ and the policy $p_\pi(a \mid s)$ stay the same. This changes the occupancy under the policy, as the same behavior will result in a different distribution of outcomes. Although not necessary, the expected return under the policy is likely to change under a different occupancy.
- **Policy shift:** $p_\pi(a \mid s)$ is changed to $\breve{p}_\pi(a \mid s)$ while the reward-dynamics $p(r \mid s, a, s')$ and the state-dynamics $p(s' \mid s, a)$ stay the same. In general, RL datasets are not sampled from the same behavioral policy. Therefore, this domain shift is inherent to RL datasets.

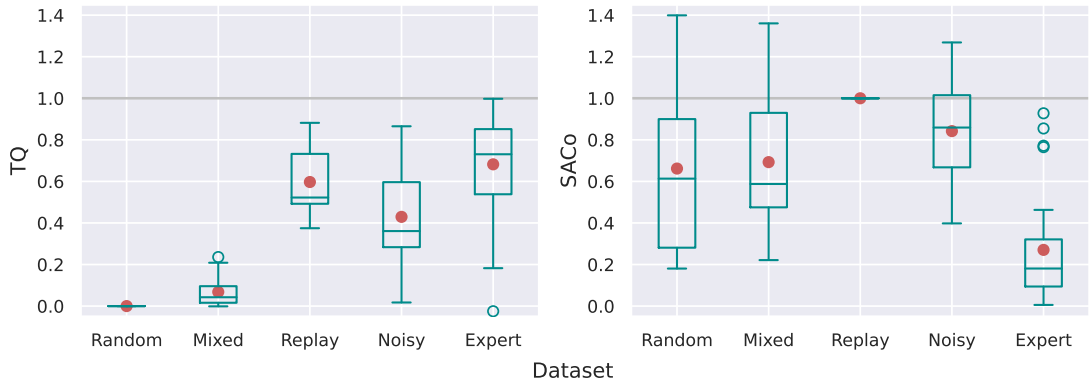


Figure 4: TQ and SACo over each dataset across dataset creation seeds and environments. Red dots represent mean values. The replay dataset exhibits a good balance between TQ and SACo, which is an explanation for the high performance across algorithms when using the replay dataset compared to other datasets.

The state-occupancy $\rho_\pi(s)$ depends on the state-dynamics $p(s' \mid s, a)$ and the policy $p_\pi(a \mid s)$ in a non-trivial way. This dependence is seen when formulating the state distribution recursively as $\rho_\pi(s') = \sum_s \rho_\pi(s) \sum_a p_\pi(a \mid s)\, p(s' \mid s, a)$. Note that a shift in the initial-state distribution or the absorbing-state distribution can also cause domain shifts of the state-occupancy. Every combination of the discussed shifts can occur as well, where we refer to the co-occurrence of all types of shifts as general domain shift. Note, that a state-dynamics and policy shift in general induce a change of the state probabilities, if their effects do not counterbalance each other. Finally, we conclude that generating datasets with different policies generally results in a shift of the dataset distribution, which we also observe empirically (see Sec. A.6 in the Appendix).

4 Experiments

We conducted our study on six different deterministic environments from multiple suites. These are two classic control environments from the OpenAI gym suite (Brockman et al., 2016), two MiniGrid (Chevalier-Boisvert et al., 2018) and two MinAtar environments (Young and Tian, 2019). For the first two suites, 10^5 samples were collected for every dataset, whereas $2 \cdot 10^6$ samples were collected for the MinAtar environments. Over all environments (six), different data generation schemes (five) and seeds (five), we generated a total number of 150 datasets.

We train nine different algorithms popular in the Offline RL literature including Behavioral Cloning (BC) (Pomerleau, 1991) and variants of DQN, Quantile-Regression DQN (QRDQN) (Dabney et al., 2017) and Random Ensemble Mixture (REM) (Agarwal et al., 2020). Furthermore, Behavior Value Estimation (BVE) (Gulcehre et al., 2021) and Monte-Carlo Estimation (MCE) are used. Finally, three widely popular Offline RL algorithms that constrain the learned policy to be near in distribution to the behavioral policy that created the dataset, Batch-Constrained Q-learning (BCQ) (Fujimoto et al., 2019b), Conservative Q-learning (CQL) (Kumar et al., 2019) and Critic Regularized Regression (CRR) (Wang et al., 2020) are considered. Details on specific implementations are given in Sec. A.8 in the Appendix.

The considered algorithms were executed on each of the 150 datasets for five different seeds. Details on online and offline training are given in Sec. A.9. We relate the performance of the best policies found during offline training to the best policy found during online training. The performance ω of the best policy found during offline training is given by $\omega(\mathcal{D}_{\text{offline}}) = (\bar{g}(\mathcal{D}_{\text{offline}}) - \bar{g}(\mathcal{D}_{\text{min}})) \,/\, (\bar{g}(\mathcal{D}_{\text{expert}}) - \bar{g}(\mathcal{D}_{\text{min}}))$. Policies are evaluated in the environment after fixed intervals during offline training. The trajectories sampled for the evaluation step yielding the highest average return represent the dataset $\mathcal{D}_{\text{offline}}$.

Results. We aim to analyze our experiments through the lens of dataset characteristics. Fig. 4 shows the TQ and the SACo of the gathered datasets, across dataset creation seeds and environments. Random and mixed datasets exhibit low TQ, while expert data has the highest TQ on average. In contrast, expert data exhibits low SACo on average, whereas random and mixed datasets attain higher values. The replay dataset provides a good balance between TQ and SACo. Fig. A.17 in the Appendix visualizes how generating the dataset influences the covered state-action space. In Fig. 5, we characterize the generated datasets using TQ and SACo. Each point represents a dataset, and each subplot contains the same datasets. The performance of the best policy found during offline training using a specific dataset is denoted by the color of the respective point.

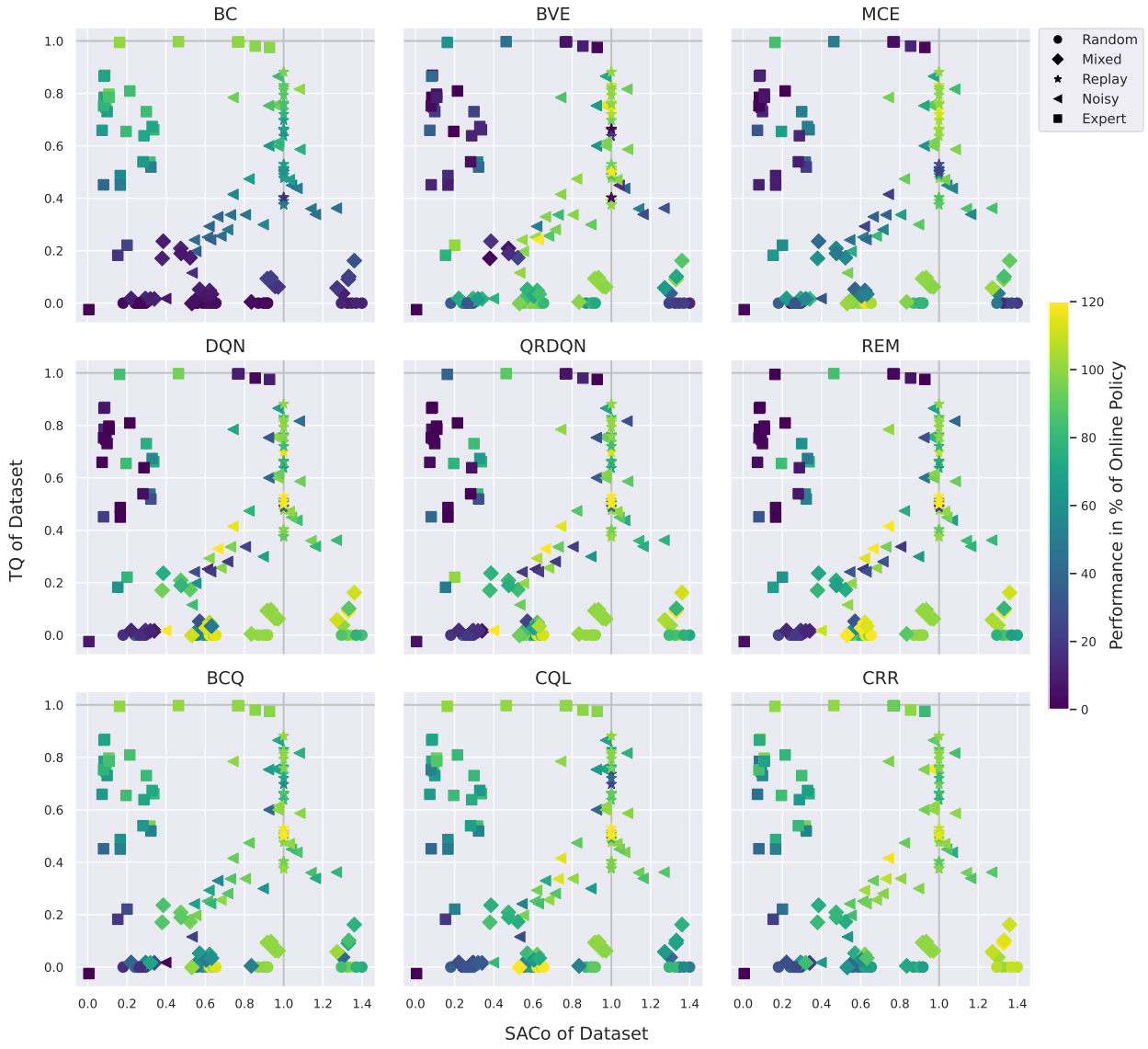


Figure 5: We characterize datasets using TQ and SACo, thus every point denotes one of the 150 datasets created for our evaluation. The position of points is thus the same in each subplot, as we trained every algorithm on every dataset. The performance of the best policy found during offline training on a dataset is denoted by the color of each individual point. We see: a) BC improves as TQ increases b) DQN variants (middle row) require high SACo to perform well c) Algorithms which constrain the learned policy towards the distribution of the behavioral policy (bottom row) perform well across datasets if they exhibit high TQ or SACo or both.

These results indicate that algorithms of the DQN family (DQN, QRDQN, REM) need datasets with high SACo to find a good policy. Furthermore, it was found that BC works well only if datasets have high TQ, which is expected as it imitates behavior observed in the dataset. BVE and MCE were found to be very sensitive to the specific environment and dataset setting, favoring datasets with high SACo. These algorithms are unconstrained in their final policy improvement step, but approximate the action-values of the behavioral policy instead of the optimal policy as in DQN. Thus, they may not leverage datasets with high SACo as well as algorithms from the DQN family, but encounter the same limitations for datasets with low SACo. BCQ, CQL and CRR enforce explicit or implicit constraints on the learned policy to be close to the distribution of the behavioral policy. These algorithms were found to outperform algorithms of the DQN family on average, but especially for datasets with low SACo and high TQ. Furthermore, algorithms with constraints towards the behavioral policy were found to perform well if datasets exhibit high TQ or SACo or moderate values of TQ and SACo.

We observe from Fig. 5, that algorithms that are closely related (e.g, DQN family, second row) perform similarly when trained on similar datasets in terms of TQ and SACo , which is an interesting property to investigate in future work. Scatterplots between pairs of TQ, SACo and the performance are given in Fig. A.13 and Fig. A.14 in the Appendix. All scores for all environments and algorithms over datasets are given in Sec. A.12 in the Appendix. We also analyzed these experiments with a logarithmic version of SACo (logarithmic in the nominator and denominator) in Sec. A.15 in the Appendix. Furthermore, we analyzed these experiments using naïve entropy estimators instead of SACo in Sec. A.16 in the Appendix. We found that the presented version with SACo yields the visually and conceptually simplest interpretation for our experiments, while all versions yield qualitatively similar results.

5 Discussion

Limitations. We conducted our main experiments on discrete-action environments and showed initial results on continuous-action environments in Sec. A.17 in the Appendix. Nevertheless, future work would need to investigate the effects of dataset characteristics for continuous action-spaces on a wider range of different tasks, algorithms and data collection schemes as done for discrete action-spaces. Furthermore, this study is limited to model-free algorithms, while model-based algorithms have recently been reported to achieve superior results in complex continuous control industrial benchmarks (Swazinna et al., 2022). For computational reasons, we limited this study to only use the off-policy algorithm DQN as an online policy. A comparison to an on-policy method as behavioral policy would be of interest, as well as exploration algorithms to generate even more diverse datasets; e.g. using generative flow networks (Bengio et al., 2021a;b). Counting unique state-action pairs as an empirical exploration measure is simple and interpretable, but explicit entropy estimation would be especially beneficial on continuous state and action spaces. Lastly, although we did define different types of domain shifts, the definition and analysis of an appropriate domain shift measure is beyond the scope of this work.

Conclusion. In this study, we derived two theoretical measures, the transition-entropy corresponding to exploratiVeness of a policy and the expected trajectory return, corresponding to the exploitativeness of a policy. Furthermore, we analyzed stability traits of these measures under MDP homomorphisms. Moreover, we implemented two empirical measures, TQ and SACo, which correspond to the expected trajectory return and the transition-entropy in deterministic MDPs. We generated 150 datasets using six environments and five dataset sampling strategies over five seeds. On these datasets, the performance of nine model-free algorithms was evaluated over independent runs, resulting in 6750 trained offline policies. The performance of the offline algorithms shows clear correlations with the exploratory and exploitative characteristics of the dataset, measured by TQ and SACo. We found, that popular algorithms in Offline RL are strongly influenced by the characteristics of the dataset and the average performance across different datasets might not be enough for a fair comparison. Our study thus provides a blueprint to characterize Offline RL datasets and understanding their effect on algorithms.

Acknowledgements The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. IARAI is supported by Here Technologies. We thank the projects AI-MOTION (LIT-2018-6-YOU-212), AI-SNN (LIT-2018-6-YOU-214), DeepFlood (LIT-2019-8-YOU-213), Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for GranularFlow (FFG-871302), AIRI FG 9-N (FWF-36284, FWF-36235), ELISE (H2020-ICT-2019-3 ID: 951847). We thank Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, Software Competence Center Hagenberg GmbH, TÜV Austria, Frauscher Sensonic, AI Austria Reinforcement Learning Community, and the NVIDIA Corporation.

References

D. Abel. A theory of state abstraction for reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9876–9877, Jul. 2019. doi: 10.1609/aaai.v33i01.33019876.

D. Abel. A theory of abstraction in reinforcement learning. *arXiv preprint arXiv:2203.00397*, 2022.

T. Adler, J. Brandstetter, M. Widrich, A. Mayr, D. Kreil, M. Kopp, G. Klambauer, and S. Hochreiter. Cross-domain few-shot learning by representation fusion. *arXiv preprint arXiv:2010.06498*, 2020.

R. Agarwal, D. Schuurmans, and M. Norouzi. An optimistic perspective on offline reinforcement learning. *arXiv preprint arXiv:1907.04543*, 2020.

J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter. RUDDER: return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems 32*, pages 13566–13577, 2019.

G. P. Basharin. On a statistical estimate for the entropy of a sequence of independent random variables. *Theory of Probability & Its Applications*, 4(3):333–336, 1959.

E. Bengio, M. Jain, M. Korablyov, D. Precup, and Y. Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34, 2021a.

Y. Bengio, T. Deleu, E. Hu, S. Lahlou, M. Tiwari, and E. Bengio. Gflownet foundations. *arXiv preprint arXiv:2111.09266*, 2021b.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. E. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerík, O. Sushkov, D. Barker, J. Scholz, M. Denil, N. de Freitas, and Z. Wang. A framework for data-driven robotics. *arXiv preprint arXiv:abs/1909.12200*, 2019.

M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic gridworld environment for openai gym. *GitHub repository*, 2018.

D. Cohn. Neural network exploration using optimal experiment design. *Advances in neural information processing systems*, 6, 1993.

W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. *arXiv preprint arXiv:1710.10044*, 2017.

S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2020.

M.-C. Dinu, M. Hofmarcher, V. P. Patil, M. Dorfer, P. M. Blies, J. Brandstetter, J. A. Arjona-Medina, and S. Hochreiter. *XAI and Strategy Extraction via Reward Redistribution*, pages 177–205. Springer International Publishing, Cham, 2022. ISBN 978-3-031-04083-2. doi: 10.1007/978-3-031-04083-2_10.

G. Dulac-Arnold, D. J. Mankowitz, and T. Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In *in aofa '07: proceedings of the 2007 international conference on analysis of algorithms*, 2007.

J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2021.

S. Fujimoto and S.S. Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint arXiv:1910.01708*, 2019a.

S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2019b.

J. Gama, I. Zliobaite, A. Bifet, P. Mykola, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), 2014. ISSN 0360-0300. doi: 10.1145/2523813.

R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1):163–223, 2003. ISSN 0004-3702. doi: 10.1016/S0004-3702(02)00376-4.

C. Gulcehre, Z. Wang, A. Novikov, T. Le Paine, S. G. Colmenarejo, K. Zolna, R. Agarwal, J. Merel, D. Mankowitz, C. Paduraru, G. Dulac-Arnold, J. Li, M. Norouzi, M. Hoffman, O. Nachum, G. Tucker, N. Heess, and N. de Freitas. RL unplugged: Benchmarks for offline reinforcement learning. *arXiv preprint arXiv:2006.13888*, 2020.

C. Gulcehre, S. Gómez Colmenarejo, Z. Wang, J. Sygnowski, T. Paine, K. Zolna, Y. Chen, M. Hoffman, R. Pascanu, and N. de Freitas. Regularized behavior value estimation. *arXiv preprint arXiv:2103.09575*, 2021.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. Dy and A. Krause, editors, *Proceedings of Machine Learning Research*, volume 80, pages 1861–1870. PMLR, 2018. arXiv 1801.01290.

B. Harris. The statistical estimation of entropy in the non-parametric case. Technical report, Wisconsin Univ-Madison Mathematics Research Center, 1975.

J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems 29*, pages 4565–4573, 2016.

M. Holzleitner, L. Gruber, J. A. Arjona-Medina, J. Brandstetter, and S. Hochreiter. Convergence proof for actor-critic methods applied to PPO and RUDDER. *arXiv preprint arXiv:2012.01399*, 2020.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

N. K. Jong and P. Stone. State abstraction discovery from irrelevant state variables. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, page 752–757, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

K. Khetarpal, M. Riemer, I. Rish, and D. Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.

G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks, 2017.

I. Kostrikov, R. Fergus, J. Tompson, and O. Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021.

W. M. Kouw and M. Loog. A review of domain adaptation without target labels. *IEEE transactions on pattern analysis and machine intelligence*, October 2019. ISSN 0162-8828. doi: 10.1109/tpami.2019.2945942.

A. Kumar, J. Fu, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

S. Lange, T. Gabel, and M. Riedmiller. *Batch Reinforcement Learning*, pages 45–73. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27645-3. doi: 10.1007/978-3-642-27645-3_2.

S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

L. Li, T. J. Walsh, and M. L. Littman. Towards a unified theory of state abstraction for MDPs. In *In Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, pages 531–539, 2006.

R. McFarlane. A survey of exploration strategies in reinforcement learning. *McGill University*, 2003.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

L. Monier, J. Kmec, A. Laterre, T. Pierrot, V. Courgeau, O. Sigaud, and K. Beguir. Offline reinforcement learning hands-on. *arXiv preprint arXiv:2011.14379*, 2020.

G. Neu and C. Pike-Burke. A unifying view of optimism in episodic reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1392–1403, 2020.

J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with cuda. *ACM Queue*, (2):40–53, 4 2008.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

V. P. Patil, M. Hofmarcher, M.-C. Dinu, M. Dorfer, P. M. Blies, J. Brandstetter, J. A. Arjona-Medina, and S. Hochreiter. Align-RUDDER: Learning from few demonstrations by reward redistribution. *arXiv preprint arXiv:2009.14108*, 2020.

D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Comput.*, 3(1): 88–97, 1991. ISSN 0899-7667.

K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11157–11166, 2020.

B. Ravindran and A. G. Barto. Symmetries and model minimization in markov decision processes. Technical report, USA, 2001.

M. A. Riedmiller, J. T. Springenberg, R. Hafner, and N. Heess. Collect & infer - a fresh look at data-efficient reinforcement learning. *arXiv preprint arXiv:2108.10273*, 2021.

G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2:230–247, 2010.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948. doi: https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

J. Storck, S. Hochreiter, and J. Schmidhuber. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the International Conference on Artificial Neural Networks, Paris*, volume 2, pages 159–164. EC2 & Cie, Paris, 1995.

R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Dept. of Comp. and Inf. Sci., 1984.

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999. ISSN 0004-3702. doi: 10.1016/S0004-3702(99)00052-1.

P. Swazinna, S. Udluft, D. Hein, and T. Runkler. Comparing model-free and model-based algorithms for offline reinforcement learning. *arXiv preprint arXiv:2201.05433*, 2022.

E. van der Pol, D. Worrall, H. van Hoof, F. Oliehoek, and M. Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4199–4210. Curran Associates, Inc., 2020.

Z. Wang, A. Novikov, K. Zolna, J. T. Springenberg, S. Reed, B. Shahriari, N. Siegel, J. Merel, C. Gulcehre, N. Heess, and N. de Freitas. Critic regularized regression. *arXiv preprint arXiv:2006.15134*, 2020.

G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean. Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery*, 32:1179–1199, 2018.

G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1): 69–101, 1996.

M. Widrich, M. Hofmarcher, V. P. Patil, A. Bitto-Nemling, and S. Hochreiter. Modern Hopfield Networks for Return Decomposition for Delayed Rewards. In *Deep RL Workshop NeurIPS 2021*, 2021.

M. Wiering and J. Schmidhuber. Efficient model-based exploration. pages 223–228. MIT Press, 1998.

M. K. Wouter. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*, 2018.

Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. arXiv, 2019.

K. Young and T. Tian. Minatar: An Atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. *arXiv preprint arXiv:1805.04687*, 2020.

A APPENDIX

CONTENTS OF THE APPENDIX

A.1 EXPLORATIVENESS OF POLICIES IN STOCHASTIC AND DETERMINISTIC MDPS

In the following, we introduce a measure to quantify how well a given policy π explores a given MDP on expectation.

A.1.1 EXPLORATIVENESS AS EXPECTED INFORMATION

We consider an MDP with finite state, action and reward spaces throughout the following analysis. We define *explorativeness* of a policy π as the expected information of the interaction between the policy and a given MDP. In information theory, the expected information content of a measurement of a random variable X is defined as its Shannon entropy, given by $H(X) \coloneqq -\sum_i p(x_i) \log(p(x_i))$ (Shannon, 1948). Corresponding to this, we define the expected information about policy MDP interactions through the transitions (s, a, r, s'), observed according to the transition probability $p_\pi(s, a, r, s')$ by a policy π interacting with the environment. We want to explicitly stress the interconnection of the policy and the MDP as a single transition generating process. Without a given MDP, explorativeness of a policy can not be defined in this framework. The expected information is thus given by the transition-entropy

$$H(p_\pi(s, a, r, s')) \coloneqq - \sum_{\substack{s,a,r,s' \\ p_\pi(s,a,r,s')>0}} p_\pi(s, a, r, s') \log(p_\pi(s, a, r, s')). \tag{8}$$

To illustrate how $p_\pi(s, a, r, s')$ is influenced by the policy and the MDP dynamics, we rewrite the transition probability as $p_\pi(s, a, r, s') = p(r, s' \mid s, a)\ p_\pi(s, a)$. The dynamics $p(r, s' \mid s, a)$ depend solely on the MDP, while $p_\pi(s, a)$ is steered by the policy π. The state-action probability $p_\pi(s, a)$ is often referred to as the occupancy measure $\rho^h_\pi(s, a) = \mathbb{P}_\pi[s_h = s, a_h = a]$ (Neu and Pike-Burke, 2020) induced by the policy π, where h denotes the stage to specify changes of the dynamics. In the following, we only consider episodes consisting of a single stage and drop the index h accordingly, but the following derivations would extend to episodes with multiple stages. Consequently, $\rho_\pi(s, a)$ is used instead of $p_\pi(s, a)$ to emphasize that the state-action probability under a policy is the occupancy under this policy.

The occupancy depends not only on the policy alone, but also on the MDP dynamics. To illustrate this fact, consider that the occupancy of a policy can be rewritten as $\rho_\pi(s, a) = p_\pi(a \mid s)\ p_\pi(s)$, thus the probability that a policy selects an action given a state $p_\pi(a \mid s)$, times the probability of being in a certain state under that policy $p_\pi(s)$. The probability of being in a certain state can be recursively defined as $p_\pi(s') = \sum_s p_\pi(s) \sum_a p_\pi(a \mid s)\ p(s' \mid s, a)$ via the MDP state-dynamics $p(s' \mid s, a)$.

Using the above, the transition-entropy can be further decomposed:

$$\begin{aligned}
&H(p_\pi(s, a, r, s')) \\
&\quad = - \sum_{\substack{s,a,r,s' \\ p_\pi(s,a,r,s')>0}} p_\pi(s, a, r, s') \log(p_\pi(s, a, r, s')) \\
&\quad = - \sum_{\substack{s,a,r,s' \\ p_\pi(s,a,r,s')>0}} p(r, s' \mid s, a)\ \rho_\pi(s, a) \log(p(r, s' \mid s, a)\ \rho_\pi(s, a)) \\
&\quad = - \sum_{\substack{s,a,r,s' \\ p_\pi(s,a,r,s')>0}} \rho_\pi(s, a)\ p(r, s' \mid s, a) \log(p(r, s' \mid s, a)) - \sum_{\substack{s,a,r,s' \\ \rho_\pi(s,a)>0}} p(r, s' \mid s, a)\ \rho_\pi(s, a) \log(\rho_\pi(s, a)) \\
&\quad = -\sum_{s,a} \rho_\pi(s, a) \sum_{\substack{r,s' \\ p_\pi(s,a,r,s')>0}} p(r, s' \mid s, a)\ \log(p(r, s' \mid s, a)) - \sum_{\substack{s,a,r,s' \\ \rho_\pi(s,a)>0}} p(r, s' \mid s, a)\ \rho_\pi(s, a) \log(\rho_\pi(s, a)) \\
&\quad = \sum_{s,a} \rho_\pi(s, a)\ H(p(r, s' \mid s, a)) - \sum_{\substack{s,a \\ \rho_\pi(s,a)>0}} \rho_\pi(s, a) \log(\rho_\pi(s, a)) \sum_{r,s'} p(r, s' \mid s, a) \\
&\quad = \sum_{s,a} \rho_\pi(s, a)\ H(p(r, s' \mid s, a)) - \sum_{\substack{s,a \\ \rho_\pi(s,a)>0}} \rho_\pi(s, a) \log(\rho_\pi(s, a)) \\
&\quad = \sum_{s,a} \rho_\pi(s, a)\ H(p(r, s' \mid s, a)) + H(\rho_\pi(s, a)).
\end{aligned} \tag{9}$$

The transition-entropy thus equals the occupancy weighted sum of dynamics-entropies $H(p(r, s' \mid s, a))$ under every visitable state-action pair plus the occupancy-entropy $H(\rho_\pi(s, a))$ under the policy.

To maximize the transition-entropy, thus explore more on expectation, a tradeoff between two options is encountered, assuming a fixed visited state-action support under any candidate policy. First, the occupancy can be distributed such that state-action pairs where transitions are more stochastic, and thus having high dynamics-entropy, are visited more often. Second, the occupancy can be distributed evenly among all state-action pairs to have a high occupancy-entropy. However, the transition-entropy is not straightforward to optimize without further assumptions, as there is a strong interplay between the policy and the MDP dynamics, which does not allow for smooth changes in the occupancy.

Furthermore, the transition-entropy generally increases if more state-action pairs are visitable under a policy. Note that this is only strictly true, if additional assumptions on the distribution of MDP dynamics and occupancies under two policies that are compared are introduced. This holds for instance for the upper bound on the entropy, where possible outcomes follow a uniform distribution. The entropy of a random variable X that follows a uniform distribution is $H(X) = \log(N)$, which grows logarithmically by the number of possible outcomes N.

To summarize, the interaction of a policy in an MDP induces high transition-entropy if a large state-action support is occupied, more stochastic transitions are visited more frequently and all other possible transitions are visited evenly.

A.1.2 Deterministic MDPs

Deterministic MDPs differ in the sense that all information about the dynamics of a specific transition $p(r, s' \mid s, a)$ is obtained the first time this transition is observed. Note that for any deterministic MDP, there exists a function $\mathbb{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{R} \times \mathcal{S}; (s, a) \mapsto (r, s')$ that maps state-action pairs to a reward and next state. The dynamics of a deterministic MDP can thus be written as

$$p_{\text{det}}(r, s' \mid s, a) \coloneqq \begin{cases} 1 & \text{if } (r, s') = \mathbb{T}(s, a) \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Starting from the general result of Eq. 9 and inserting the definition of deterministic dynamics from Eq. 10, the expected information about the dynamics in a deterministic MDP is given by

$$\begin{aligned} H(p_\pi(s, a, r, s')) &= \sum_{s,a} \rho_\pi(s, a)\ H(p_{\text{det}}(r, s' \mid s, a)) + H(\rho_\pi(s, a)) \\ &= -\sum_{s,a} \rho_\pi(s, a) \sum_{\substack{r,s' \\ (r,s')=\mathbb{T}(s,a)}} p_{\text{det}}(r, s' \mid s, a) \log(p_{\text{det}}(r, s' \mid s, a)) + H(\rho_\pi(s, a)) \\ &= -\sum_{s,a} \rho_\pi(s, a)\ 1 \log(1) + H(\rho_\pi(s, a)) \\ &= H(\rho_\pi(s, a)). \end{aligned} \tag{11}$$

The transition-entropy thus reduces to the occupancy-entropy under the policy for deterministic dynamics. Therefore, the more uniform the policy visits the state-action space, the higher the transition-entropy and thus the more explorative the policy is on expectation. Additionally, having a large state-action support is still beneficial to increase the transition-entropy as it is for stochastic MDPs.

A.2 BIAS OF ENTROPY ESTIMATORS

We discussed two possible estimators for the occupancy-entropy, which we want to characterize further in this section. The first is the naïve entropy estimator for a dataset $\hat{H}(\mathcal{D}) = -\sum_{i=1}^{K} \hat{p}_i \log(\hat{p}_i)$ with $\hat{p}_i = n_{s,a}/N$, where $n_{s,a}$ is the count of a specific state-action pairs (s, a) in the dataset of size N. The second is the upper bound of the naïve entropy estimator, $\log(u_{s,a}(\mathcal{D}))$, which is a non-consistent estimator of $H(\rho_\pi)$. We start with from the bias analysis of the naïve entropy estimator $\hat{H}(\mathcal{D})$ given by Basharin (1959) and extended to the second order by Harris (1975):

$$H(\rho_\pi) - \mathbb{E}[\hat{H}(\mathcal{D})] = \frac{K-1}{2N} + \frac{1}{12N^2}\left(\sum_{k}^{K} \frac{1}{p_k} - 1\right) + \mathcal{O}(N^{-3}) \tag{12}$$

where K denotes the number of visitable state-action pairs under the policy, thus $p_k = \rho_\pi(s, a)$ for the k-th visitable state-action pair. For brevity, let

$$Z := \frac{K-1}{2N} + \frac{1}{12N^2}\left(\sum_{k}^{K} \frac{1}{p_k} - 1\right) + \mathcal{O}(N^{-3}) \tag{13}$$

We can rewrite Eq. 13 to arrive at an expression of the bias of $\log(u_{s,a}(\mathcal{D}))$:

$$\begin{aligned} H(\rho_\pi) - \mathbb{E}[\hat{H}(\mathcal{D})] &= Z \\ H(\rho_\pi) &= Z + \mathbb{E}[\hat{H}(\mathcal{D})] \\ H(\rho_\pi) - \mathbb{E}[\log(u_{s,a}(\mathcal{D}))] &= Z + \mathbb{E}[\hat{H}(\mathcal{D})] - \mathbb{E}[\log(u_{s,a}(\mathcal{D}))] \end{aligned} \tag{14}$$

We use the fact that:

$$0 \geq \mathbb{E}[\hat{H}(\mathcal{D})] - \mathbb{E}[\log(u_{s,a}(\mathcal{D}))] \geq -\log(N) \tag{15}$$

and arrive at:

$$Z \geq H(\rho_\pi) - \mathbb{E}[\log(u_{s,a}(\mathcal{D}))] \geq Z - \log(N) \geq \frac{K-1}{2N} - \log(N) \tag{16}$$

The bias of $\log(u_{s,a}(\mathcal{D}))$ is thus smaller than the naïve entropy estimator $\hat{H}(\mathcal{D})$ as long as $\frac{K-1}{2N} - \log(N) \geq 0$. This bias will not become negative, which would result in $\log(u_{s,a}(\mathcal{D}))$ overestimating $H(\rho_\pi)$, if the following holds:

$$\begin{aligned} \frac{K-1}{2N} - \log(N) &\geq 0 \\ K &\geq 2N\log(N) + 1. \end{aligned} \tag{17}$$

Therefore, as long as $K \geq 2N\log(N) + 1$, the estimator $\log(u_{s,a}(\mathcal{D}))$ is equally or less biased than the naïve entropy estimator $\hat{H}(\mathcal{D})$.

A.3 AMDP DEFINITION

In the following paragraphs, we extend the details for Definition 1:

Definition. *Given two MDPs $M = (\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$ and $\tilde{M} = (\tilde{\mathcal{S}}, \tilde{\mathcal{A}}, \tilde{\mathcal{R}}, \tilde{p}, \gamma)$, we assume there exists a common abstract MDP (AMDP) $\hat{M} = (\hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{R}}, \hat{p}, \gamma)$ (see Fig. 3), whereas M and $\tilde{M}$ are homomorphic images of $\hat{M}$. We base the definition of the AMDP on prior work from Sutton et al. (1999); Jong and Stone (2005); Li et al. (2006); Abel (2019); van der Pol et al. (2020); Abel (2022) by considering state and action abstractions. We define an MDP homomorphism by the surjective abstraction functions as $\phi : \mathcal{S} \to \hat{\mathcal{S}}$ and $\tilde{\phi} : \tilde{\mathcal{S}} \to \hat{\mathcal{S}}$, with $\phi(s), \tilde{\phi}(\tilde{s}) \in \hat{\mathcal{S}}$ for the state abstractions and $\{\psi_s : \mathcal{A} \to \hat{\mathcal{A}} \mid s \in \mathcal{S}\}$ and $\{\tilde{\psi}_{\tilde{s}} : \tilde{\mathcal{A}} \to \hat{\mathcal{A}} \mid \tilde{s} \in \tilde{\mathcal{S}}\}$, with $\psi_s(a), \tilde{\psi}_{\tilde{s}}(\tilde{a}) \in \hat{\mathcal{A}}$ for the action abstractions.*

The inverse images $\phi^{-1}(\hat{s})$ with $\hat{s} \in \hat{\mathcal{S}}$, and $\psi_s^{-1}(\hat{a})$ with $\hat{a} \in \hat{\mathcal{A}}$, are the set of ground states and actions that correspond to $\hat{s}$ and $\hat{a}$, under abstraction function ϕ and ψ_s respectively. Under these assumptions, $\{\phi^{-1}(\hat{s}) \mid \hat{s} \in \hat{\mathcal{S}}\}$ and $\{\psi_s^{-1}(\hat{a}) \mid \hat{a} \in \hat{\mathcal{A}}\}$ partition the ground state $\mathcal{S}$ and ground action $\mathcal{A}$. The inverse mappings hold equivalently for $\tilde{\phi}^{-1}(\hat{s})$, $\tilde{\psi}_{\tilde{s}}^{-1}(\hat{s})$.

The mappings are built to satisfy the following conditions:

$$\hat{p}(r \mid \phi(s), \psi_s(a), \phi(s')) \triangleq p(r \mid s, a, s') \qquad \forall s, s' \in \mathcal{S}, a \in \mathcal{A} \tag{18}$$

$$\hat{p}(r \mid \tilde{\phi}(\tilde{s}), \tilde{\psi}_{\tilde{s}}(\tilde{a}), \tilde{\phi}(\tilde{s}')) \triangleq \tilde{p}(r \mid \tilde{s}, \tilde{a}, \tilde{s}') \qquad \forall \tilde{s}, \tilde{s}' \in \tilde{\mathcal{S}}, \tilde{a} \in \tilde{\mathcal{A}} \tag{19}$$

$$\hat{p}(\phi(s') \mid \phi(s), \psi_s(a)) \triangleq \sum_{\bar{s}' \in \phi^{-1}(\phi(s'))} p(\bar{s}' \mid s, a) \qquad \forall s, s' \in \mathcal{S}, a \in \mathcal{A} \tag{20}$$

$$\hat{p}(\tilde{\phi}(\tilde{s}') \mid \tilde{\phi}(\tilde{s}), \tilde{\psi}_{\tilde{s}}(\tilde{a})) \triangleq \sum_{\bar{s}' \in \tilde{\phi}^{-1}(\tilde{\phi}(\tilde{s}'))} \tilde{p}(\bar{s}' \mid \tilde{s}, \tilde{a}) \qquad \forall \tilde{s}, \tilde{s}' \in \tilde{\mathcal{S}}, \tilde{a} \in \tilde{\mathcal{A}} \tag{21}$$

Note that ϕ and ψ_s are surjective functions, therefore expressions such as $\phi^{-1}(\phi(s'))$ return a set.

A.4 BOUND TRANSITION-ENTROPIES OF HOMOMORPHIC IMAGES

In the following, we give the missing proof of Theorem 1 in Sec. 2.3.

Lemma 1. *Let $\pi(a \mid s)$ be a policy of a homomorphic image M of an AMDP $\hat{M}$ with corresponding abstract policy $\hat{\pi}(\hat{a} \mid \hat{s})$. Let $p(s, a, r, s')$ and $\hat{p}(s, a, r, s')$ be the transition probabilities induced by $\pi(a \mid s)$ and $\hat{\pi}(\hat{a} \mid \hat{s})$ respectively. The transition-entropy of the common AMDP $H(\hat{p}(\hat{s}, \hat{a}, r, \hat{s}'))$ provides a lower-bound to $H(p(s, a, r, s'))$.*

Proof. We start with an intuition. The space states, actions and next-states of M can be partitioned in a way, such that each partition maps to a single state-action-next-state tuple $(\hat{s}, \hat{a}, \hat{s}')$ in $\hat{M}$. This is illustrated in Fig. 3. For this mapping, the probability mass is conserved, formally $\hat{p}(\hat{s}, \hat{a}, \hat{s}') = \sum_{s \in \phi^{-1}(\hat{s}), a \in \psi_s^{-1}(\hat{a}), s' \in \phi^{-1}(\hat{s}')} p(s, a, s')$. By these mappings from $\hat{M}$ to M, the entropy can only be increased or be equal.

Here is a formal proof. We start with the definitions of the entropies.

$$H(\hat{p}(s, a, r, s')) = -\sum_{\hat{s}, \hat{a}, r, \hat{s}'} p(\hat{s}, \hat{a}, r, \hat{s}') \cdot \log(\hat{p}(\hat{s}, \hat{a}, r, \hat{s}')) \tag{22}$$

$$H(p(s, a, r, s')) = -\sum_{s, a, r, s'} p(s, a, r, s') \cdot \log(p(s, a, r, s')) \tag{23}$$

First we look at the function $f(x) = -x \log(x)$ for $x > 0$. We can show that $f(x)$ is concave by showing that its second derivative is non-positive.

$$f'(x) = -\log(x) - 1 \tag{24}$$

$$f''(x) = -\frac{1}{x} \tag{25}$$

Since $f(x) > 0$ for $x > 0$ and $f(x)$ is concave, it follows that $f(x)$ is *sub-additive* meaning that for every $x_1, x_2 > 0$ that $f(x_1) + f(x_2) \geq f(x_1 + x_2)$. More generally, we can say that for any $x_i > 0$ that $\sum_i f(x_i) \geq f(\sum_i x_i)$. From the assumptions of the MDP homomorphism it follows that the probability mass of an abstract state-action-next-state 3-tuple is equal to the sum of probability masses of the state-action-next-state 3-tuples in its inverse images.

$$\hat{p}(\hat{s}, \hat{a}, r, \hat{s}') = \sum_{\substack{s \in \phi^{-1}(\hat{s}) \\ a \in \psi_s^{-1}(\hat{a}) \\ s' \in \phi^{-1}(\hat{s}')}} p(s, a, r, s') \tag{26}$$

We fix an arbitrary abstract transition $(\hat{s}, \hat{a}, r, \hat{s}')$. From sub-additivity of $f(x)$ follows that

$$\sum_{\substack{s \in \phi^{-1}(\hat{s}) \\ a \in \psi_s^{-1}(\hat{a}) \\ s' \in \phi^{-1}(\hat{s}')}} f(p(s, a, r, s')) \geq f(\hat{p}(\hat{s}, \hat{a}, r, \hat{s}')) \tag{27}$$

$$-\sum_{\substack{s \in \phi^{-1}(\hat{s}) \\ a \in \psi_s^{-1}(\hat{a}) \\ s' \in \phi^{-1}(\hat{s}')}} p(s, a, r, s') \cdot \log(p(s, a, r, s')) \geq -\hat{p}(\hat{s}, \hat{a}, r, \hat{s}') \cdot \log(\hat{p}(\hat{s}, \hat{a}, r, \hat{s}')) \tag{28}$$

As Eq. 28 holds for *every* abstract transition, the inequality also holds for the sum over all transitions:

$$-\sum_{\hat{s}, \hat{a}, r, \hat{s}'} \sum_{\substack{s \in \phi^{-1}(\hat{s}) \\ a \in \psi_s^{-1}(\hat{a}) \\ s' \in \phi^{-1}(\hat{s}')}} p(s, a, r, s') \cdot \log(p(s, a, r, s')) \geq -\sum_{\hat{s}, \hat{a}, r, \hat{s}'} p(\hat{s}, \hat{a}, r, \hat{s}') \cdot \log(p(\hat{s}, \hat{a}, r, \hat{s}')) \tag{29}$$

These are the transition-entropy for the abstract MDP (Eq. 22) and the MDP that is an homomorphic image of the abstract MDP (Eq. 23). Therefore Eq. 29 results in

$$H(p(s, a, r, s')) \geq H(p(\hat{s}, \hat{a}, r, \hat{s}')). \tag{30}$$

This concludes the proof. □

A.5 Analyzing AMDPs

This section provides an illustrative overview of the properties of abstract MDPs (AMDPs).

A.5.1 Common AMDP Transformations

This section provides an illustrative overview on the core properties of AMDPs.

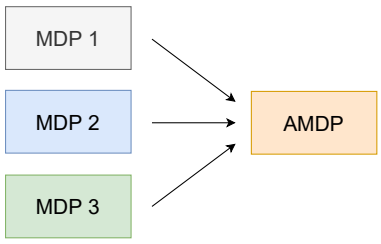


Figure A.1: Schematic depiction of three homomorphic images of an abstract MDP (AMDP).

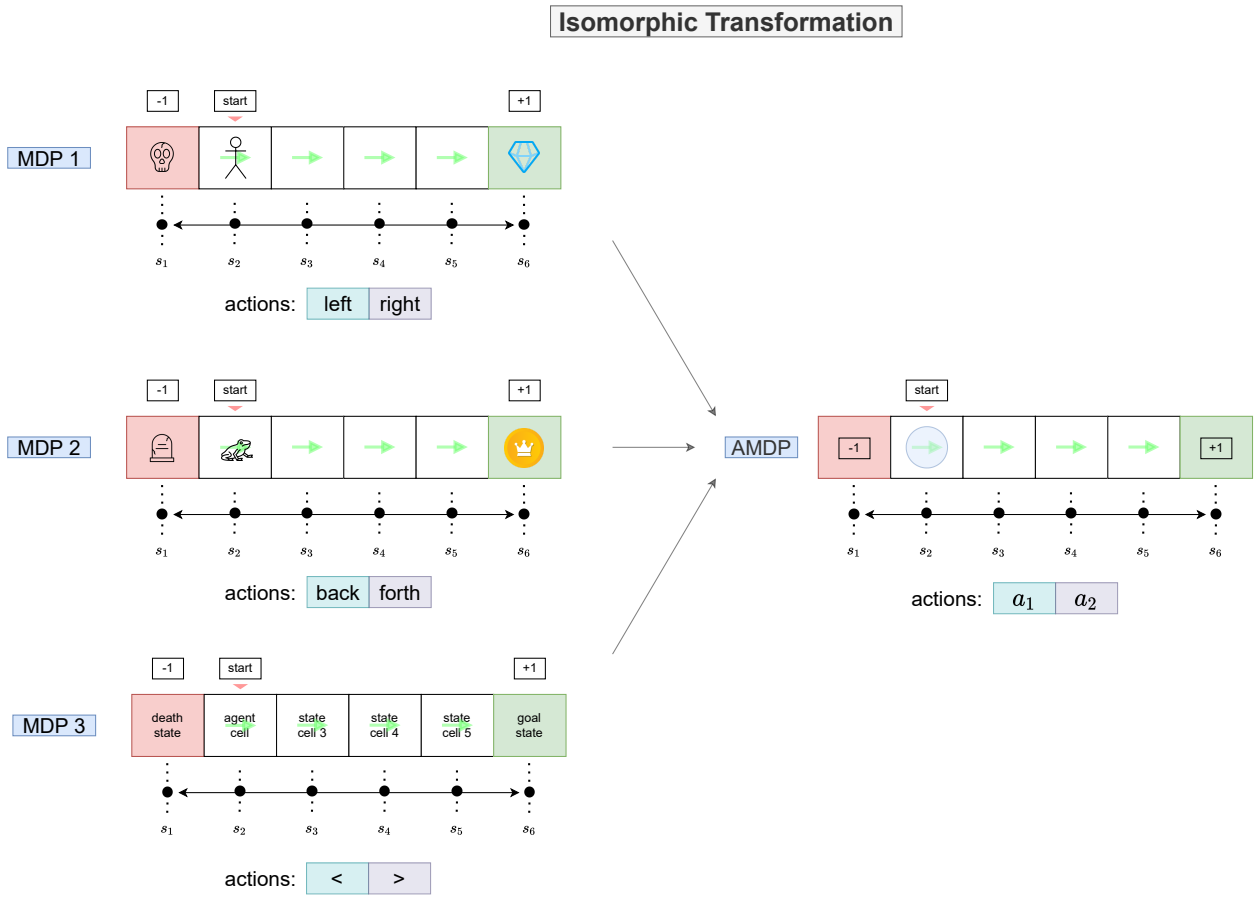


Figure A.2: An example of three homomorphic images of an abstract MDP (AMDP). MDP 1 to 3 are distinct in their state and/or action space.

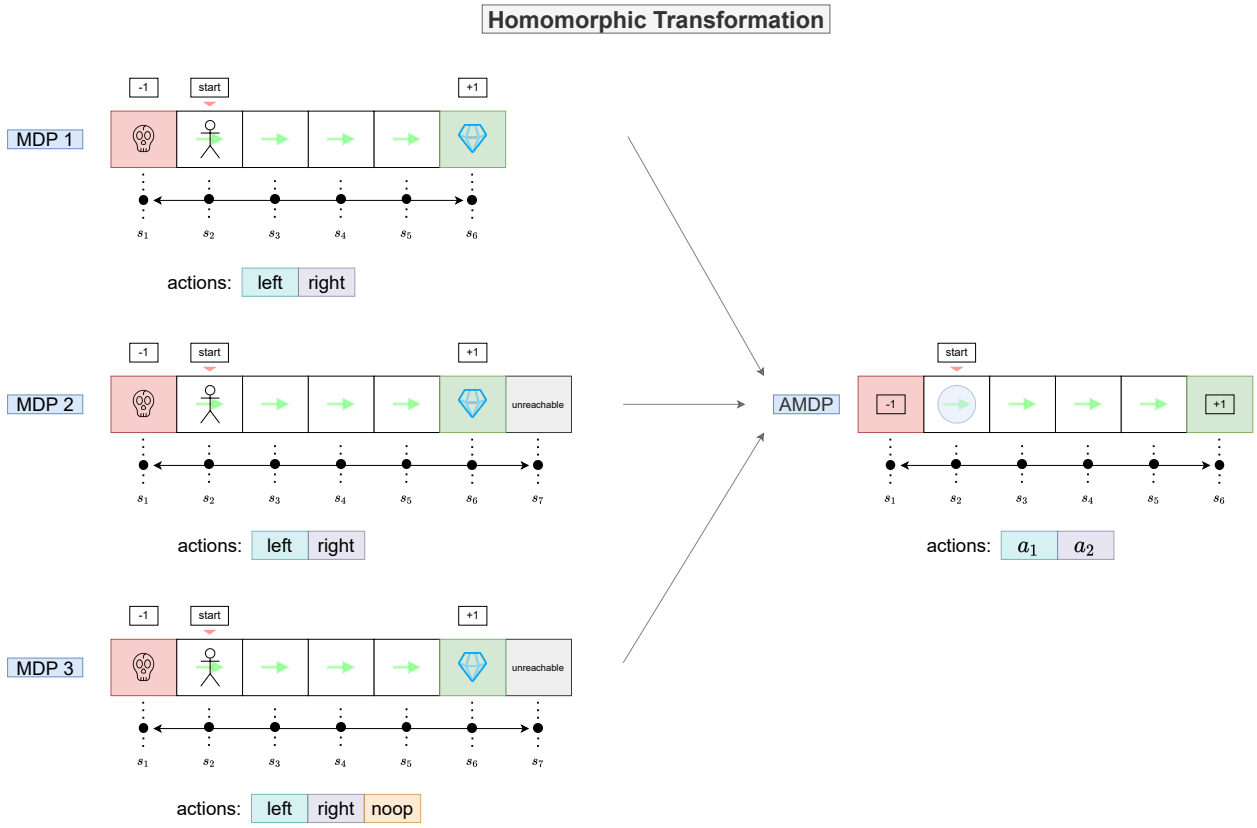


Figure A.3: An example of three homomorphic images of an abstract MDP (AMDP). All MDPs are assumed to have the same state-action support. Their realization only show minor differences, e.g. in different symbols for the agent or the goal.

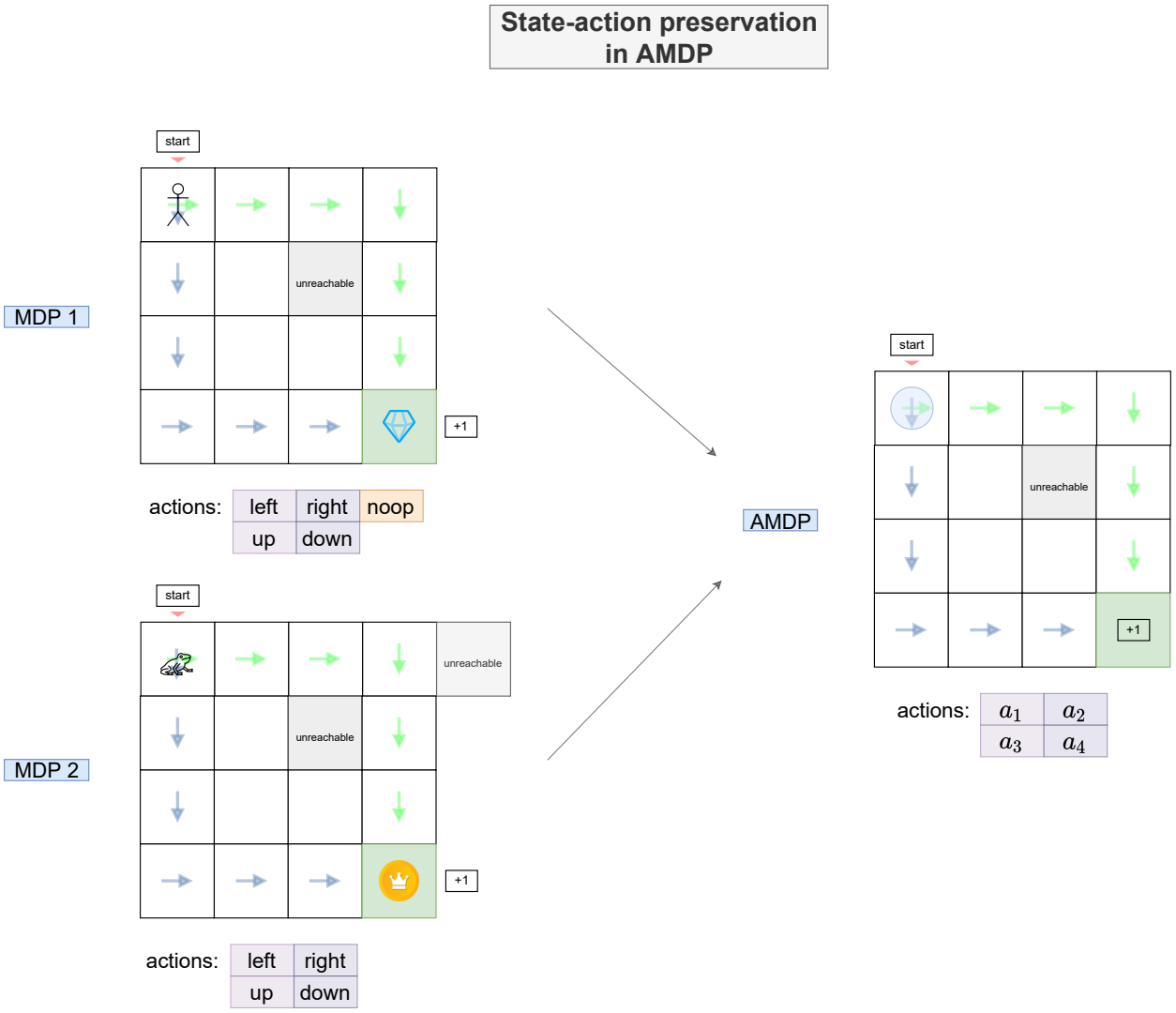


Figure A.4: An example of state-action preservation for two policy paths given two MDPs. MDP 1 and 2 are homomorphic images of an abstract MDP (AMDP). The AMDP preserves the paths from all its homomorphic images.

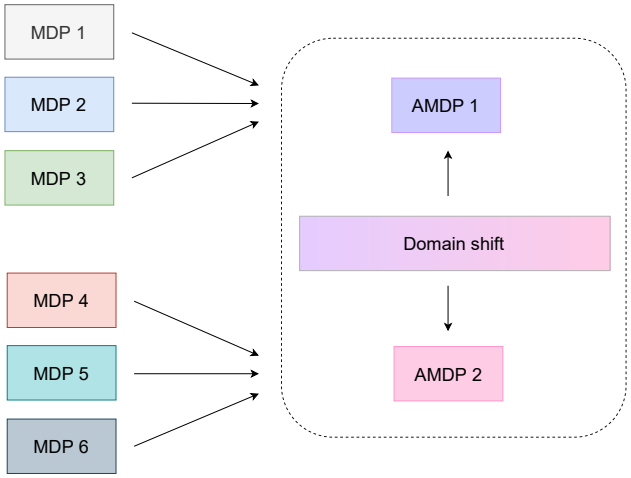


Figure A.5: We show six MDPs that are mapped to their respective abstract MDP (AMDP). The AMDPs are assumed to have the same state-action support, and are realized as domain shifted transformations of each other. Under such transformation, the dynamics of the MDPs may differ and therefore abstract policies succeeding in one AMDP may fail in the other.

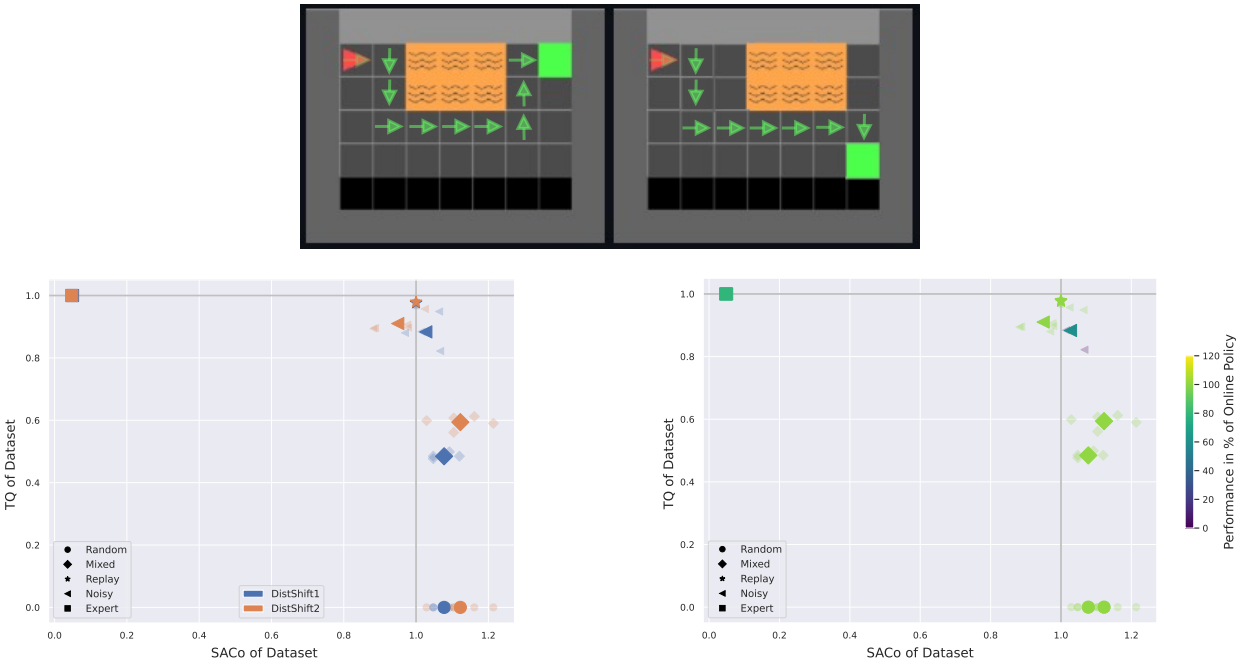


Figure A.7: In this experiment we evaluate MiniGrid with two settings, the original MDP and a domain shifted version of the orignal MDP (top, left: original, and top, right: domain shifted). In the lower left figure, we see SACo and TQ for the two generated datasets of the MDPs (blue: original, orange: domain shifted; 5 seeds each). The generated datasets measures are similar since the dynamics are similar, and therefore, are slightly shifted versions of each other. On the lower right, we see the performance of offline trained DQN on the generated datasets. We observe that the performances match almost everywhere, with slight deviation of DistShift1, which is to be expected, since it is the slightly more difficult environment. In DistShift1 the agent has to longer paths next to the lava to get toward the goal state in the corner.

A.5.2 AMDPs WITH DOMAIN SHIFT

This section provides an illustrative overview of the core properties of abstract MDPs (AMDPs) with domain shift.

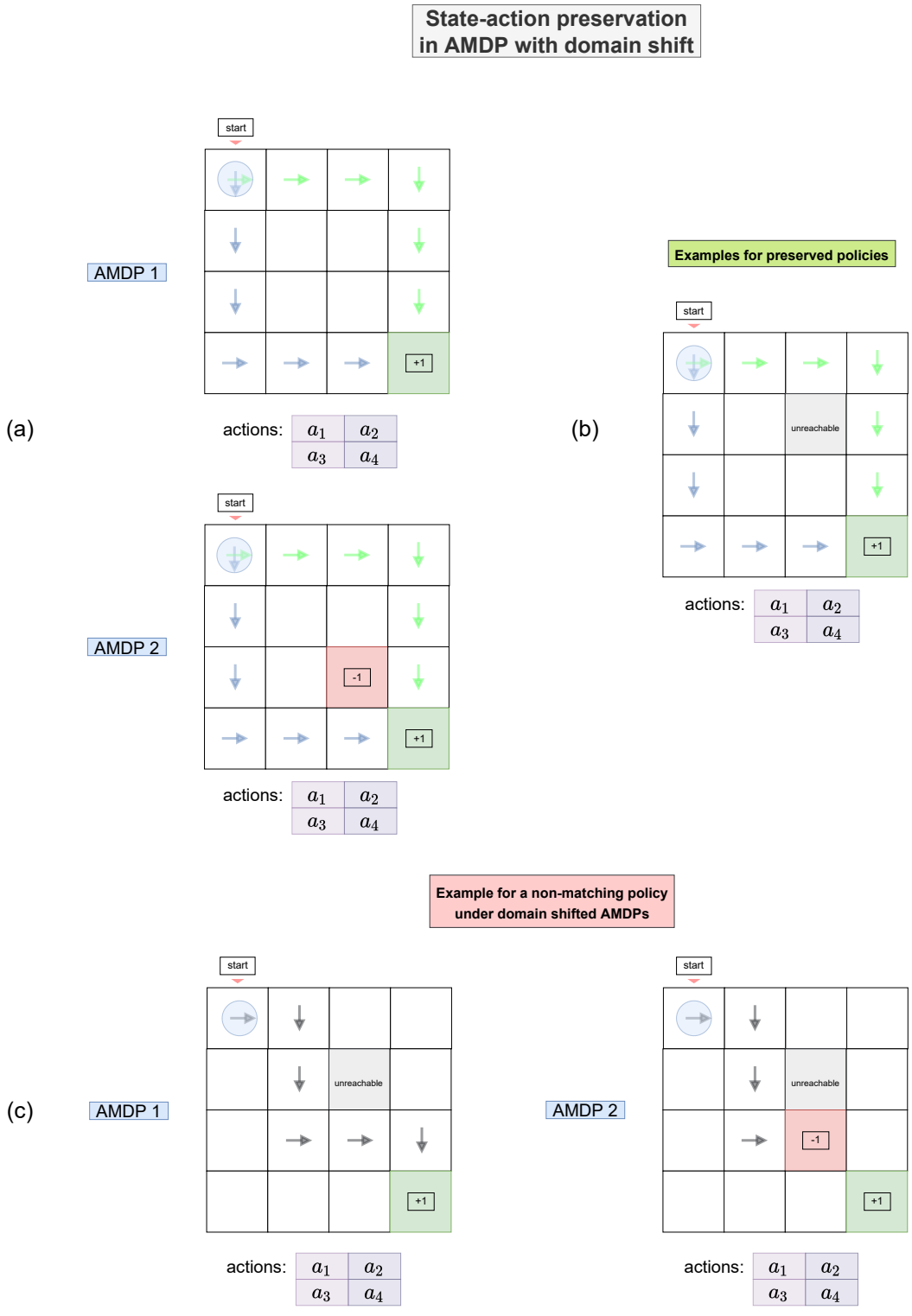


Figure A.6: (a) We consider two abstract MDPs (AMDPs) with domain shift between them, and with two policy paths. (b) In this example, we see that the two defined AMDPs partially preserve the state-action paths. (c) In this example, we see on the same AMDPs conflicting policy paths, since both AMDPs have different policies.

A.5.3 ASSESSING SACO IN CONTINUOUS STATE SPACES

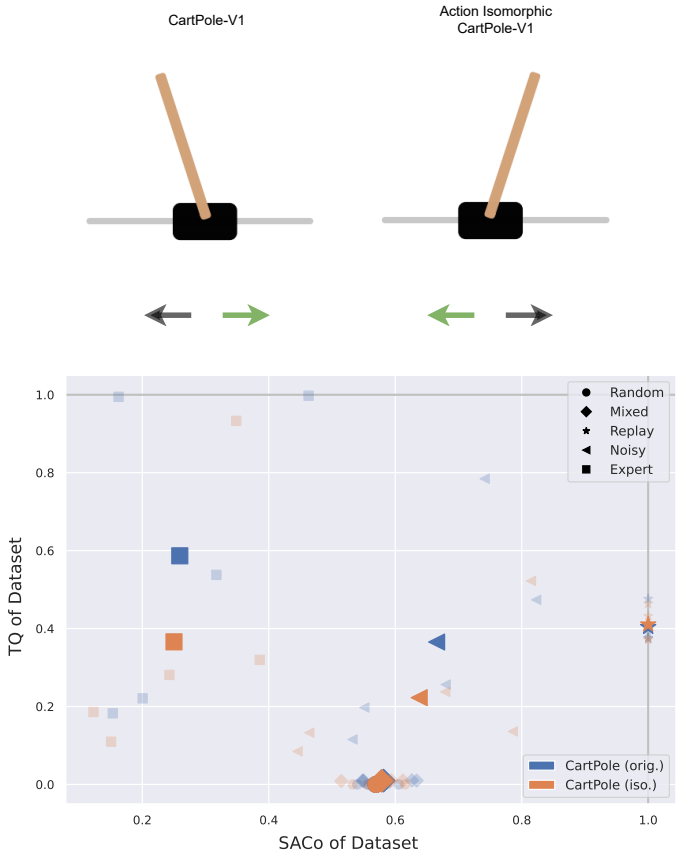


Figure A.8: In this experiment we have evaluated CartPole with an isomorphically transformed MDP that flips the actions. We conducted this experiment to assess the stability of our measures, even for continuous states. Although CartPole has a continuous state space, the values for TQ and SACo are concentrated in distinctive regions (though with high variance), which we have observed also in experiments with discrete state and action spaces. We evaluated five seeds for each MDP.

A.5.4 ASSESSING DQN VS QRDQN DATASET COLLECTION PROPERTIES

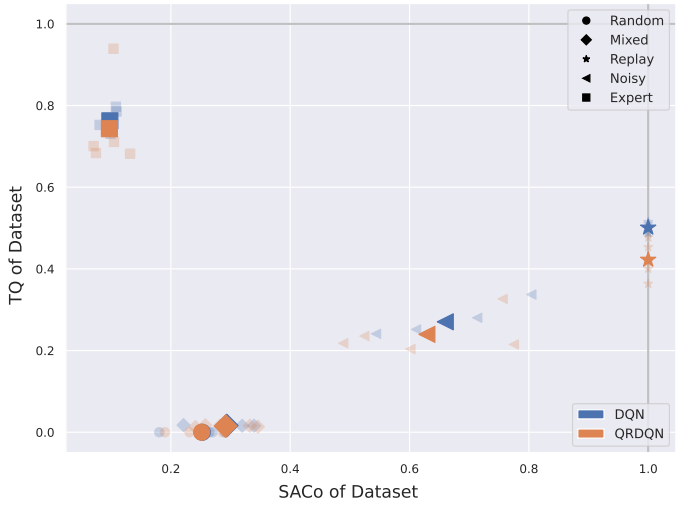


Figure A.9: In this experiment, we also considered evaluating DQN to contrast it with QRDQN when generating the datasets. We see that both methods are projected to similar cluster regions. This result is expected since DQN and QRDQN are closely related algorithms. We evaluated five seeds for each DQN variant.

A.6 EMPIRICAL EVALUATIONS OF DOMAIN SHIFTS IN DIFFERENT MDP SETTINGS

We investigate our implemented measures, TQ and SACo, regarding their properties under domain shifts. Specifically, we are interested in the stability of TQ and SACo and whether they indicate domain shifts in the underlying datasets. Our experimental setup utilizes the dataset generation schemes described in Sec. 3.1. We create datasets using these schemes on a total of six environments, that are transformations (isomorphic, homomorphic) of each other, or exhibit different domain shifts. The results are presented in Fig. A.10.

We see, that on the same environment, the TQ and SACo of different types of behavioral policies (random, expert, ...) result in different clusters of datasets. This matches the intuition that changing the policy, thus introducing a policy shift, changes the dataset distribution drastically. How prevalent the changes are becomes apparent, when comparing the results between different MDP settings. The locations of clusters, representing the same behavioral policy types, change only minor when switching between the original MDP (Breakout) to isomorphic and homomorphic transformations of it. Furthermore, we present results on general domain shifts, where we are not assured that TQ and SACo indicate the domain shift, since the joint probability distribution and thus both the policy and the environment changes.

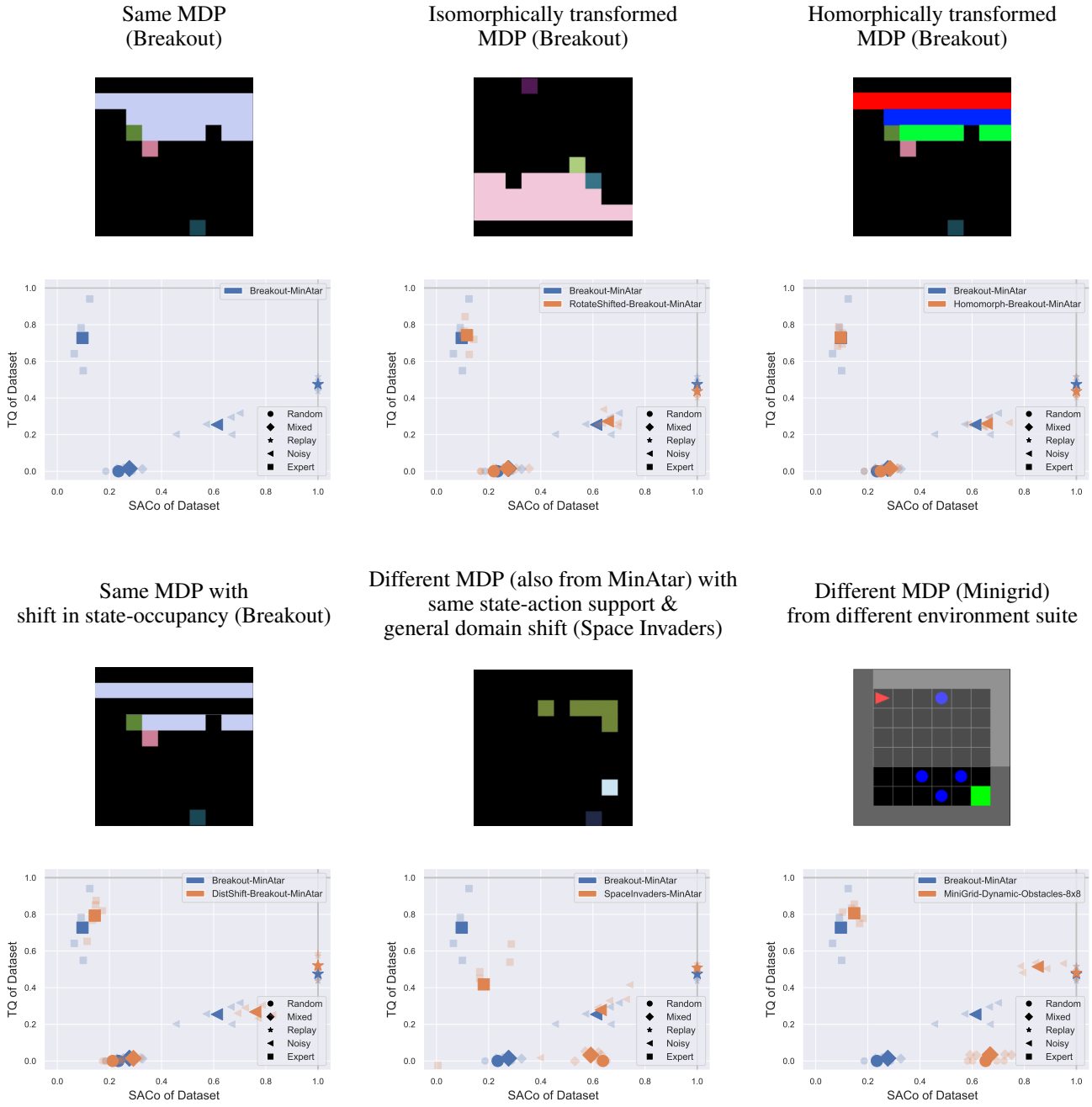


Figure A.10: We show TQ and SACo of datasets sampled from different behavioral policies in different MDP settings. Sampling from different behavioral policies corresponds to a policy shift, which is expected to result in different values for TQ and SACo. However, we also show that small deviations between datasets may occur by sampling with different seeds, even with the same type of behavioral policy. Breakout (upper left) is the reference environment all other environments are compared to. Environments in the first row exhibit isomorphic and homomorphic transformations. As expected, datasets sampled from the same behavioral policies are very similar in terms of TQ and SACo and deviate only slightly under isomorphic and homomorphic transformations of the environment. Environments in the second row exhibit domain shifts and reflect different MDP settings, which results in stronger deviations between datasets even among the same behavioral policies.

A.7 ENVIRONMENTS

Although the dynamics of the environments used throughout the main experiments are rather different and range from motion equations to predefined game rules, they share common traits. This includes the dimension of the state $dim(\boldsymbol{s})$, the number of eligible actions $|\mathcal{A}|$, the maximum episode length T_{max} as well as the minimum and maximum expected return g_{min}, g_{max} of an episode. Furthermore, the discount factor γ is fixed for every environment regardless of the specific experiment executed on it and is thus listed with the other parameters in Tab. A.1. An overview of the environments, depicted by their graphical user interfaces is given in Fig. A.11.

Two environments contained in the MinAtar suite, Breakout and SpaceInvaders, do not have an explicit maximum episode length, as the episode termination is ensured through the game rules. Breakout terminates either if the ball is hitting the ground or two rows of bricks were destroyed, which results in the maximum of 60 reward. An optimal agent could attain infinite reward for SpaceInvaders, as the aliens always reset if they are eliminated entirely by the player and there is a speed limit that aliens can maximally attain. Nevertheless, returns much higher than $200 - 300$ are very unlikely due to stochasticity in the environment dynamics that is introduced through sticky actions with a probability of 0.1 for all MinAtar environments.

Environment	$dim(\boldsymbol{s})$	$\lvert\mathcal{A}\rvert$	T_{max}	g_{min}	g_{max}	γ
`CartPole-v1`	4	2	500	9^*	500	0.95
`MountainCar-v0`	2	3	200	-200	-90^*	0.99
`MiniGrid-LavaGapS7-v0`	98	3	196	0	0.945^*	0.95
`MiniGrid-Dynamic-Obstacles-8x8-v0`	98	3	256	-1	0.935^*	0.95
`Breakout-MinAtar-v0`	100	3	-	0	60	0.99
`SpaceInvaders-MinAtar-v0`	100	4	-	0	∞	0.99

Table A.1: Environment specific characteristics and parameters. *Minimum or maximum expected returns depend on the starting state.

Action-spaces for MiniGrid and MinAtar are reduced to the number of eligible actions and state representations simplified. Specifically, the third layer in the symbolical state representation of MiniGrid environments was removed as it contained no information for the chosen environments. The state representation of MinAtar environments was collapsed into one layer, where the respective entries have been set to the index of the layer, divided by the total number of layers. The resulting two-dimensional state representations are flattened for MiniGrid as well as for MinAtar environments.

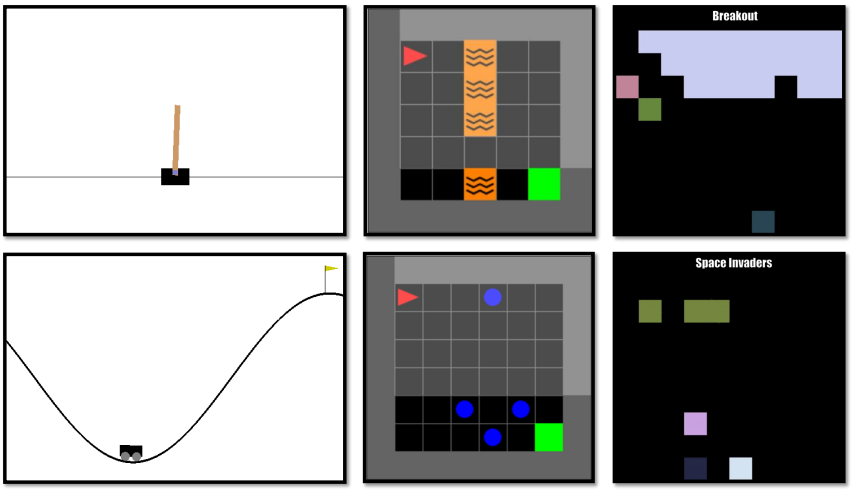


Figure A.11: Graphical interfaces of environments used throughout the main experiments. Enumerating from top to bottom, left to right, these environments are `CartPole-v1`, `MountainCar-v0`, `MiniGrid-LavaGapS7-v0`, `MiniGrid-Dynamic-Obstacles-8x8-v0`, `Breakout-MinAtar-v0` and `SpaceInvaders-MinAtar-v0`.

A.8 ALGORITHMS

We conducted an evaluation of the different dataset compositions using nine different algorithms applicable in an Offline RL setting. The selection covers recent advances in the field, as well as off-policy methods not specifically designed for Offline RL that are often utilized for comparison. Behavioral cloning (BC) (Pomerleau, 1991) serves as a baseline algorithm, as it mimics the behavioral policy used to create the dataset. Consequently, its performance is expected to be strongly correlated with the TQ of the dataset.

Behavior Value Estimation (BVE) (Gulcehre et al., 2021) is utilized without the ranking regularization that it was proposed to be coupled with. This way, extrapolation errors are circumvented during training as the action-value of the behavioral policy is evaluated. Policy improvement only happens during inference, when the action is greedily selected on the learned action-values. BVE uses SARSA updates where the next state and action are sampled from the dataset, utilizing temporal difference updates to evaluate the policy.
As a comparison, Monte-Carlo Estimation (MCE) evaluates the behavioral policy that created the dataset from the observed returns. Again, actions are greedily selected on the action-values obtained from Monte-Carlo estimates.

Deep Q-Network (DQN) (Mnih et al., 2013) is used to obtain the online policy, but can be applied in the Offline RL setting as well, as it is an off-policy algorithm. The dataset serves as a replay buffer in this case, which remains constant throughout training. As it is not originally designed for the Offline RL setting, there are no countermeasures to the erroneous extrapolation of action-values during training nor during inference.
Quantile-Regression DQN (QRDQN) (Dabney et al., 2017) approximates a set of K quantiles of the action-value distribution instead of a point estimate during training. During inference, the action is selected greedy through the mean values of the action-value distribution.
Random Ensemble Mixture (REM) (Agarwal et al., 2020) utilizes an ensemble of J action-value approximations to attain a more robust estimate. During training, the influence of each approximation on the overall loss is weighted through a randomly sampled categorical distribution. Selecting an action is done greedy on the average of the action-value estimates. Batch-Constrained Deep Q-learning (BCQ) (Fujimoto et al., 2019a) for discrete action-spaces is based on DQN, but uses a BC policy on the dataset to constrain eligible actions during training and inference. A relative threshold τ is utilized for this constraint, where eligible actions must attain at least τ times the probability of the most probable action under the BC policy.
Conservative Q-learning (CQL) (Kumar et al., 2020) introduces a regularization term to policy evaluation. The general framework might be applied to any off-policy algorithm that approximates action-values, therefore we based it on DQN as used for the online policy. Furthermore, the particular regularizer has to be chosen, where we used the KL-divergence against a uniform prior distribution, referred to as CQL($\mathcal{H}$) by the authors. The influence of the regularizing term is controlled by a temperature parameter α.
Critic Regularized Regression (CRR) (Wang et al., 2020) aims to ameliorate the problem that the performance of BC suffers from low-quality data, by filtering actions based on action-value estimates. Two filters which can be combined with several advantage functions were proposed by the authors, where the combination referred to as binary max was utilized in this study. Furthermore, DQN is used instead of a distributional action-value estimator for obtaining the m action-value samples in the advantage estimate.

A.9 IMPLEMENTATION DETAILS

A.9.1 NETWORK ARCHITECTURES

The state input space is as defined in Tab. A.1, followed by 3 linear layers with a hidden size of 256. The number of output actions for the final linear layer is defined by the number of eligible actions for action-value networks. For QRDQN and REM, the number of actions times the number of quantiles or estimators respectively is used as output size. All except the last linear layer use the SELU activation function (Klambauer et al., 2017) with proper initialization of weights, whereas the final one applies a linear activation. Behavioral cloning networks use the softmax activation in the last layer to output a proper probability distribution, but are otherwise identical to the action-value networks.

A.9.2 ONLINE TRAINING

For every environment, a single online policy is obtained through training with DQN. This policy is the one used to generate the datasets under the different settings described in Sec. 3.1. All hyperparameters are listed in Tab. A.2.

Initially, as many samples as the batch size are collected by a random policy to pre-populate the experience replay buffer. Rather than training for a fixed amount of episodes, the number of policy-environment interactions is used as training steps. Consequently, the number of training steps is independent of the agent's intermediate performance and comparable across environments. The policy is updated in every of those steps, after a single interaction with the

environment, where tuples (s, a, r, s') are collected and stored in the buffer. After the buffer has reached the maximum size, the oldest tuple is discarded for every new one. Action selection during environment interactions to collect samples starts out with an initial ϵ that linearly decays over a period of steps towards the minimal ϵ, which remains fixed throughout the rest of the training procedure. Training batches are sampled randomly from the experience replay buffer. The Adam optimizer was used for all algorithms and the target network parameters θ' is updated to match the parameters θ of the current action-value estimator every 100 training steps.

The policy is evaluated periodically after a certain number of training steps, depending on the used environment. It interacts greedy based on the current value estimate with the environment for 10 episodes, averaging over the returns to estimate its performance.

Hyperparameter	Value
Algorithm	DQN
Learning rate	0.0001
Batch size	32
Optimizer	Adam
Loss	Huber with $\lambda = 1$
Initial ϵ	1.0
Linear ϵ decay period	1 000 steps
Minimal ϵ	0.01
Target update frequency	100 steps
Training steps	100 000 (2 000 000)
Network update frequency	1 step
Experience-Replay Buffer size	50 000 (500 000)
Evaluation frequency	200 (4 000) steps

Table A.2: Online training hyperparameters, values in parenthesis apply for MinAtar environments.

A.9.3 Offline Training

If not stated otherwise, the hyperparameters for offline training are identical to the ones used during online training, stated in Tab. A.2. All others which differ in an Offline RL setting are listed in Tab. A.3. Furthermore, parameters specific to the used algorithms are stated as well, relying on the parameters provided by the original authors.

Five times as many training steps as in the online case are used for training, which is common in Offline RL since one is interested in asymptotic performance on the fixed dataset. Algorithms are evaluated after a certain number of training steps through 10 interaction episodes with the environment, as it is done during the online training. Resulting returns for each of those evaluation steps are averaged over five independent runs, given an algorithm and a dataset. The maximum of this returns is then compared to the online policy to obtain the performance of the algorithm on a specific dataset.

Algorithm	Hyperparameter	Value
All	Evaluation frequency	1 000 (20 000) steps
All	Training steps	500 000 (10 000 000)
All	Batch size	128
QRDQN	Number of quantiles K	50
REM	Number of estimators J	200
BCQ	Threshold τ	0.3
CQL	Temperature parameter α	0.1
CRR	samples for advantage estimate m	4

Table A.3: Offline training hyperparameters, values in parenthesis apply for MinAtar environments.

A.9.4 Counting Unique State-Action Pairs

Counting unique state-action pairs of large datasets is often infeasible due to time and memory restrictions. Therefore, we evaluate several methods to enable counting on large benchmark datasets. We compared 1) a simple list-based approach to store all state-action pairs, 2) a Hash-Table and 3) HyperLogLog (Flajolet et al., 2007), a probabilistic counting method. We specifically chose the HyperLogLog approach, because it can be optimally parallelized or distributed across machines and can be adapted to a "sliding window" usage (Flajolet et al., 2007), which makes it

especially useful to RL scenarios. HyperLogLog has a worst-case time complexity of $\mathcal{O}(N)$ and worst-case memory complexity of $\Theta(\log_2 \log_2 N)$, as there is no need to store a list of unique values. Even for large $N > 10^9$, estimations typically deviate by a maximum of 2% from the true counts, as shown in Flajolet et al. (2007). An overview of the time and memory complexities of all methods are provided in Tab. A.4.

Algorithm	Time complexity	Memory complexity
List of uniques	$\mathcal{O}(N^2)$	$\Theta(N)$
Hash Table	$\mathcal{O}(N)$	$\Theta(N)$
HyperLogLog	$\mathcal{O}(N)$	$\Theta(\log_2 \log_2 N)$

Table A.4: Time and Memory complexities of different algorithms that count unique state-action pairs.

Based on the presented findings, we chose HyperLogLog as a probabilistic counting method to determine the number of unique state-action pairs for each dataset.

A.9.5 Hardware and Software Specifications

Throughout the experiments, PyTorch 1.8 (Paszke et al., 2019) with CUDA toolkit 11 (Nickolls et al., 2008) on Python 3.8 (Rossum and Drake, 2009) was used. Plots are created using Matplotlib 3.4 (Hunter, 2007).

We used a mixture of 27 GPUs, including GTX 1080 Ti, TITAN X, and TITAN V. Runs for Classic Control and MiniGrid environments took 96 hours in total, the executed runs for MinAtar environments took around 20 days.

A.10 Calculating TQ and SACo

All necessary measurements for calculating the TQ and SACo are listed in this section. The maximum returns attained by the online policy are listed in Tab. A.5, the average return attained by the random policy in Tab. A.6. Furthermore, the average return and unique state-action pairs of each dataset are given in Tab. A.7 and Tab. A.8.

Environment	Maximum return of online policy $\bar{g}(\mathcal{D}_{\text{expert}})$				
	Run 1	Run 2	Run 3	Run 4	Run 5
`CartPole-v1`	500.00	500.00	500.00	500.00	500.00
`MountainCar-v0`	−99.78	−102.07	−102.70	−100.19	−99.82
`MiniGrid-LavaGapS7-v0`	0.80	0.91	0.86	0.81	0.85
`MiniGrid-Dynamic-Obstacles-8x8-v0`	0.93	0.93	0.93	0.93	0.92
`Breakout-MinAtar-v0`	18.02	19.46	17.00	18.47	19.32
`SpaceInvaders-MinAtar-v0`	26.31	25.17	28.45	28.09	28.08

Table A.5: Maximum return of the policy trained online.

Environment	Average return of the random policy $\bar{g}(\mathcal{D}_{\text{min}})$				
	Run 1	Run 2	Run 3	Run 4	Run 5
`CartPole-v1`	22.23	22.12	22.04	22.51	22.05
`MountainCar-v0`	−200.00	−200.00	−200.00	−200.00	−200.00
`MiniGrid-LavaGapS7-v0`	0.02	0.02	0.02	0.02	0.03
`MiniGrid-Dynamic-Obstacles-8x8-v0`	−1.00	−1.00	−1.00	−1.00	−1.00
`Breakout-MinAtar-v0`	0.51	0.51	0.51	0.51	0.51
`SpaceInvaders-MinAtar-v0`	2.84	2.83	2.84	2.85	2.85

Table A.6: Average return of the random policy.

Environment	Dataset	Average return of dataset trajectories $\bar{g}(\mathcal{D})$				
		Run 1	Run 2	Run 3	Run 4	Run 5
`CartPole-v1`	Random	22.23	22.12	22.04	22.51	22.05
	Mixed	27.47	27.15	26.79	26.87	26.17
	Replay	208.05	249.72	201.13	215.27	201.98
	Noisy	397.03	144.62	248.48	116.77	77.21
	Expert	497.48	498.82	279.08	127.98	109.23
`MountainCar-v0`	Random	−200.00	−200.00	−200.00	−200.00	−200.00
	Mixed	−176.36	−183.04	−179.71	−182.96	−181.01
	Replay	−159.69	−135.38	−135.44	−133.67	−136.20
	Noisy	−156.13	−164.55	−164.98	−155.13	−166.10
	Expert	−118.90	−135.23	−128.93	−134.63	−132.52
`MiniGrid -LavaGapS7-v0`	Random	0.02	0.02	0.02	0.02	0.03
	Mixed	0.09	0.05	0.16	0.10	0.08
	Replay	0.59	0.70	0.71	0.57	0.62
	Noisy	0.61	0.56	0.70	0.70	0.65
	Expert	0.63	0.42	0.75	0.70	0.57
`MiniGrid-Dynamic -Obstacles-8x8-v0`	Random	−1.00	−1.00	−1.00	−1.00	−1.00
	Mixed	−0.87	−0.88	−0.82	−0.81	−0.99
	Replay	0.58	0.71	0.53	0.57	0.46
	Noisy	−0.09	0.14	0.16	0.19	−0.42
	Expert	0.89	0.89	0.92	0.93	0.00
`Breakout-MinAtar-v0`	Random	0.51	0.51	0.51	0.51	0.51
	Mixed	0.80	0.81	0.80	0.80	0.81
	Replay	9.53	10.04	8.92	9.25	9.72
	Noisy	4.91	6.90	4.48	4.86	5.78
	Expert	13.59	15.61	12.56	14.02	15.28
`SpaceInvaders -MinAtar-v0`	Random	2.84	2.83	2.84	2.85	2.85
	Mixed	4.07	2.81	4.07	3.54	3.71
	Replay	14.85	14.66	15.62	15.46	15.48
	Noisy	9.71	3.22	11.46	11.16	13.32
	Expert	14.26	2.28	16.65	14.21	18.96

Table A.7: Average return of dataset trajectories per environment and dataset creation setting for every run.

Environment	Dataset	Unique state-action pairs in dataset $u_{s,a}(\mathcal{D})$				
		Run 1	Run 2	Run 3	Run 4	Run 5
`CartPole-v1`	Random	55 916	52 888	58 127	52 100	54 085
	Mixed	52 409	59 350	60 820	52 896	53 467
	Replay	95 384	94 749	95 950	96 499	97 263
	Noisy	70 710	64 392	78 952	53 173	51 771
	Expert	15 496	43 860	30 434	19 349	14 909
`MountainCar-v0`	Random	3 315	3 294	3 448	3 015	3 212
	Mixed	5 294	5 838	5 891	4 725	5 980
	Replay	13 740	11 183	12 411	12 444	12 549
	Noisy	14 669	14 187	14 138	12 934	14 575
	Expert	2 947	3 768	3 709	2 432	4 123
`MiniGrid` `-LavaGapS7-v0`	Random	1 842	1 847	1 879	1 919	1 840
	Mixed	1 819	1 813	1 827	1 866	1 808
	Replay	1 368	1 394	1 343	1 401	1 421
	Noisy	1 310	1 288	1 450	1 360	1 311
	Expert	114	112	116	116	104
`MiniGrid-Dynamic` `-Obstacles-8x8-v0`	Random	41 497	40 791	40 843	41 591	41 110
	Mixed	43 278	44 118	42 968	43 164	37 401
	Replay	45 283	45 423	46 916	46 191	44 801
	Noisy	46 571	49 191	45 526	44 998	40 115
	Expert	38 704	42 140	36 202	35 331	14 435
`Breakout-MinAtar-v0`	Random	16 218	15 915	16 459	16 247	16 182
	Mixed	18 351	18 608	20 175	18 179	19 166
	Replay	62 737	54 810	91 183	61 433	59 980
	Noisy	38 326	44 074	49 592	38 527	42 789
	Expert	5 809	5 914	9 006	4 950	6 535
`SpaceInvaders` `-MinAtar-v0`	Random	935 920	920 093	925 641	934 557	933 024
	Mixed	860 935	777 601	898 787	869 611	901 127
	Replay	1 507 798	1 463 980	1 446 246	1 439 305	1 426 702
	Noisy	933 016	582 548	1 053 096	955 208	1 057 379
	Expert	250 085	8 163	407 306	239 007	409 359

Table A.8: Unique state-action pairs per environment and dataset creation setting for every run.

A.11 Correlations between TQ, SACo, Policy-Entropy and Agent Performance

In the following, detailed plots showing all correlations between TQ, SACo, the entropy of the approximated behavioral policy that created the dataset, and the performance of offline agents are given. The policy's entropy is calculated by the probabilities of sampling actions for a state, given by the final network output of the same network used for BC. We included this measure as a comparison to TQ and SACo, as it is easy to obtain by a practitioner. Nevertheless, the entropy of the behavioral policy has no direct connection to exploitation or exploration as we have shown for our other measures. To give the intuition why this is the case, acting as random as possible does not guarantee to explore the state-action space thoroughly as there could be a bottleneck such as the door of a room that prevents to explore the environment further if the policy does not navigate through the door by chance. Similarly, low entropy thus acting very deterministic, does not necessarily correspond to high exploitation as it could be bad deterministic behaviour as well. Other possible measures such as using the reward distribution, episode length distribution (Monier et al., 2020), reward sparsity rate, etc. suffer from the same issue.

Scatterplots of TQ, SACo and the entropy estimate of the behavioral policy are depicted in Fig. A.12, where we observe that there is no correlation between TQ and SACo, a stronger negative correlation between the entropy and TQ and a medium positive correlation between the entropy and SACo.

Figure A.12: Scatterplots of TQ, SACo and Entropy. Annotation insets state correlation coefficients with corresponding p-value in brackets.

Scatterplots between offline agent performance and the TQ are given in Fig.A.13. As expected, BC has a very strong positive correlation with the TQ of the underlying dataset. We observe that the performance of off-policy algorithms DQN, QRDQN and REM that do not constrain the learned policy towards the behavioral policy, exhibit weak negative correlations with the TQ. Conversely, algorithms that do constrain the learned policy towards the behavioral policy, BCQ, CQL and CRR, exhibit weak positive correlations.

Scatterplots between offline agent performance and the SACo are given in Fig.A.14. While BC shows very weak if any correlation with the SACo of the underlying dataset, all other algorithms exhibit medium to high positive correlations.

Scatterplots between offline agent performance and the entropy are given in Fig.A.15. As shown in Fig.A.12, TQ and Entropy as well as SACo and Entropy are correlated. Therefore, BC exhibits a medium negative correlation with entropy, as lower-entropy datasets were created by high performing policies in our experiments.

Algorithms that constrain the learned policy towards the behavioral policy were found to be uncorrelated with the entropy of the behavioral policy, whereas all other algorithms that do not enforce such a constraint have medium positive correlations. An intuitive explanation would be that algorithms that constrain towards the behavioral policy implicitly use the entropy of the behavioral policy during training, to adjust between searching for the optimal policy and staying close to the behavioral policy. This is especially easy to see in the case of BCQ, which results in DQN if the behavioral policy is the random policy or to BC if the behavioral policy selected actions greedily.

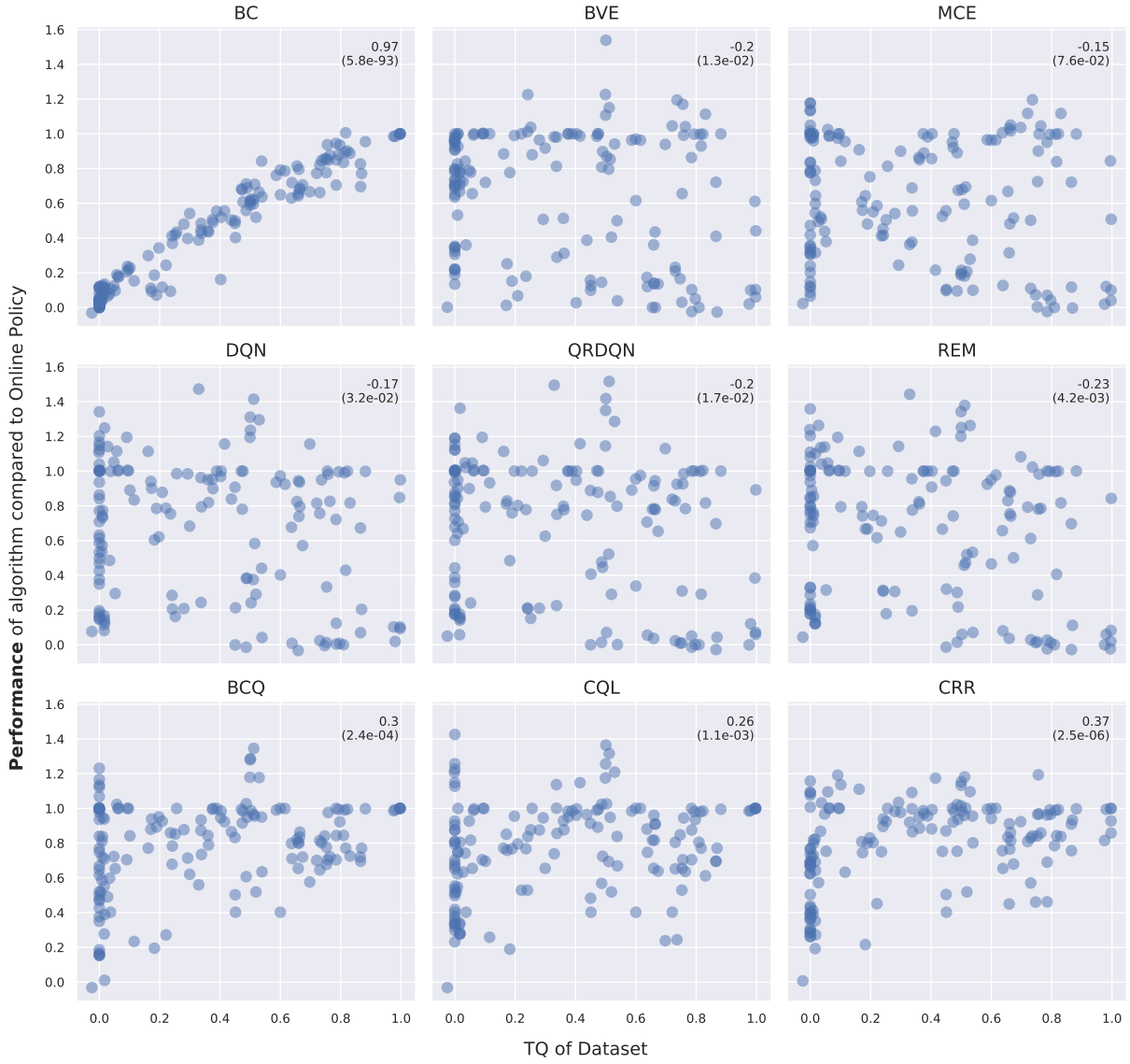


Figure A.13: Scatterplots of performance of algorithms and TQ. Annotation insets state correlation coefficients with corresponding p-value in brackets.

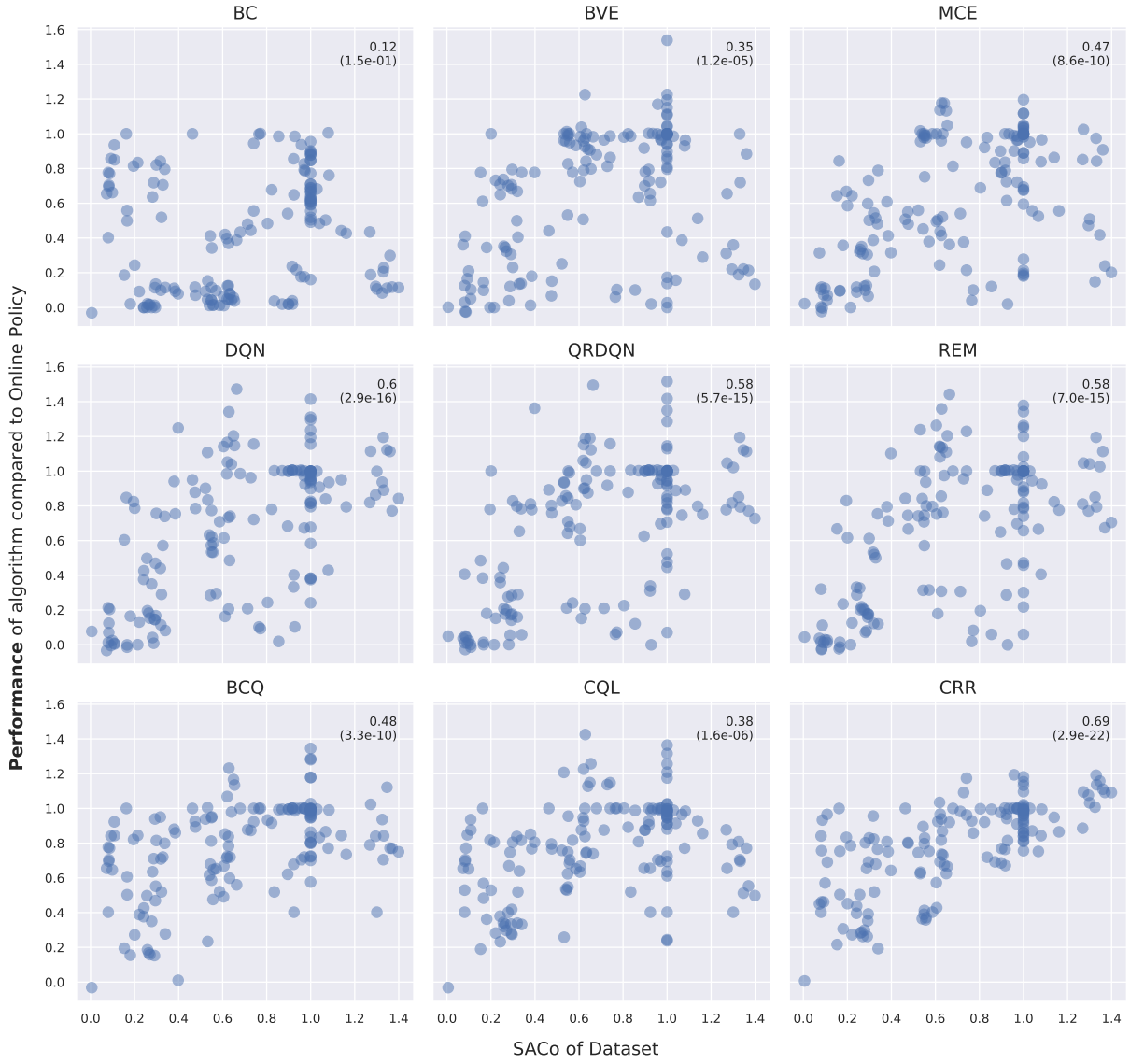


Figure A.14: Scatterplots of performance of algorithms and SACo. Annotation insets state correlation coefficients with corresponding p-value in brackets.

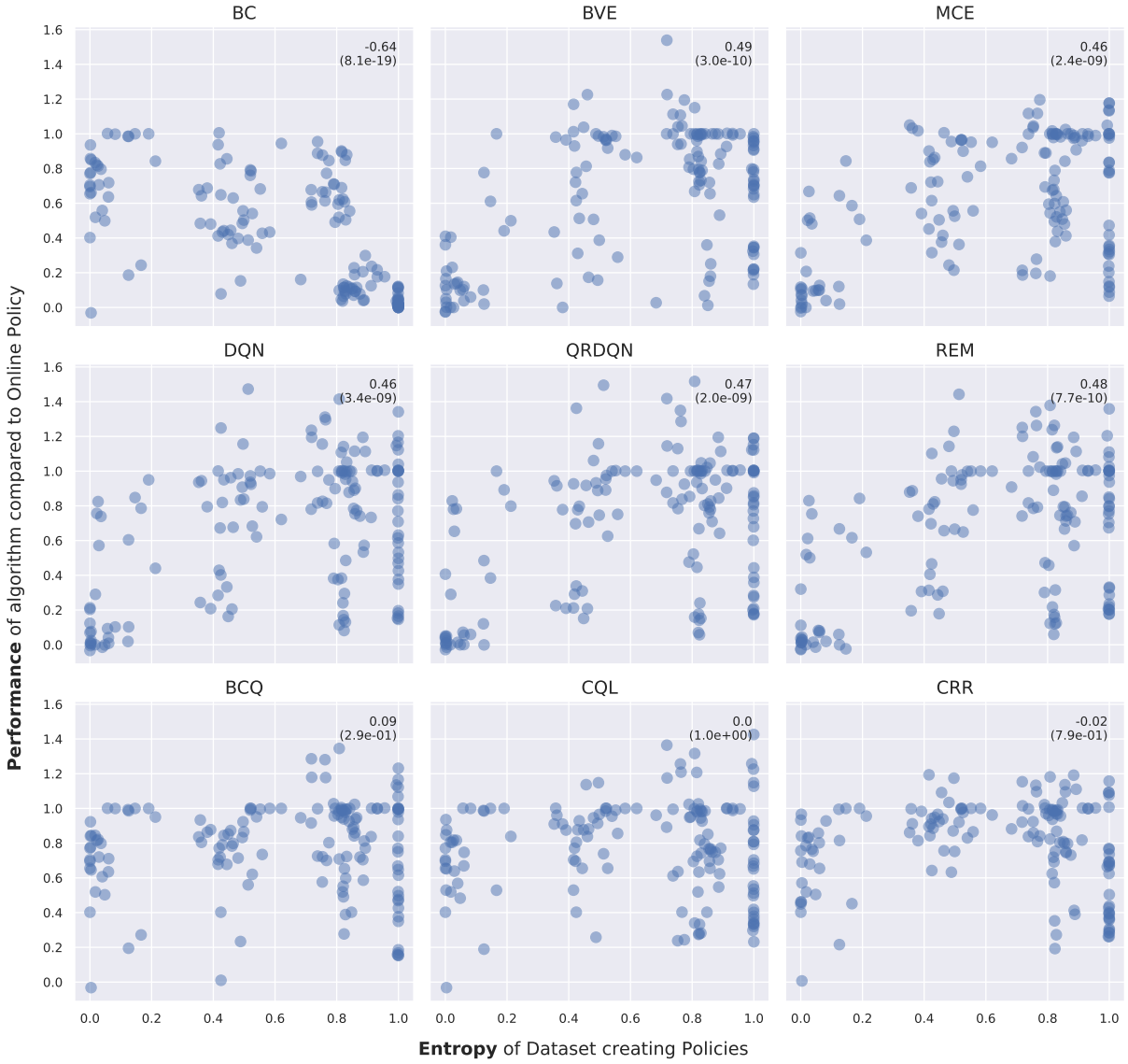


Figure A.15: Scatterplots of performance of algorithms and Entropy. Annotation insets state correlation coefficients with corresponding p-value in brackets.

A.12 Performance of Offline Algorithms

Results for the best policies learned during the offline training given the generation scheme of the dataset used for training are provided in Fig. A.16.

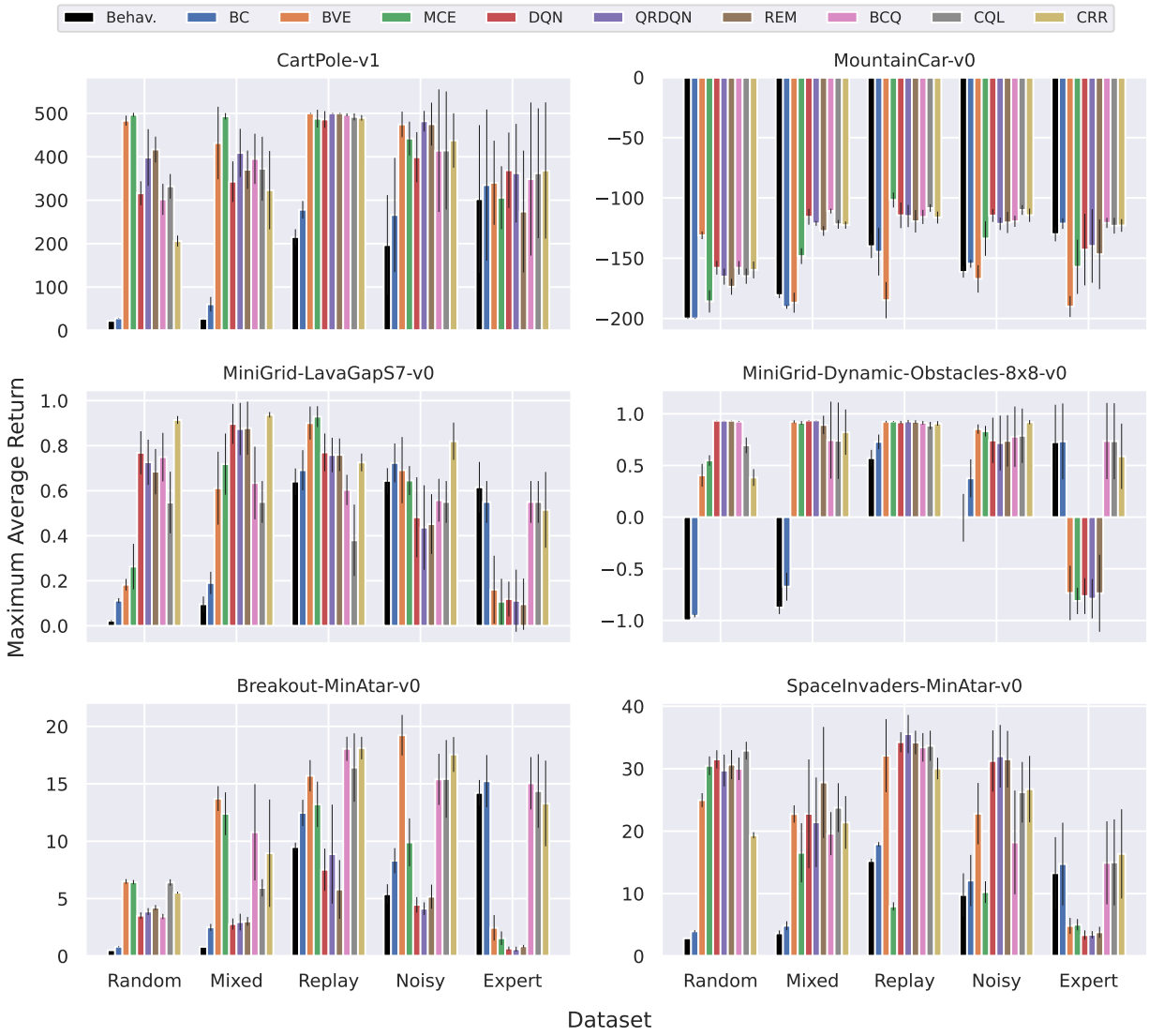


Figure A.16: Maximum average return of policies learned during the offline training. Error bars denote the standard deviation over runs and datasets created by the same dataset creation scheme on the same environment. Behav. denotes the behavioral policy used to generate the dataset, thus is the average return of episodes contained in the dataset.

Performances ω for every algorithm with the respective dataset settings are given in Tab. A.9. The results are averaged over different dataset creation seeds and multiple runs carried out with each algorithm, compared to the respective online policy used to create the dataset.

Dataset	BC	BVE	MCE	DQN	QRDQN	REM	BCQ	CQL	CRR
					CartPole-v1				
Random	0.01 ±0.00	**0.96** ±0.02	**0.99** ±0.01	0.61 ±0.06	0.79 ±0.14	0.83 ±0.06	0.59 ±0.07	0.65 ±0.06	0.38 ±0.03
Mixed	0.08 ±0.04	0.86 ±0.17	**0.99** ±0.02	0.67 ±0.10	0.81 ±0.12	0.73 ±0.09	0.78 ±0.12	0.73 ±0.15	0.63 ±0.19
Replay	0.54 ±0.04	**1.00** ±0.00	**0.97** ±0.04	**0.97** ±0.04	**1.00** ±0.00	**1.00** ±0.00	**0.99** ±0.01	**0.98** ±0.02	**0.98** ±0.01
Noisy	0.51 ±0.28	**0.95** ±0.06	0.88 ±0.08	0.79 ±0.12	**0.96** ±0.05	**0.95** ±0.10	0.82 ±0.30	0.82 ±0.28	0.87 ±0.13
Expert	**0.65** ±0.36	**0.67** ±0.20	**0.59** ±0.15	**0.73** ±0.18	**0.71** ±0.24	0.53 ±0.29	**0.68** ±0.37	**0.71** ±0.31	**0.72** ±0.33
					MountainCar-v0				
Random	0.00 ±0.00	**0.70** ±0.03	0.14 ±0.09	0.42 ±0.06	0.35 ±0.06	0.26 ±0.06	0.42 ±0.06	0.35 ±0.07	0.41 ±0.07
Mixed	0.10 ±0.02	0.13 ±0.08	0.52 ±0.07	**0.85** ±0.07	0.80 ±0.02	0.73 ±0.04	**0.90** ±0.03	0.79 ±0.03	0.78 ±0.03
Replay	0.56 ±0.20	0.15 ±0.15	**0.99** ±0.07	0.86 ±0.11	0.86 ±0.10	0.81 ±0.10	0.85 ±0.05	**0.92** ±0.03	0.85 ±0.05
Noisy	0.46 ±0.03	0.33 ±0.12	0.67 ±0.15	**0.86** ±0.06	0.79 ±0.05	0.80 ±0.09	0.81 ±0.05	**0.91** ±0.04	**0.86** ±0.06
Expert	**0.79** ±0.05	0.10 ±0.09	0.43 ±0.23	0.58 ±0.30	0.61 ±0.31	0.54 ±0.29	**0.80** ±0.04	**0.78** ±0.07	**0.78** ±0.06

Table A.9: Performance of algorithms averaged over dataset creation seeds and offline runs, where ± captures the standard deviation. Results are for Classic Control environments on all nine algorithms.

Dataset	BC	BVE	MCE	DQN	QRDQN	REM	BCQ	CQL	CRR
					MiniGrid-LavaGapS7-v0				
Random	0.11 ±0.01	0.20 ±0.03	0.30 ±0.13	0.91 ±0.12	0.86 ±0.14	0.80 ±0.13	0.88 ±0.14	0.63 ±0.14	**1.09** ±0.05
Mixed	0.21 ±0.06	0.72 ±0.22	0.85 ±0.18	**1.06** ±0.11	1.03 ±0.13	1.04 ±0.13	0.75 ±0.20	0.65 ±0.13	**1.11** ±0.05
Replay	0.81 ±0.08	**1.07** ±0.08	**1.10** ±0.06	0.91 ±0.13	0.90 ±0.12	0.90 ±0.13	0.71 ±0.07	0.43 ±0.17	0.85 ±0.04
Noisy	**0.85** ±0.12	0.82 ±0.21	0.76 ±0.10	0.57 ±0.25	0.51 ±0.26	0.53 ±0.18	0.66 ±0.13	0.65 ±0.13	**0.97** ±0.12
Expert	**0.65** ±0.13	0.17 ±0.19	0.10 ±0.12	0.12 ±0.09	0.10 ±0.15	0.08 ±0.13	**0.65** ±0.13	**0.65** ±0.13	**0.60** ±0.21
					MiniGrid-Dynamic-Obstacles-8x8-v0				
Random	0.02 ±0.01	0.73 ±0.06	0.80 ±0.03	**1.00** ±0.00	**1.00** ±0.00	**1.00** ±0.00	**1.00** ±0.00	0.88 ±0.04	0.72 ±0.04
Mixed	0.17 ±0.07	**1.00** ±0.01	**0.99** ±0.01	**1.00** ±0.00	**1.00** ±0.00	**0.98** ±0.05	0.90 ±0.19	0.90 ±0.19	0.94 ±0.11
Replay	0.90 ±0.04	**1.00** ±0.00	**1.00** ±0.00	**0.99** ±0.01	**1.00** ±0.01	**1.00** ±0.01	**0.99** ±0.01	**0.98** ±0.02	**0.99** ±0.01
Noisy	0.71 ±0.09	**0.96** ±0.02	**0.95** ±0.03	0.90 ±0.11	0.89 ±0.14	0.90 ±0.13	0.92 ±0.15	0.93 ±0.14	**0.99** ±0.01
Expert	**0.90** ±0.19	0.14 ±0.14	0.10 ±0.07	0.12 ±0.09	0.11 ±0.10	0.14 ±0.19	**0.90** ±0.19	**0.90** ±0.19	**0.82** ±0.16

Table A.10: Performance of algorithms averaged over dataset creation seeds and offline runs, where ± captures the standard deviation. Results are for MiniGrid environments on all nine algorithms.

Dataset	BC	BVE	MCE	DQN	QRDQN	REM	BCQ	CQL	CRR
					Breakout-MinAtar-v0				
Random	0.02 ±0.00	**0.33** ±0.02	**0.33** ±0.02	0.17 ±0.02	0.19 ±0.02	0.21 ±0.02	0.16 ±0.01	**0.33** ±0.02	0.28 ±0.02
Mixed	0.11 ±0.01	**0.73** ±0.05	**0.66** ±0.09	0.13 ±0.03	0.14 ±0.04	0.14 ±0.02	0.57 ±0.23	0.30 ±0.03	0.47 ±0.25
Replay	0.67 ±0.05	0.85 ±0.04	0.71 ±0.10	0.39 ±0.11	0.47 ±0.25	0.30 ±0.15	**0.98** ±0.03	**0.88** ±0.15	**0.98** ±0.03
Noisy	0.43 ±0.04	**1.04** ±0.09	0.52 ±0.09	0.22 ±0.04	0.20 ±0.03	0.26 ±0.06	0.83 ±0.09	0.82 ±0.16	**0.95** ±0.04
Expert	**0.82** ±0.09	0.11 ±0.07	0.06 ±0.04	0.01 ±0.01	0.01 ±0.01	0.02 ±0.01	**0.81** ±0.09	**0.77** ±0.15	**0.71** ±0.18
					SpaceInvaders-MinAtar-v0				
Random	0.05 ±0.01	0.91 ±0.06	**1.13** ±0.05	**1.18** ±0.10	**1.11** ±0.12	**1.15** ±0.13	**1.12** ±0.09	**1.24** ±0.11	0.68 ±0.04
Mixed	0.08 ±0.03	**0.82** ±0.08	0.57 ±0.23	**0.82** ±0.35	**0.76** ±0.29	**1.02** ±0.36	**0.69** ±0.17	**0.87** ±0.20	**0.76** ±0.15
Replay	0.62 ±0.02	**1.19** ±0.20	0.21 ±0.04	**1.29** ±0.07	**1.34** ±0.13	**1.29** ±0.07	**1.25** ±0.07	**1.26** ±0.07	1.12 ±0.06
Noisy	0.37 ±0.16	0.82 ±0.18	0.30 ±0.06	**1.16** ±0.19	**1.20** ±0.21	**1.17** ±0.16	0.62 ±0.33	**0.96** ±0.17	**0.97** ±0.18
Expert	**0.48** ±0.26	0.08 ±0.05	0.09 ±0.04	0.02 ±0.03	0.02 ±0.02	0.04 ±0.03	**0.48** ±0.27	**0.49** ±0.27	**0.54** ±0.29

Table A.11: Performance of algorithms averaged over dataset creation seeds and offline runs, where ± captures the standard deviation. Results are for MinAtar environments on all nine algorithms.

A.13 ILLUSTRATION OF SAMPLED STATE-ACTION SPACE ON MOUNTAINCAR

In Fig. A.17 we illustrate the effect of the behavioral policy on the distribution of the sampled dataset on the example of the `MountainCar` environment. This environment was chosen as the state space is two-dimensional and thus provides axes with physical meaning.

In this example, the dataset obtained through a random policy has only limited coverage of the whole state-action space. This is, because the random policy is not able to transition far from the starting position due to restrictive environment dynamics, necessitating long sequences with identical actions (swinging left and right) to deviate farther from the starting state distribution.

Furthermore, the expert policies obtained in each independent run differ from one another in how they steer the agent towards the goal, for instance, refraining to use the action *"Don't accelerate"* in the first run.

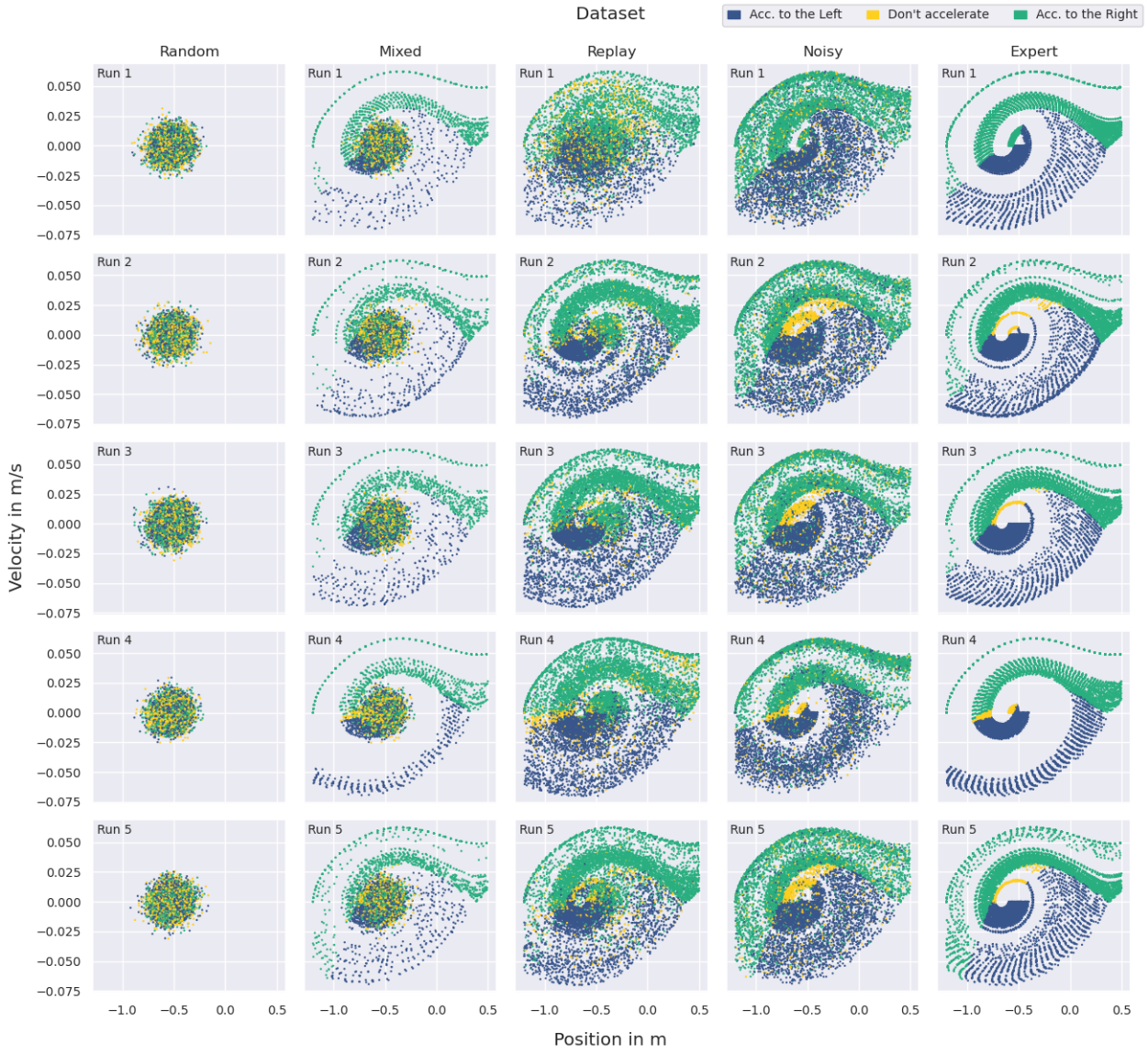


Figure A.17: State-action space for different datasets created from the environment `MountainCar` under different dataset schemes for five independent runs. 10% of the datasets were sub-sampled for plotting.

A.14 Performance per Dataset Generation Scheme

To obtain results per dataset generation scheme, the results for the five dataset creation runs per scheme are averaged. Therefore, the TQ and SACo are averaged as well as the performance for the respective algorithm on each dataset. Results are depicted in Fig. A.18.

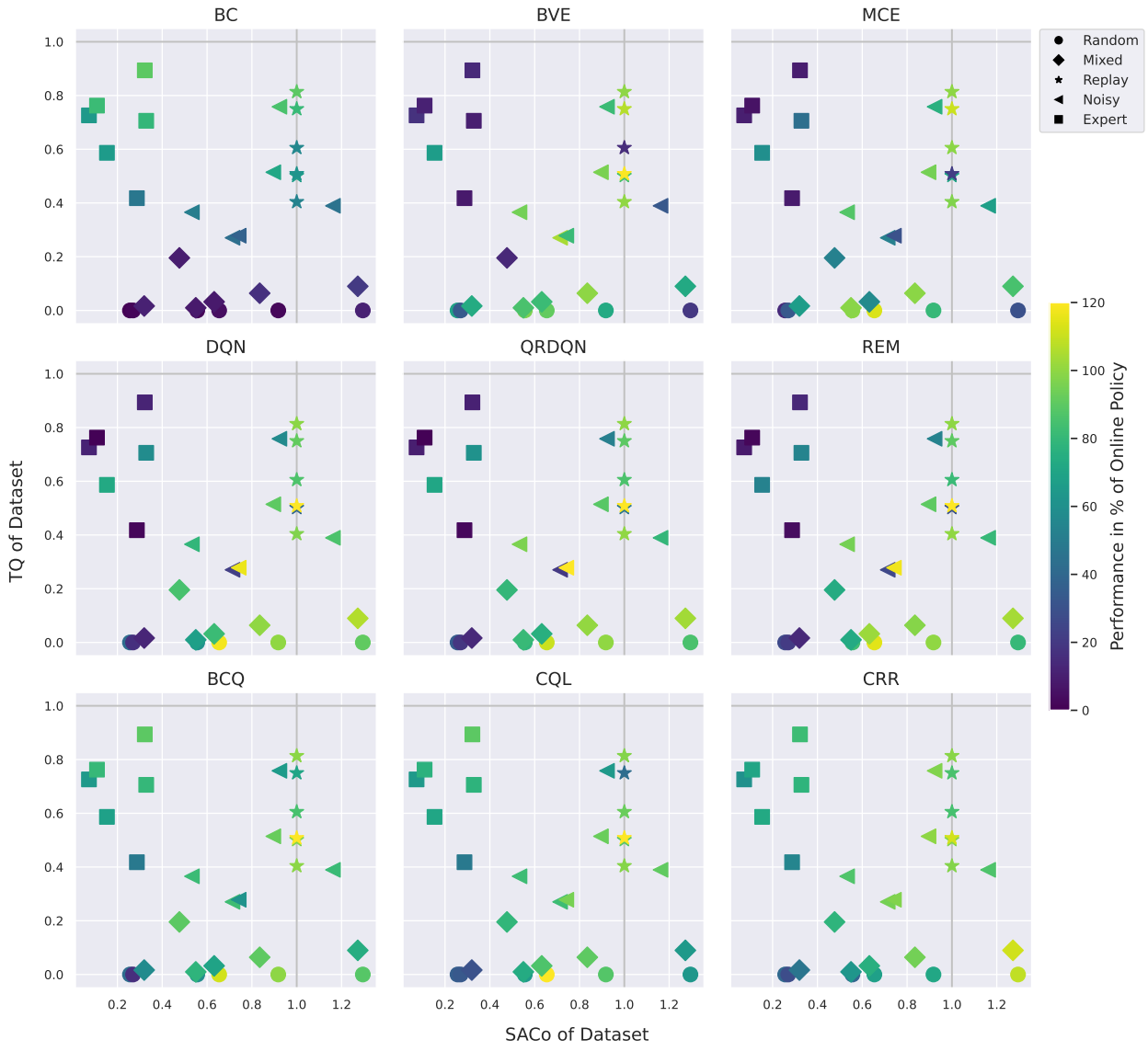


Figure A.18: Performance of algorithms compared to the online policy used to create the datasets, with respect to the TQ and SACo of the dataset. Points denote the different datasets. TQ, SACo and performance are averaged over results for each of the five dataset creation seeds.

A.15 RESULTS WITH *lSACo*

We define lSACo analogous to SACo as defined in Sec. 2.4 as

$$lSACo(\mathcal{D}) := \frac{\log(u_{s,a}(\mathcal{D}))}{\log(u_{s,a}(\mathcal{D}_{\text{ref}}))}. \tag{31}$$

In Fig. A.19 we visualize the same results as presented in Fig. 5. We conclude that our findings elaborated in Sec.4 remain the same, but are harder to interpret from an empirical point of view due to the logarithmic scaling of lSACo.

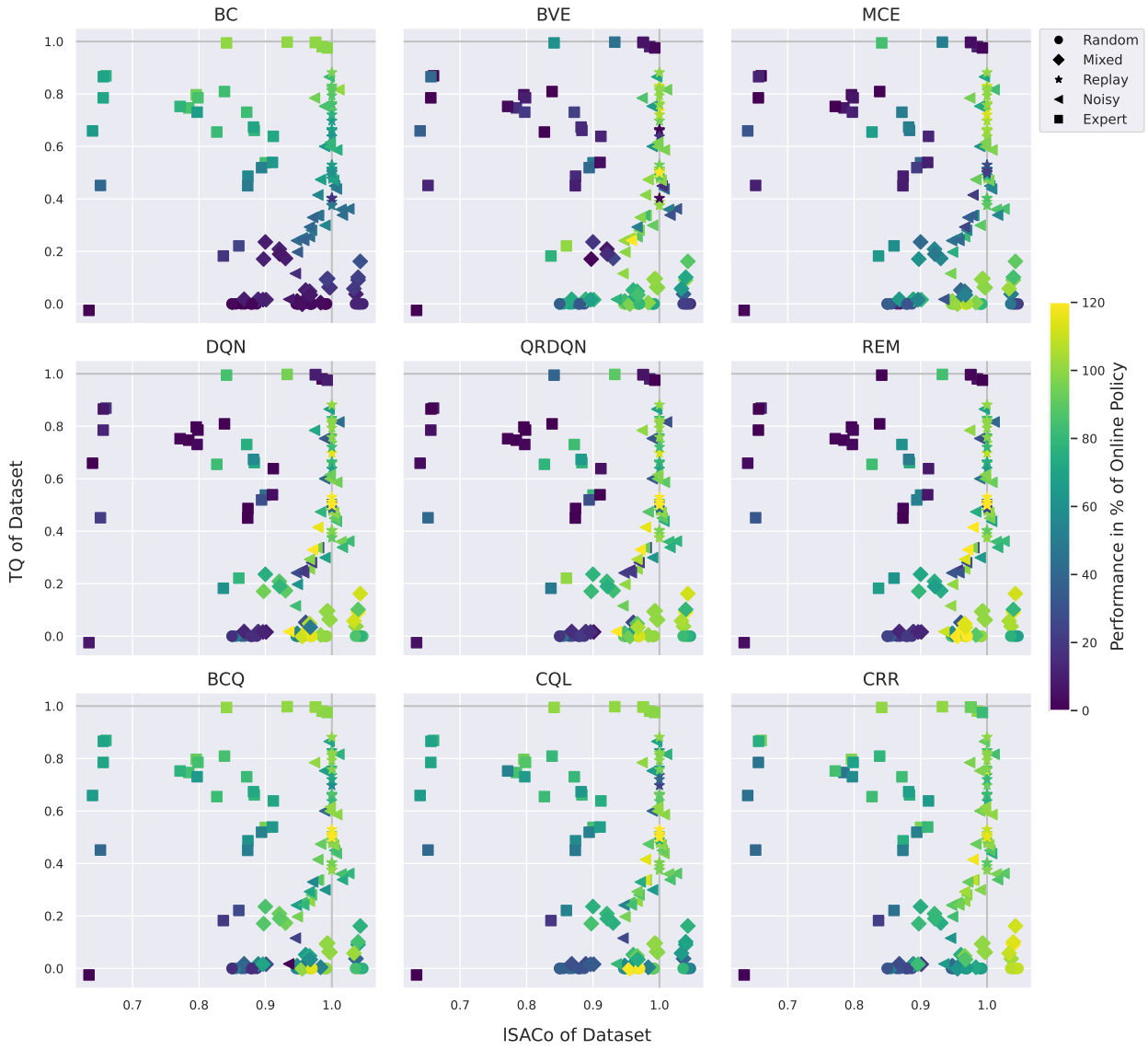


Figure A.19: Performance of algorithms compared to the online policy used to create the datasets, with respect to the TQ and lSACo of the dataset. Points denote the different datasets, color denotes performance of the respective algorithm.

A.16 Results with Naïve Entropy Estimator

As discussed in Sec. 2.4, we also evaluated our main experiments with an empirical measures based on the naïve entropy estimator $\hat{H}(\mathcal{D}) = -\sum_i^K \hat{p}_i \log(\hat{p}_i)$ with $\hat{p}_i = n_{s,a}/N$, where $n_{s,a}$ is the count of a specific visitable state-action pair (s, a) of the visitable state-action pairs K under the policy, in the dataset of size N. The entropy estimate of a given dataset is also normalized with a reference dataset $\mathcal{D}_{\text{ref}}$, where we use the replay dataset throughout all experiments. The empirical exploration measure of a dataset is thus $\hat{H}(\mathcal{D})/\hat{H}(\mathcal{D}_{\text{ref}})$.

We find qualitatively similar results to our main experiments using SACo. A major difference is the strong concentration around one, thus the entropy estimate of different datasets are close together in value. Furthermore, random datasets and mixed datasets (consisting to 80% of the random dataset) exhibit much higher scores than for SACo. This is expected, because even if a smaller number of state-action pairs is visited, chances are high the reachable states are visited relatively even under a random policy which results in a high entropy estimate.

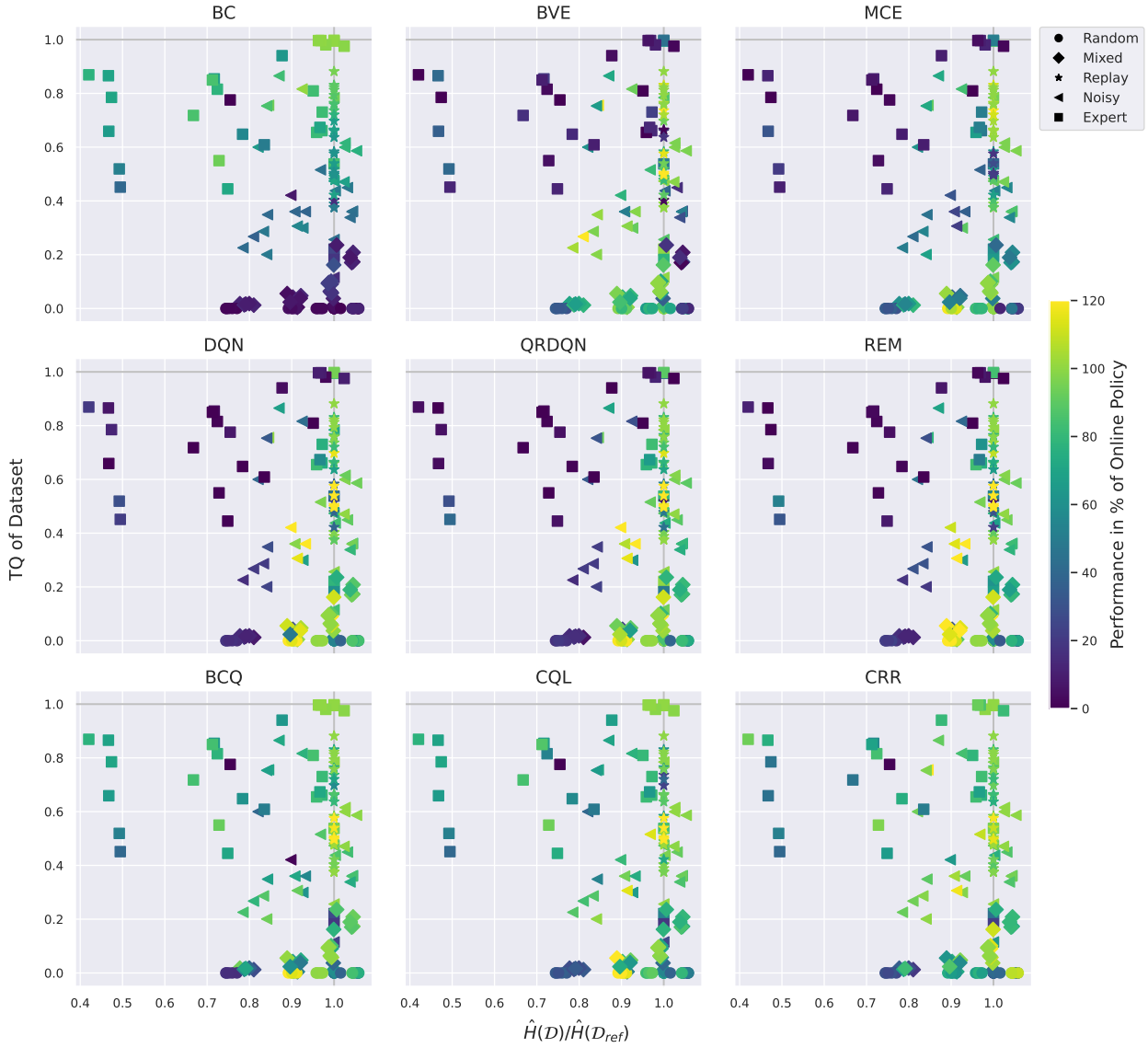


Figure A.20: Performance of algorithms compared to the online policy used to create the datasets, with respect to the TQ and $\hat{H}(\mathcal{D})/\hat{H}(\mathcal{D}_{\text{ref}})$ of the dataset. Points denote different datasets, color denotes performance of the respective algorithm.

A.17 Additional Results on D4RL Gym-MuJoCo Datasets

Many recent algorithmic advances in Offline RL for continuous action spaces (Wu et al., 2019; Kumar et al., 2019; 2020; Kostrikov et al., 2021; Fujimoto and Gu, 2021) report results on D4RL datasets (Fu et al., 2021), most prominently on the Gym-MuJoCo datasets. Here we present some additional results on these datasets, based on the results reported by Kumar et al. (2020), to investigate the applicability of our proposed measures to characterize datasets sampled from continuous state-action spaces.

Calculating SACo in continuous state-action spaces requires discretization. We discretized for each dimension for the state and action space individually, using the same ranges and bin sizes for each datasets of the same environment. In line with our experiments depicted in Fig. 5 in the main paper, we use the replay dataset as reference dataset. The reference scores for TQ are the return of the best online policy and the return of the random policy, which are generally used to normalize the performance when reporting results on these datasets (Fu et al., 2021). Performances of all algorithms are taken from Kumar et al. (2020). Results are depicted in Fig. A.21.

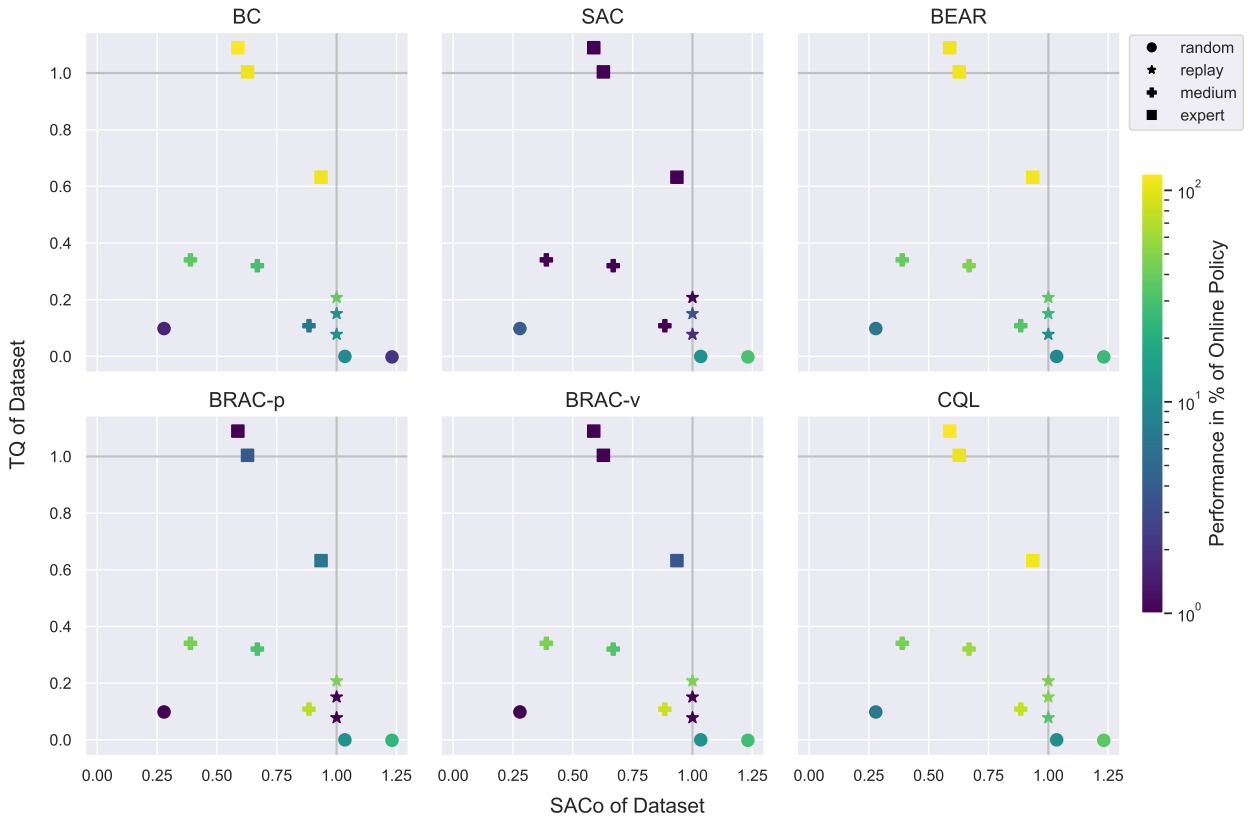


Figure A.21: Results of a variety of algorithms applicable to continuous action-spaces on D4RL Gym-MuJoCo `halfcheetah`, `walker2d` and `hopper` environments and feature the random, replay, mixed and expert datasets. Datasets are `-v0` variants as published by Fu et al. (2021). Performance of algorithms are taken from Kumar et al. (2020), who evaluates on the same datasets. The color coding of the performance is scaled logarithmic to highlight performance differences on the lower performance end.

In line with the results on discrete action environments, we find that BC has a strong correlation with TQ. The algorithm used for online learning, SAC (Haarnoja et al., 2018), only finds policies that are better than random for datasets with high SACo. Furthermore, we find that BEAR Kumar et al. (2019) and CQL (Kumar et al., 2020) attain good policies (relative to BC and SAC) for both datasets with high TQ and SACo. The results for BRAC-p and BRAC-v Wu et al. (2019) are worse than for BEAR and CQL overall, especially for the expert datasets with high TQ, where they fail to meet the performance of BC.

Overall, other than exhibited in our main experiments, algorithms only attain performance close to the online agent if trained on large portions of expert data. We hypothesize, this might be due to the nature of the robotic control environments, where steering the agent to any point in the state space is in general very complex and useful behavior

only operates in a small subspace of the full state-action space. Future work would have to investigate the effects of dynamics and dimensionality on continuous state-action space problems, as all the environments considered have similarly high dimensional state-action spaces and similar dynamics (especially `walker2d` and `hopper`).