Action Redundancy in Reinforcement Learning

Nir Baram[*,1] **Guy Tennenholtz**[*,1] **Shie Mannor**[1,2]

[*]Equal Contribution
[1]Technion, Israel Institute of Technology
[2]Nvidia Research

Abstract

Maximum Entropy (MaxEnt) reinforcement learning is a powerful learning paradigm which seeks to maximize return under entropy regularization. However, action entropy does not necessarily coincide with state entropy, e.g., when multiple actions produce the same transition. Instead, we propose to maximize the transition entropy, i.e., the entropy of next states. We show that transition entropy can be described by two terms; namely, model-dependent transition entropy and **action redundancy**. Particularly, we explore the latter in both deterministic and stochastic settings and develop tractable approximation methods in a near model-free setup. We construct algorithms to minimize action redundancy and demonstrate their effectiveness on a synthetic environment with multiple redundant actions as well as contemporary benchmarks in Atari and Mujoco. Our results suggest that action redundancy is a fundamental problem in reinforcement learning.

1 INTRODUCTION

Maximum Entropy (MaxEnt) algorithms are successful reinforcement learning (RL) approaches [Ziebart et al., 2008, Yin, 2002, Haarnoja et al., 2018], achieving superior performance through entropy regularization. The perturbed objective function promotes policies to maximize the reward while remaining stochastic, thereby encouraging exploration and stability. [Ziebart, 2010]. Nevertheless, concurrent MaxEnt methods maximize action entropy, a proxy of the desired state entropy.

Indeed, maximizing action entropy instead of state entropy may prove detrimental when these quantities diverge. Consider for example the case in which a subset of actions yields the same next state s', as depicted in Figure 1. In this scenario, a uniform action distribution would increase the probability of visiting state s' at the expense of not exploring other states.

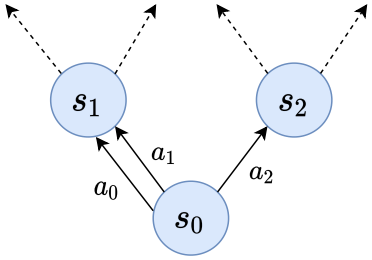


Figure 1: **Suboptimality of Action Entropy**. Plot depicts a deterministic MDP in which two actions transition to the same state. Maximizing the action entropy would bias transitions toward s_1, thereby producing suboptimal next-state entropy. Particularly, maximizing action entropy would result in a uniform distribution over the sets $\{a_0\}, \{a_1\}, \{a_2\}$, whereas maximizing transition entropy would result in a uniform distribution over $\{a_0, a_1\}, \{a_2\}$.

From an exploratory standpoint, generating uniform state visitation frequencies is an optimal strategy one can opt for in absence of any reward signal [Pitis et al., 2020, Pratissoli, 2020]. It turns out that finding a policy with a uniform state distribution can be cast as a convex optimization problem [De Farias and Van Roy, 2003]. However, this result holds only if the underyling MDP is fully specified. A less demanding setting is discussed in Hazan et al. [2019], where a provably efficient exploration algorithm requires access to a black-box planning oracle.

In this work we propose to use transition entropy, i.e., next-state entropy, as a proxy for the state entropy. As the name implies, the next-state entropy measures the entropy a policy induces over states after a single step. Perhaps surprisingly, we show that estimating it is possible without having to learn a forward model for transitions.

Supplement for the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021).

We begin by showing the expected transition entropy can be cast as the expectation of two terms, one representing model-dependent entropy, and the other *action redundancy*. While the first term is attractive from a pure exploration point of view, it is undesirable when optimizing a controlled policy. For this reason, we focus on action redundancy and its implications. We derive and illustrate action redundancy in the stochastic and deterministic settings. We then derive value functions for the cumulative action-redundancy-to-go, allowing us to incorporate a novel regularization technique to both value-based [Watkins and Dayan, 1992] as well as actor-critic methods [Grondman et al., 2012].

2 PRELIMINARIES

We assume a Markov Decision Process (MDP) [Puterman, 2014] which specifies the environment as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, r, P, \gamma, \rho_0 \rangle$, consisting of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a reward function $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, a transition probability function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$, a discount factor $\gamma \in (0, 1)$, and an initial state distribution $\rho_0 : \mathcal{S} \mapsto [0, 1]$.

A policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ interacts with the environment sequentially, starting from an initial state $s_0 \sim \rho_0$. At time $t = 0, 1, \ldots$ the policy produces a probability distribution over the action set $\mathcal{A}$ from which an action $a_t \in \mathcal{A}$ is sampled and played. The environment then generates a scalar reward $r(s_t, a_t)$ and a next state s_{t+1} is sampled from the transition probability function $P(\cdot|s_t, a_t)$. The value of a policy may be expressed by the state-action value function $Q^\pi(s, a) = \mathbb{E}^\pi\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a\big]$, or the state value function $v^\pi(s) = \mathbb{E}_{a\sim\pi} Q^\pi(s, a)$.

We use $\rho_\pi(s, a) \triangleq (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P^\pi(s_t = s, a_t = a | s_0 \sim \rho_0)$ and $\rho_\pi(s) \triangleq \sum_a \rho_\pi(s, a)$ to denote the discounted state-action and state visitation distributions of policy π, respectively. Finally we use $\mathbb{1}_A(x)$ to denote the indicator function of a subset A at x.

Maximum Action Entropy. The MaxEnt framework augments the expected return by introducing a discounted action entropy term

$$O_{AE}(s, \pi) = v^\pi(s) + \alpha \mathbb{H}(s, \pi),$$

where $\alpha \in \mathbb{R}_+$ and $\mathbb{H}(s, \pi)$ is the the discounted entropy, recursively defined by

$$\mathbb{H}(s, \pi) = \mathbb{E}_{a\sim\pi(\cdot|s)}\big[g_\pi(a, s) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)} \mathbb{H}(s', \pi)\big], \tag{1}$$

and $g_\pi(a, s) = -\log \pi(a|s)$. The discounted entropy can then be expressed recursively as [Nachum et al., 2017]

$$O_{AE}(s, \pi) = \mathbb{E}_{a\sim\pi(\cdot|s)}\big[r(s, a) + \alpha g_\pi(a, s) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)} O_{AE}(s', \pi)\big].$$

3 TRANSITION ENTROPY

We generalize the maximum action entropy framework to next-state transition entropy[1]. Intuitively, actions resulting in similar transitions should be treated similarly [Tennenholtz and Mannor, 2019, Chandak et al., 2019]. Maximum action entropy translates to maximum transition entropy whenever no overlap between actions exists, i.e., whenever they induce distinct next states. In practice, this assumption is rarely met, as shown in Figure 1. It is thus reasonable to assume that maximum action entropy rarely coincides with maximum next-state transition entropy.

In this section we formalize and define the transition entropy criterion, showing it can be cast as a sum of two terms – a model entropy term and an "action redundancy" term. Informally, redundancy in actions corresponds to their proximity w.r.t. their induced next-state transitions. In the sections that follow we provide methods to approximate action redundancy and propose to reweigh action scores of O_{AE} (see Section 2) according to their redundancy, allowing for better exploration in terms of state entropy.

3.1 FROM ACTION TO TRANSITION ENTROPY

We treat transition entropy as a myopic proxy for the discounted state entropy. As its name implies, transition entropy measures the entropy π induces over states after a single step. As such, it is possible to estimate it from single-step transitions. To this end, we are interested in quantifying the discounted next-state transition entropy induced by a policy π. Let $P(s'|s, \pi) = \mathbb{E}_{a\sim\pi(s)} P(s'|s, a)$ denote the probability of transitioning from state s to s' following policy π. We define the transition entropy to-go as follows.

Definition 1. *We define the discounted transition entropy to-go following state s by*

$$\mathbb{F}(s, \pi) = -\sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t\sim\rho_\pi} \mathbb{E}_{s'\sim P(\cdot|s_t,\pi)} \Big[\log P(s'|s_t, \pi) \Big| s_0 = s\Big].$$

We can derive a recursive formula similar to Equation 1 for the discounted transition entropy $\mathbb{F}(s, \pi)$, as shown by the following proposition.

Proposition 1. *Let $\mathbb{F}(s, \pi)$ as defined above. Then,*

$$\mathbb{F}(s, \pi) = \mathbb{E}_{a\sim\pi(\cdot|s)}\big[g_\pi(s, a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)} \mathbb{F}(s', \pi)\big],$$

where

$$g_\pi(s, a) = -\mathbb{E}_{s'\sim P(\cdot|s,a)} \log P(s'|s, \pi). \tag{2}$$

[1] Although state entropy is a preferred exploration criterion, it is by no means straightforward to estimate. As an intermediate step, we explore next-state transition entropy.

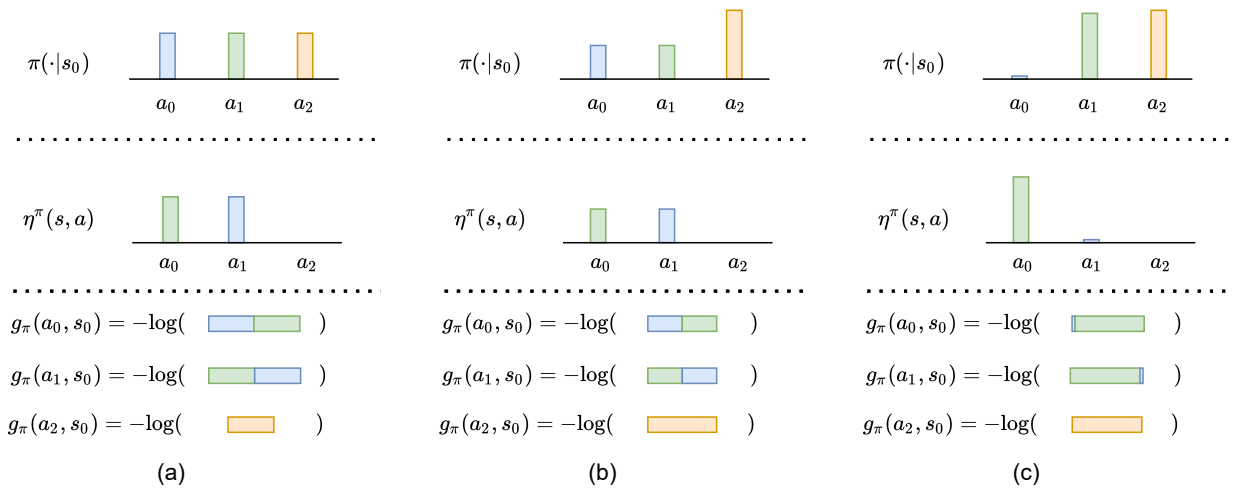


Figure 2: **Action Redundancy Score (ARS)**. Plot depicts ARS and g_π w.r.t. the deterministic MDP in Figure 1. Here, both actions a_0 and a_1 transition to the same state. Top plot shows the action probabilities $\pi(a|s)$, middle plot the ARS, and bottom plot depicts the value of $g_\pi(s,a) = -\log(\pi(a|s) + \eta^\pi(s,a))$ (see Proposition 2). **(a)** The uniform policy which maximizes action entropy does not maximize transition entropy. **(b, c)** Examples of policies which maximize the transition entropy by minimizing overall action redundancy.

Notice the resemblance of $\mathbb{F}(s,\pi)$ in Proposition 1 to that of $\mathbb{H}(s,\pi)$ in Equation 1 with a distinct definition for g_π. We can now define the Transition Entropy objective similarly to the Action Entropy objective in Section 2 as

$$O_{TE}(s,\pi) = v^\pi(s) + \alpha\mathbb{F}(s,\pi).$$

In what follows, we decompose $g_\pi(s,a)$ in Equation 2 to two terms: (1) a model-dependent entropy term, and (2) the action-redundancy term, for which we derive practical approximation methods for in both deterministic and stochastic MDPs.

3.2 FROM TRANSITION ENTROPY TO ACTION REDUNDANCY

Recall the definition of g_π in Equation 2, and notice that it can equivalently be written as follows

$$\begin{aligned} g_\pi(s,a) &= -\mathbb{E}_{s'\sim P(\cdot|s,a)} \log P(s'|s,\pi) \\ &= -\mathbb{E}_{s'\sim P(\cdot|s,a)} \left(\log P(s'|s,a) + \log \frac{P(s'|s,\pi)}{P(s'|s,a)} \right) \\ &= \underbrace{\mathcal{H}(s'|s,a)}_{\text{model entropy}} + \underbrace{D_{KL}(P(\cdot|s,a)||P(\cdot|s;\pi))}_{\text{action redundancy}}. \end{aligned} \tag{3}$$

We find that $g_\pi(s,a)$ is comprised of two meaningful terms. The first term represents the transition entropy given a state action pair (s,a), whereas the second term represents the Kullback-Leibler divergence between the transition distribution given the state-action pair (s,a) and the *expected* next-state distribution induced by π.

The result in Equation 3 implies that two terms account for the *transition score* of an action a in a state s. First, the transition entropy of an action reflects the intrinsic (uncontrollable) entropy of the environment. Note that in deterministic or quasi-deterministic environments this term vanishes. Indeed, physics-based domains are mostly deterministic, allowing us to shift our focus to the second, controllable term. The second term reflects the "importance" of an action w.r.t. the mean policy-induced transition. It is defined by the KL divergence between the state distribution an action produces and the expected distribution π induces. As such, actions that achieve similar next-state distributions to the mean behavior would receive low scores. We thus recognize the latter as the action redundancy term.

While the intrinsic model entropy is appealing from an exploratory viewpoint, it is undesired from a control perspective [Pong et al., 2019]. In the remainder of the paper, we will focus on the second term, where we explain its meaning in terms of action redundancy and present tractable methods to approximate it.

4 ACTION REDUNDANCY

In the previous section we showed that the expected next-state entropy to-go can be written as the sum of a model entropy term and an action redundancy term. While the former is intrinsic to the given environment, the latter is policy-dependent, and can thus be optimized to increase next-state entropy. In this section we derive expressions for action redundancy in both the deterministic and stochastic settings and design approximation methods. In the sections that follow we will construct algorithmic solutions based on these methods and demonstrate their effectiveness to eliminate redundancy in various domains.

Entropy Type	Action Redundancy	$g_\pi(s,a)$	Comments
Action Entropy	0	$-\log \pi(a\|s)$	[Ziebart et al., 2008]
Transition Entropy (Deterministic MDP)	$\eta^\pi(s,a)$ (ARS)	$-\log\left[\pi(a\|s)+\eta^\pi(s,a)\right]$	$\eta^\pi(s,a)=\mathbb{E}_{\tilde{a}\sim\pi(\cdot\|s)}\mathbb{1}_{R_q(s,a)}(\tilde{a})$ (Proposition 3)
Transition Entropy (Stochastic MDP)	$\zeta^\pi(s,a,s')$ (ARR)	$\mathcal{H}(s'\|s,a)+\mathbb{E}_{s'\sim P(\cdot\|s,a)}\zeta^\pi(s,a,s')$	$\zeta^\pi(s,a,s')=\log\frac{q^\pi(a\|s,s')}{\pi(a\|s)}$

Table 1: Action and transition entropy from an action redundancy perspective. $\mathbb{F}(s,\pi)=\mathbb{E}_{a\sim\pi(\cdot|s)}\left[g_\pi(s,a)+\gamma\mathbb{F}(s',\pi)\right]$.

4.1 DETERMINISTIC CASE

To better understand the implications of action redundancy and its relation to action entropy, we first turn to analyze the special case of deterministic MDPs, i.e., deterministic transitions. As we will see, in the deterministic case the model entropy vanishes, and action redundancy is reduced to an action equivalence set.

Assume a deterministic MDP with transition function $f:\mathcal{S}\times\mathcal{A}\to\mathcal{S}$. We begin by defining the action redundancy score (ARS) of an action $a\in\mathcal{A}$.

Definition 2. *Let π be a policy. The action redundancy score (ARS) of action a at state s is defined by*

$$\eta^\pi(s,a)=\mathbb{E}_{\tilde{a}\sim\pi(\cdot|s)}\mathbb{1}_{R(s,a)}(\tilde{a}), \tag{4}$$

where $R(s,a)=\{\tilde{a}\in\mathcal{A}:\tilde{a}\neq a, f(s,\tilde{a})=f(s,a)\}$.

The ARS measures the total probability measure that π assigns to other actions, $\tilde{a}\neq a$, such that $f(s,\tilde{a})=f(s,a)$. Indeed, whenever $\mathcal{A}$ is discrete we have that

$$\eta^\pi(s,a)=\sum_{\substack{\tilde{a}\neq a\\ f(s,\tilde{a})=f(s,a)}}\pi(\tilde{a}|s).$$

The ARS is closely related to the action redundancy term in Equation 3, as shown by the following proposition (see the appendix for proof).

Proposition 2. *Let M be a deterministic MDP. Then*

$$g_\pi(s,a)=-\log\left(\pi(a|s)+\eta^\pi(s,a)\right).$$

Recall that for deterministic MDPs, $\mathcal{H}(s'|s,a)=0$. Then, by Proposition 2 and Equation 3 we have that

$$D_{KL}(P(\cdot|s,a)||P(\cdot|s;\pi))=-\log\left(\pi(a|s)+\eta^\pi(s,a)\right).$$

This result suggests that by adding $\eta^\pi(s,a)$ to the traditional entropy bonus, $\log\pi(a|s)$, we essentially maximize the action entropy of a **modified action space** in which all redundant actions are unified, as depicted in Figure 2. Specifically, notice that if $\eta^\pi(s,a)=0,\forall s,a$ (i.e., no two actions are assigned the same transition) the ARS is reduced to the discounted action entropy, as stated by the following corollary.

Corollary 1. *Let M be a deterministic MDP such that $f(s,a)\neq f(s,a'),\forall s,a\in\mathcal{S}\times\mathcal{A}$. Then*

$$\mathbb{F}(s,\pi)=\mathbb{H}(s,\pi)\quad,\quad\forall s\in S.$$

Estimating $\eta^\pi(s,a)$. Proposition 2 provides us with the basic building block for minimizing action redundancy (and thus maximizing transition entropy) using the scores $g_\pi(s,a)=-\log\left(\pi(a|s)+\eta^\pi(s,a)\right)$. Nevertheless, estimating the ARS, $\eta^\pi(s,a)$, entails two major difficulties. First, as the transition function $f(s,a)$ is unknown, one must construct a forward model of $f(s,a)$. Second, a proper metric over states must be designed to execute the comparison over transitions. It turns out that we can sidestep these issues using an action equivalence relation.

Denote by $q^\pi(a|s,s')$ the posterior action distribution given that the state s transitioned to s'. The posterior action distribution can be used to estimate the ARS, as shown by the following proposition (see the appendix for proof).

Proposition 3. *Let M be a deterministic MDP, and let π be a policy satisfying $\pi(a|s)>0$ for all $s,a\in\mathcal{S}\times\mathcal{A}$. Then, for any two actions $a,\tilde{a}\in\mathcal{A}$ and state $s\in\mathcal{S}$ we have*

$$f(s,a)=f(s,\tilde{a})\iff q^\pi(a|s,f(s,\tilde{a}))\neq 0.$$

Moreover the ARS can be written as

$$\eta^\pi(s,a)=\mathbb{E}_{\tilde{a}\sim\pi(\cdot|s)}\mathbb{1}_{R_q(s,a)}(\tilde{a}), \tag{5}$$

where $R_q(s,a)=\{\tilde{a}\in\mathcal{A}:\tilde{a}\neq a, q^\pi(\tilde{a}|s,f(s,a))>0\}$.

Note the difference in definition of the set R_q in Equation 5 from R in equation Equation 4. For discrete actions, we can equivalently write Equation 5 as

$$\eta^\pi(s,a)=\sum_{\tilde{a}\neq a}\pi(\tilde{a}|s)\mathbb{1}[q^\pi(\tilde{a}|s,f(s,a))>0].$$

We find that η^π is in fact a function of the action posterior support. Given a transition (s,a,s'), approximating q, and in particular its support, is in general a much easier task than identifying $f(s,a)$ (whenever $|\mathcal{S}|\gg|\mathcal{A}|$).

Proposition 3 lets us estimate the ARS using the posterior action distribution q^π. We can efficiently learn the ARS

Algorithm 1 MinRed SAC

Input: Parametric models $Q_\psi, \pi_\theta, q_\phi$

Init: Empty replay buffer $\mathcal{D}$
for $i = 0, 1, \ldots, T$ **do**
  for $t = 0, 1, \ldots, K$ **do**
    Sample $a_t \sim \pi_\theta(\cdot|s_t)$.
    Receive $r(s_t, a_t), s_{t+1} \sim P(\cdot|s_t, a_t)$.
    Add $\{s_t, a_t, r_t, s_{t+1}, \pi_\theta(s_t)\}$ to replay $\mathcal{D}$.
  end for
  for $n = 0, 1, \ldots, N$ **do**
    Sample $(s_i, a_i, r_i, s'_i, \pi_i) \sim D$.
    Update Q_ψ with $r_i + \alpha \frac{\pi_\theta}{\pi_i} \zeta_\phi^{\pi_i}(s_i, a_i, s'_i)$.
    Update policy π_θ according to policy gradient.
    Update posterior q_ϕ according to Equation 7.
  end for
end for

Algorithm 2 MinRed Q-Learning

Input: Parametric models Q_ψ, q_ϕ, threshold $\delta > 0$

Init: Empty replay buffer $\mathcal{D}$
for $i = 0, 1, \ldots, T$ **do**
  for $t = 0, 1, \ldots, K$ **do**
    Select $a_t \in \arg\max_a Q_\psi(s_t, a)$.
    Receive r_t and $s_{t+1} \sim P(\cdot|s_t, a_t)$.
    for $\bar{a} \in \mathcal{A}_\delta(s_t, s_{t+1}) \cup \{a_t\}$ **do**
      Add $\{s_t, \bar{a}, r_t, s_{t+1}\}$ to replay $\mathcal{D}$.
    end for
  end for
  for $n = 0, 1, \ldots, N$ **do**
    Update Q_ψ with reward $r(s, a)$.
    Update posterior q_ϕ according to Equation 7.
  end for
end for

using a backward model, sidestepping the need to learn a complex forward model of the environment. In fact, learning the exact posterior is not needed, but rather, only identifying its support is required to calculate the ARS. Our reliance on a learned model is thus minimized, allowing us to enjoy a near model-free setup.

4.2 STOCHASTIC CASE

In the general stochastic case, estimating g_π amounts to estimating both the model entropy as well as the action redundancy terms. Estimating the model entropy in non-tabular settings is an open problem closely related to out-of-distribution detection and uncertainty estimation. As such, estimating it can amount to large errors unless carefully designed for the specific domain at hand. In contrast, the action redundancy term is dependent on π at state s and can be greedily and efficiently optimized. We therefore focus on action redundancy, noting estimation techniques for model entropy as an orthogonal direction for future work.

Recall $q^\pi(a|s, s')$, the posterior action distribution defined in the previous subsection. Similar to the deterministic setting, we are interested in expressing the action redundancy term in Equation 3 using q^π, mitigating its estimation complexity. We define the action redundancy ratio (ARR) as follows.

Definition 3. *Let π be a policy satisfying $\pi(a|s) > 0$ for all $s, a \in \mathcal{S} \times \mathcal{A}$. The action redundancy ratio (ARR) of action a at state s transitioning to state s' is defined by*

$$\zeta^\pi(s, a, s') = \log \frac{q^\pi(a|s, s')}{\pi(a|s)}$$

The ARR measures the overall dependence of action a at state s executed by π and its corresponding next state s'. In fact, the ARR is defined by the pointwise mutual information[2] (PMI) of $a \sim \pi(\cdot|s)$ and $s' \sim P(\cdot|s, \pi)$ as

$$\zeta^\pi(s, a, s') = \text{PMI}(a, s' \mid s, \pi) = \log \frac{P(a, s'|s, \pi)}{\pi(a|s)P(s'|s, \pi)}.$$

The ARR is also closely related to the ARS (see Section 4.1), as stated by the following corrolary.

Corollary 2. *Let M be a deterministic MDP. Then*

$$\zeta^\pi(s, a, s') = \begin{cases} -\log\left(\pi(a|s) + \eta^\pi(s, a)\right) & , s' = f(s, a) \\ 0 & , o.w. \end{cases}$$

As such, the ARR generalized the ARS for stochastic MDPs. As before, the ARR requires estimating the action posterior distribution, q^π, a generally easier task than estimating $P(s'|s, a)$. The following proposition relates the ARS to g_π (see the appendix for proof).

Proposition 4. *Let π be a policy satisfying $\pi(a|s) > 0$ for all $s, a \in \mathcal{S} \times \mathcal{A}$. Then*

$$g_\pi(s, a) = \mathcal{H}(s'|s, a) + \mathbb{E}_{s' \sim P(\cdot|s,a)} \zeta^\pi(s, a, s') \qquad (6)$$

The above proposition lets us estimate g_π using the expected ARS over next state transitions. While estimating the expectation in Equation 6 requires sampling multiple transitions from $s' \sim P(\cdot|s, a)$, we can utilize historical information to overcome this need (e.g., by sampling from a replay buffer).

In the next section we propose to leverage an estimate of the action posterior q^π for minimizing redundancy over actions in Actor-Critic and Q-learning variants. We then experimentally show in Section 6 this approach is beneficial, and evidently crucial in domains in which action-redundancy is present.

[2]The pointwise mutual information of outcomes belonging to random variables X, Y is defined by $\text{PMI}(x, y) = \log \frac{P(x,y)}{P(x)P(y)}$.

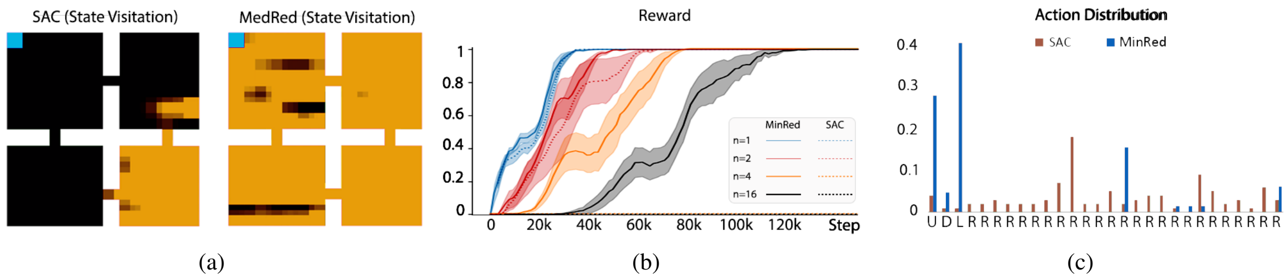


Figure 3: **Synthetic Action Redundancy**. A four-room environment in which an agent is initialized at the bottom right corner and the goal is always located at the top left. The action space contains n redundant “Right" actions. **(a)** A heatmap of the visitation locations of maximum entropy (left) and minimum redundancy (right) agents. **(b)** Comparison of SAC (dotted) and MinRed (solid) agents for different values of n. As n increases, the SAC agent fails to reach the goal. **(c)** Action selection histogram for $n = 35$ shows that the policy practiced by the MinRed agent is far from uniform.

5 ALGORITHMS FOR MINIMIZING ACTION REDUNDANCY

In the previous sections we defined transition entropy and showed its relation to action redundancy (see summary in Table 1). We analyzed action redundancy in both deterministic as well as stochastic MDPs, based relations to the ARS and ARR, and proposed to estimate them by modeling the action posterior $q^\pi(a|s,s')$. This allows us to bypass the need to construct a forward model of the environment and derive efficient algorithms for minimizing action redundancy, as we show next.

Estimation the action posterior. We construct a parametric model for the posterior action distribution q^π. Specifically, given a set of past interactions with the environment, $D = \{s^{(i)}, a^{(i)}, s'^{(i)}\}_{i=1}^N$, the maximum likelihood estimator, ϕ^*, of $q_\phi(a|s,s')$ is given by:

$$\phi^* \in \arg\max_\phi \sum_{i=1}^N \log q_\phi(a^{(i)}|s^{(i)}, s'^{(i)}) \tag{7}$$

We optimize Equation 7 by minimizing the cross entropy in a standard supervised-learning fashion.

5.1 MINRED SAC

We begin by deriving an off-policy algorithm which leverages action redundancy in a MaxEnt Actor-Critic framework. We propose MinRed SAC, shown in Algorithm 1, which uses a replay buffer to store transitions and updates a critic w.r.t. an approximate ARR bonus. The ARR is estimated by our parametric model q_ϕ (Equation 7) and updated with the critic. Alternatively, the ARR can be replaced by the ARS for deterministic domains. As the ARR depends on π, importance weights can be used to correct for biases in the updates. Specifically, the policy probabilities π_i ared stored with the transitions in the replay buffer and correct for the difference using the importance ratio $\frac{\pi}{\pi_i}$.

5.2 MINRED Q-LEARNING

A drawback of MinRed Actor Critic is its need of off-policy corrections for the shaped reward, as the ARR and ARS are updated by an old policy sampled from the replay buffer. To overcome this problem, we propose an alternative approach for minimizing action redundancy. Our approach, MinRed Q-Learning, shown in Algorithm 2, uses an effective reduced action set to fight over-exploration of redundant actions. This method can be easily incorporated into any discrete algorithm (not necessarily MaxEnt).

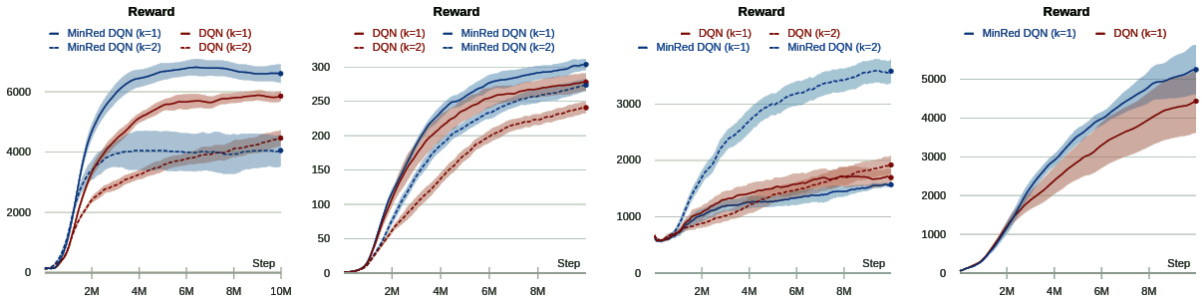
Motivated by the dependence of the ARR and ARS on the action posterior q^π, we define the δ-Redundant Action Set as follows. Given a state, next-state pair (s, s') and a scalar $\delta > 0$, we define

$$\mathcal{A}_\delta(s,s') = \{a \in \mathcal{A} \mid q^\pi(a|s,s') > \delta\}\,.$$

MinRed Q-Learning leverages the δ-redundant action set $\mathcal{A}_\delta$ to accelerate exploration in the face of action redundancy. Specifically, whenever a new transition (s, a, r, s') is sampled it is added to a replay buffer $\mathcal{D}$. Additionally, all equivalent actions (conditioned on the sensitivity threshold parameter δ), are also added to $\mathcal{D}$ i.e., $\{(s, \bar{a}, r, s) \mid \bar{a} \in \mathcal{A}_\delta(s, s')\}$. These synthetic transitions induce “grouped exploration" of redundant actions, effectively reducing the dimensionality of the action space.

6 EXPERIMENTS

In this section we empirically evaluate our proposed algorithms on a series of domains. Specifically, we first construct a synthetic grid-like environment to analyze the effect of explicit action redundancy (i.e., duplication of actions) on performance. We then run experiments on standard discrete (Atari) and continuous control (Mujoco) benchmarks, showing contemporary algorithms suffer from the inherent redundancy.



(a) DemonAttack (b) Breakout (c) AirRaid (d) Seaquest

Figure 4: Results of MinRed DQN on Atari2600 benchmarks. We synthesized an action space consisting of all action sequences of length $k = 2$ (i.e., cardinality $|\mathcal{A}|^k$). For Seaquest, the original action space (18 actions) was used, as we expect large action spaces to already contain redundancy. See Appendix B for further details.

Synthetic Action Redundancy. We wish to determine how well our method handles redundancy in a controlled synthetic environment. As such, we constructed a four-room environment in which the initial position of the agent and the goal were at two opposite corners (see Figure 3). The agent earns a reward of 1 at the goal and 0 elsewhere. An episode ends whenever the agent achieves the goal or attempts 100 steps. We constructed a synthetic redundant action space $\mathcal{A} = \{\texttt{Top}, \texttt{Left}, \texttt{Bottom}, \overbrace{\texttt{Right}, \ldots, \texttt{Right}}^{n}\}$, for which the action `Right` was repeated n times. To reach the goal on the far left and receive positive reward, the agent must counteract redundancy in actions pulling her to the right.

We compared MinRed SAC and vanilla SAC for various values for n (the number of repeats). Figure 3 depicts the reward of the agents as well as a visual representation of the areas visited by the agents. While SAC roams close to the origin, the MedRed agent reaches the goal, able to counteract action redundancy.

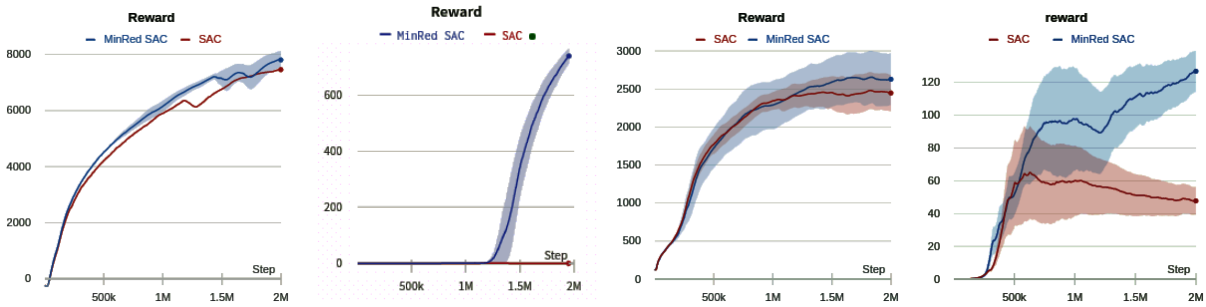
Redundant Macro Actions. Macro actions are commonly used to improve performance of RL agents. Macro actions are sequences of primitive actions, which, when chosen properly, may greatly shorten the effective horizon of the problem. However, designing macro-actions is a difficult task that is accomplished either through hierarchical reinforcement learning setups [Vezhnevets et al., 2017], or transfer learning algorithms [Konidaris and Barto, 2007]. Unfortunately, using all action sequences of length k results in an action space of cardinality $|\mathcal{A}|^k$. Such enlargement of the action space may be detrimental to the exploration process. Instead, we propose to leverage action redundancy to distinguish between the principal macro actions of the problem at hand.

We tested the MinRed Q-Learning agent on a series of games from the Atari2600 Arcade Learning Environment. We constructed a combinatorial action space on top of the original action space comprising of all action sequences (macro actions) of length k. While exploring such a large action space may be extremely challenging, through action redundancy the agent can potentially benefit from a short planning horizon at the cost of a moderately-sized action space. For environments with a small action space we extended the action space to action sequences of length $k = 2$. For environments with a large amount of actions (e.g., Seaquest) we used the original action space, as redundancy was already evident.

Results, as presented in Figure 4, show improvement in performance when redundancy is accounted for. Significant improvement is evident when the number of actions increases due to our action sequence augmentation. Particularly, in various enviornments, when introducing action sequences, vanilla DQN did not see an increase in performance whereas our method showed either better or significantly superior performance. These results suggest that action redundancy is an intrinsic component of the reinforcement learning framework (in particular large action spaces), and overcoming it can greatly improve performance.

Continuous Control. We compared SAC against MinRed SAC on a series of continuous control benchmarks in Mujoco [Todorov et al., 2012]. Specifically, we constructed dense and sparse reward variants of the standard OpenAI-Gym Mujoco environments. In the standard, dense reward setting, the agent is rewarded at every time t according to its absolute position, e.g., how much a humanoid has progressed in the x direction. The agent is also penalized for taking actions with large magnitude. Conversely, in the sparse reward setting, the agent is only rewarded once it has traversed at least a *threshold* distance, after which the original (dense) reward is given.

In general, sparse reward settings are highly sensitive to efficient exploration. As such, we expect to observe significant improvement in the sparse reward variants. On the contrary, well designed dense rewards are less prone to exploration



(a) Halfcheetah (Dense) (b) Halfcheetah (Sparse) (c) Humanoid (Dense) (d) Humanoid (Sparse)

Figure 5: Results of MinRed SAC compared to SAC show an increase in performance in sparse reward domains, whereas dense reward domains remain mostly unaffected.

issues, but rather robust control. Indeed, results depicted in Figure 5 show a significant improvement of MinRed SAC over SAC on sparse reward benchmarks, whereas performance is mostly unaffected in the dense setup. These results suggests that action redundancy is an inherent artifact of continuous control domains, obscured by concurrent benchmarks which utilize dense reward setups.

7 RELATED WORK

Maximum Entropy. Over the past decade MaxEnt algorithms have become a dominant solution for hard exploration environments [Ziebart et al., 2008]. MaxEnt versions of popular RL algorithms were introduced [Haarnoja et al., 2017], often outperforming their baseline counterparts [Haarnoja et al., 2018].

Orthogonal research has also focused on state-based exploration. Based on a planning oracle that finds the best policy under a well-specified reward signal, Hazan et al. [2019] showed a provably efficient method that can achieve near-optimal state-entropy. On the other hand, maximizing a lower bound on the entropy of the steady-state distribution both removes the need for a planning oracle and allows learning fast mixing policies [Mutti and Restelli, 2020]. However, like other state-entropy methods that are oblivious to the geometry of states, this method is theoretically justified only for discrete states spaces. An adaptation to continuous domains requires using Geometric Entropy Maximisation (GEM) methods that respect the geometry of the Shannon entropy [Guo et al., 2021] or alternative entropy measures [Mensch et al., 2019].

Reaching diverse sets of states is also beneficial when learning‘ skills [Sutton et al., 1999]. This is possible by using methods that promote diversity between trajectories [Lim and Auer, 2012, Gajane et al., 2019, Gregor et al., 2016], or techniques that maximize the diversity within a single trajectory [Badia et al., 2020]. The idea of maximum state entropy was also investigated in the context of multi-task RL [Zhang and Yang, 2017]. Training goal-dependent policies [Andrychowicz et al., 2017] to accomplish a diverse set of goals is possible by maximizing an information-theoretic surrogate function [Pong et al., 2019].

Large Action Spaces. Reasoning in environments with a large number of discrete actions or high dimensionality of continuous actions is a central obstacle for reinforcement learning. Various methods have been proposed to mitigate this difficulty, including action elimination [Even-Dar et al., 2003, Zahavy et al., 2018], action embeddings [Dulac-Arnold et al., 2012, Tennenholtz and Mannor, 2019, Chandak et al., 2019], and factorizations [Pazis and Parr, 2011, Dulac-Arnold et al., 2012].

Our work provides an alternative perspective on the problem, suggesting that action redudnancy is present in problems of high dimensional action spaces and hard exploration. As such minimizing redundancy is a principal component for solving reinforcement learning tasks. Our approach reduces the action space effective dimensionality, thus allowing for more efficient learning. It can be readily applied to any reinforcement learning algorithm, as illustrated by our proposed MinRed methods (i.e., Algorithms 1 and 2).

8 CONCLUSION

This paper identifies and addresses the fundamental weakness of action redundancy in view of maximizing entropy. Through transition entropy, we isolated the problem and identified action redundancy as a key factor in increasing overall transition entropy. Our approach can be viewed as a MaxEnt framework in a modified action space that softly combines actions according to their induced transitions. We constructed two variants of contemporary algorithms to reduce action redundancy, showing signficantly improved performance on various benchmarks, and suggesting that action redundancy is indeed an inherent problem in RL.

Author Contributions

Nir Baram and Guy Tennenholtz contributed equally to this paper.

Acknowledgements

This research was partially supported by the ISF under contract 2199/20.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt, et al. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020.

Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In *International Conference on Machine Learning*, pages 941–950. PMLR, 2019.

Daniela Pucci De Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations research*, 51(6):850–865, 2003.

Gabriel Dulac-Arnold, Ludovic Denoyer, Philippe Preux, and Patrick Gallinari. Fast reinforcement learning with large action sets using error-correcting output codes for mdp factorization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 180–194. Springer, 2012.

Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for reinforcement learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 162–169, 2003.

Pratik Gajane, Ronald Ortner, Peter Auer, and Csaba Szepesvari. Autonomous exploration for navigating in non-stationary cmps. *arXiv preprint arXiv:1910.08446*, 2019.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

Zhaohan Daniel Guo, Mohammad Gheshlagi Azar, Alaa Saade, Shantanu Thakoor, Bilal Piot, Bernardo Avila Pires, Michal Valko, Thomas Mesnard, Tor Lattimore, and Rémi Munos. Geometric entropic exploration. *arXiv preprint arXiv:2101.02055*, 2021.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691. PMLR, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

George Dimitri Konidaris and Andrew G Barto. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, volume 7, pages 895–900, 2007.

Shiau Hong Lim and Peter Auer. Autonomous exploration for navigating in mdps. In *Conference on Learning Theory*, pages 40–1, 2012.

Arthur Mensch, Mathieu Blondel, and Gabriel Peyré. Geometric losses for distributional learning. In *International Conference on Machine Learning*, pages 4516–4525. PMLR, 2019.

Mirco Mutti and Marcello Restelli. An intrinsically-motivated approach for learning highly exploring and fast mixing policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5232–5239, 2020.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *arXiv preprint arXiv:1702.08892*, 2017.

Jason Pazis and Ronald Parr. Generalized value functions for large action sets. In *ICML*, 2011.

Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 7750–7761. PMLR, 2020.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Lorenzo Pratissoli. Task-agnostic exploration via maximum state entropy policy optimization. 2020.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3. `https://github.com/DLR-RM/stable-baselines3`, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Guy Tennenholtz and Shie Mannor. The natural language of actions. In *International Conference on Machine Learning*, pages 6196–6205. PMLR, 2019.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Peng-Yeng Yin. Maximum entropy-based optimal threshold selection using deterministic reinforcement learning with controlled randomization. *Signal Processing*, 82(7):993–1006, 2002.

Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31: 3562–3573, 2018.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

A MISSING PROOFS

We will use the following notation for our proofs.

$$\mathcal{F}_\pi(s) = -\mathbb{E}_{s'\sim P(\cdot|s,\pi)} \log P(\tilde{s}|s,\pi).$$

Proof of Proposition 1.

We have that

$$\begin{aligned}
\mathbb{F}(s,\pi) &= -\sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t\sim\rho_\pi} \mathbb{E}_{s'\sim P(\cdot|s_t,\pi)} \left[\log P(s'|s_t,\pi)\Big| s_0 = s\right] \\
&= -\mathbb{E}_{s'\sim P(\cdot|s,\pi)} \log P(\tilde{s}|s,\pi) - \sum_{t=1}^{\infty} \gamma^t \mathbb{E}_{s_t\sim\rho_\pi} \mathbb{E}_{s'\sim P(\cdot|s_t,\pi)} \left[\log P(s'|s_t,\pi)\Big| s_0 = s\right] \\
&= \mathcal{F}_\pi(s) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,\pi)} \mathbb{F}(s',\pi)
\end{aligned}$$

Recall that $P(\cdot|s,\pi) = \mathbb{E}_{a\sim\pi(s)} P(s'|s,a)$. Then

$$\mathbb{E}_{s'\sim P(\cdot|s,\pi)} \mathbb{F}(s',\pi) = \mathbb{E}_{a\sim\pi(s)} \mathbb{E}_{s'\sim P(\cdot|s,a)} \mathbb{F}(s',\pi).$$

Similarly,

$$\mathcal{F}_\pi(s) = -\mathbb{E}_{a\sim\pi(s)} \mathbb{E}_{s'\sim P(\cdot|s,a)} \log P(s'|s,\pi) = \mathbb{E}_{a\sim\pi(s)} g_\pi(s,a)$$

where,

$$g_\pi(s,a) = -\mathbb{E}_{s'\sim P(\cdot|s,a)} \log P(s'|s,\pi).$$

This completes the proof.

Proof of Proposition 2.

We have that

$$\begin{aligned}
g_\pi(s,a) &= -\mathbb{E}_{s'\sim P(\cdot|s,a)} \log P(s'|s,\pi) \\
&= -\mathbb{1}_{s'=f(s,a)} \log P(s'|s,\pi) \\
&= -\mathbb{1}_{s'=f(s,a)} \log \mathbb{E}_{\tilde{a}\sim\pi(s)} P(s'|s,\tilde{a}) \\
&= -\log \mathbb{E}_{\tilde{a}\sim\pi(s)} P(s' = f(s,a)|s,\tilde{a}) \\
&= -\log \left(\mathbb{E}_{\tilde{a}\sim\pi(\cdot|s)} \left[\pi(a|s) + \mathbb{1}_{R(s,a)}(\tilde{a})\right]\right),
\end{aligned}$$

where $R(s,a) = \{\tilde{a} \in \mathcal{A} : \tilde{a} \neq a, f(s,\tilde{a}) = f(s,a)\}$. This completes the proof as $\eta^\pi(s,a) = \mathbb{E}_{\tilde{a}\sim\pi(\cdot|s)} \mathbb{1}_{R(s,a)}(\tilde{a})$.

Proof of Proposition 3.

$f(s,a) = f(s,\tilde{a}) \iff q^\pi(a|s, f(s,\tilde{a})) \neq 0.$

Let $a, \tilde{a} \in \mathcal{A}$ and $s \in \mathcal{S}$. If $f(s,a) = f(s,\tilde{a})$ then trivially $q^\pi(a|s, f(s,\tilde{a})) > 0$. Conversely, if $q^\pi(a|s, f(s,\tilde{a})) \neq 0$ then using the fact that the transitions are deterministic and that $\pi_b(a|s) > 0$ for all $a \in \mathcal{A}$ we get that

$$\begin{aligned}
q^\pi(a|s, f(s,\tilde{a})) &= \frac{P^\pi(f(s,\tilde{a}), a|s)}{P^\pi(f(s,\tilde{a})|s)} \\
&= \frac{P(f(s,\tilde{a})|s,a)\pi(a|s)}{P^\pi(f(s,\tilde{a})|s)}.
\end{aligned}$$

By definition $P(f(s,a)|s,a) = 1$, therefore $q^\pi(a|s, f(s,\tilde{a})) \neq 0$ results in

$$0 \neq P(f(s,\tilde{a})|s,a)\pi(a|s) = P(f(s,a)|s,a)\pi(a|s).$$

Since $\pi(a|s) > 0$, the proof is complete.

Proof of Proposition 4.

Calculating action redundancy in stochastic MDPs amounts to measuring the KL divergence

$$D_{KL}\left(P(\cdot|s,a)\,||\,P(\cdot|s;\pi)\right) = \mathbb{E}_{s'\sim P(\cdot|s,a)}\log\left[\frac{P(s'|s,a)}{\mathbb{E}_{\hat{a}\sim\pi(\cdot|s)}P(s'|s,\hat{a})}\right]$$

Applying Bayes Rule yields

$$\begin{aligned}\mathbb{E}_{s'\sim P(\cdot|s,a)}\log\left[\frac{P(s'|s,a)}{\mathbb{E}_{\hat{a}\sim\pi(\cdot|s)}P(s'|s,\hat{a})}\right] &= \mathbb{E}_{s'\sim P(\cdot|s,a)}\log\left[\frac{q^\pi(a|s,s')P^\pi(s'|s)}{\pi(a|s)\mathbb{E}_{\hat{a}\sim\pi(\cdot|s)}\frac{P^\pi(\hat{a}|s,s')P^\pi(s'|s)}{\pi(\hat{a}|s)}}\right]\\ &= \mathbb{E}_{s'\sim P(\cdot|s,a)}\log\left[\frac{q^\pi(a|s,s')}{\pi(a|s)\mathbb{E}_{\hat{a}\sim\pi(\cdot|s)}\frac{P^\pi(\hat{a}|s,s')}{\pi(\hat{a}|s)}}\right]\\ &= \mathbb{E}_{s'\sim P(\cdot|s,a)}\log\left[\frac{q^\pi(a|s,s')}{\pi(a|s)}\right],\end{aligned}$$

where in the last step we used the fact that

$$\mathbb{E}_{\hat{a}\sim\pi(\cdot|s)}\frac{P^\pi(\hat{a}|s,s')}{\pi(\hat{a}|s)} = \int_{\hat{a}}\pi(\hat{a}|s)\frac{P^\pi(\hat{a}|s,s')}{\pi(\hat{a}|s)}d\hat{a} = \int_{\hat{a}}P^\pi(\hat{a}|s,s')d\hat{a} = 1.$$

This completes the proof.

B IMPLEMENTATION DETAILS

Specific Implementation Details. A convolutional neural network is used to implement the policy and the critic. The layer specification of the network is provided in Appendix B. We use an online actor critic algorithm as a baseline [Schulman et al., 2017]. The baseline is modified to correct for action redundancy using Table 1. We experiment with varying lengths of action sequences, k and report the value used in the figure.

We run each task for 3 million steps and report an average of 5 seeds. We use the Adam optimizer [Kingma and Ba, 2014] with a learning rate of 0.0003 and a batch size of 256 to train the algorithms. After collecting experience, the algorithm applies 4 consecutive stochastic gradient steps and clip the advantage as suggested in Schulman et al. [2017]. All other hyper-parameters are set to their default values as implemented in the Proximal-Policy-Optimization algorithm as part of the stable-baselines framework [Raffin et al., 2019].

B.1 MINRED Q-LEARNING

This section describes the implementation details and hyper-parameters used to train the MinRed DQN agent as well as other results.

Network Architecture. The input to the policy is a concatenation of the last four frames, each of which is of the size 84×84. We implement the Q function using a neural network with the following specification.

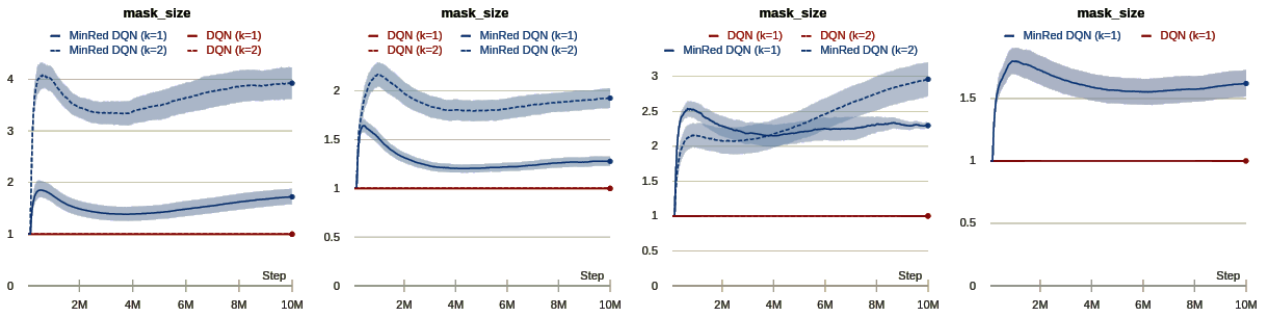
Hyper-Parameters. The hyperparameters used to train the MinRed DQN agents are presented in Table 3.

Action Space Size. Table 4 describes the action space size used to train the MinRed DQN agents.

Redundancy Size. Figure 6 shows the average mask size of the MinRed DQN agent. That is, the number of synthetic transitions loaded to the replay buffer. Note that the mask-size of the baseline method is identically 1.

Layer	Size	Comments
Conv2D	$4 \times 32 \times 8 \times 8$	input channels $\times$ ouput channels $\times$ kernel size $\times$ kernel size
ReLU	–	
Conv2D	$32 \times 64 \times 4 \times 4$	
ReLU	–	
Conv2D	$64 \times 64 \times 3 \times 3$	
ReLU	–	
Flatten	–	
Linear	64	
ReLU	–	
Linear	64	
ReLU	–	
Linear	$\lvert A \rvert$	

Table 2: MinRed DQN agent. A detailed description of the neural network function approximation architecture.



(a) DemonAttack (b) Breakout (c) AirRaid (d) Seaquest

Figure 6: **Redundancy Size.** The graph shows the average number of synthetic transitions created per single transition sampled from a real environment.

B.2 MINIMUM REDUNDANCY SOFT ACTOR CRITIC

This section describes the implementation details and hyper-parameters used to train the minimum redundancy agent as well as other results.

Network Architecture. The input to the policy is a vector indicating the positions and velocities of the joints of the agent. Using a neural network with the following specification, we implement both the soft Q function and the policy. The input to the Q-function is a concatenation of s and a and the output is a single value corresponding to $Q(s, a)$. The policy network accepts a state s as input and produces two vectors of size $|A|$. The two outputs represent the first two moments of a Gaussian distribution, from which an action is sampled. The action in this case, represents the forces to apply to each joint. The action is squashed to the range of $[-1, 1]$ using a `Tanh` layer. Table 5 lists the parametric layers of the neural network function approximation.

Hyper-Parameters. The hyperparameters used by the MinRed agents are shown in Table 6.

Entropy Coefficients. Table 7 describes the entropy and action redundancy coefficients α, and λ respectively used for different environments. As a thumb rule, we set $\lambda = 0.01\alpha$.

Name	Value	Comments
Batch size	32	
Learning rate	0.0001	
Buffer size	$150,000$	
Total timesteps	$10,000,000$	
Exploration initial value (ϵ-greedy)	1.	
Exploration final value (ϵ-greedy)	0.05	
Learning starts	$50,000$	iteration to start learning
Regularization starts	$150,000$	iteration to start loading synthetic transitions
γ	0.99	Discount factor
τ	1	target network Polyak averaging
Target network update interval	$1,000$	

Table 3: Hyper-parameters used to train the MinRed DQN agent. A detailed description of the hyper-parameters used to train the MinRed DQN agents.

Environment	$\|A\|$ (original)	$\|\tilde{A}\|$ (extended)
Breakout	4	16
DemonAttack	6	36
AirRaid	6	36
NameThisGame	6	36

Table 4: Action Space Size. A detailed description of the action space size used by the MinRed DQN agents.

Layer	Size	Comments
Linear	256	
ReLU	–	
Linear	256	
ReLU	–	
Linear	$\|A\|$	action space size

Table 5: MinRed DQN agent. A detailed description of the neural network function approximation architecture.

Name	Value	Comments
Batch size	256	
Learning rate	0.0003	
Buffer size	$100,000$	
Total timesteps	$2,000,000$	
Learning starts	$10,000$	iteration to start learning
train frequency	1000	Update the model every train-freq steps
gradient-steps	1000	How many gradient steps to do after each rollout
γ	0.99	Discount factor
τ	0.05	target network Polyak averaging
Target network update interval	1	

Table 6: Hyper-parameters used to train the Minimum Redundancy agents. A detailed description of the hyper-parameters used to train the minimum redundancy agents.

Environment	α (entropy coefficient)	λ (action redundancy)	sparsity threshold
Ant-v2	0.1	0.001	–
HalfCheetah-v2	0.05	0.05	–
Humanoid-v2	0.1	0.001	–
HumanoidStandUp-v2	1.0	0.01	–
Walker-v2	0.1	0.1	–
SparseAnt-v2	0.05	0.005	15
SparseHalfCheetah-v2	0.05	0.005	19.5
SparseHumanoid-v2	0.05	0.0005	0.6
SparseHumanoidStandUp-v2	0.05	0.005	–
SparseWalker-v2	0.1	0.001	15

Table 7: Entropy Coefficients. The entropy coefficient α is set by running the baseline method, soft-actor-critic, with a tenable entropy coefficient and taking the average value of the last 100k iterations. The action redundancy coefficient, λ is taken to be $\lambda = 0.01\alpha$ unless otherwise mentioned.