# CASH: Cache Alignment with Specified Horizons

**Jacob Adamczyk**[1], **Josiah C. Kratz**[2]

[1]Department of Physics, University of Massachusetts Boston,
The NSF AI Institute for Artificial Intelligence and Fundamental Interactions
[2]Computational Biology Department, Carnegie Mellon University

## Abstract

In this work, we offer CASH (Cache Alignment with Specified Horizons), a new reinforcement learning algorithm to effectively leverage prior knowledge. In this framework, previously computed policies are stored in a cache and compared against online rollouts to determine similarity based on a new heuristic derived from the Kendall tau distance. We prove error bounds on the estimated value function which allows for an optimizable rollout length based on a cost-accuracy tradeoff. We then introduce a second horizon to specify execution time of the most similar cached policy. We show that even in the case of adversarial caches, the algorithm performs no worse than standard $Q$-learning. This preliminary work provides a new perspective on the intersection of transfer learning and model-based reinforcement learning.

## Introduction

The application of reinforcement learning (RL) to sequential decision-making tasks has seen significant growth in recent years, showcasing its capacity to uncover innovative solutions in complex and unfamiliar environments (Schrittwieser et al. 2020). In RL, model-based methods — which focus on learning or utilizing a model of the environment's dynamics — have demonstrated greater sample efficiency compared to model-free approaches, especially in scenarios with constrained interaction budgets (Hafner et al. 2023). This efficiency arises from the ability of model-based methods to perform planning over imagined trajectories, employing techniques like Monte Carlo tree search to evaluate and refine policies. Unfortunately, these methods incur higher computational cost and depend heavily on the precision of the learned dynamics model.

In the traditional RL paradigm, solutions to previously learned tasks are often ignored or discarded when tackling a new task. However, in many settings, new tasks are frequently related to or share features with earlier ones. Consequently, a significant amount of useful information is discarded and relearned, causing inefficiencies in training. This has motivated previous work to incorporate learned solutions into training. However, these techniques usually require some understanding of how the current task is related to prior tasks (e.g. in terms of their reward functions), which can limit the utility of such approaches. Furthermore, if past tasks do not contain useful information to solve the new task, their incorporation can potentially distract the agent and impede performance instead of improving it, leading to "negative transfer" (Taylor and Stone 2009; Rusu et al. 2016). As a result, it remains an open question of how to best incorporate previously learned policies into the training for new tasks.

To remedy this inefficiency, we propose a modified training approach which leverages previously learned policies for "source" tasks to guide training for the new "target" task in the same environment. In this approach, short parallelized rollouts are performed for each action. The optimal trajectory is then selected based on the accumulated reward, and then compared with the stored policy cache to identify the most similar corresponding stored policy. We propose a new method to measure the distance between value functions, allowing solutions in the cache to be aligned with the recently experienced online data. Additionally, we propose to include the current solution estimate in the cache itself. Critically, if the cached tasks do not aid in performance, they are ignored, allowing this algorithm to succeed even in the presence of an adversarial cache.

In the following sections, we review related prior work and essential background material before introducing our proposed algorithm and presenting our initial theoretical analysis.

## Prior Work

In previous work, many methods have been proposed to use a set of solutions to transfer knowledge to the agent (cf. the reviews (Taylor and Stone 2009; Zhu et al. 2023)). Of primary interest in our approach is the notion of a policy cache (a set of stored solutions) which can be used to determine a specific source to transfer to the target task. This setup exists in many forms in the literature, for example in successor features, compositionality, and hierarchical structures. Within the successor feature (Barreto et al. 2017a) framework, a "policy cache" is used to form zero-shot solutions of target tasks within the convex hull of the feature set. Continuing this line of work, (Nemecek and Parr 2021) derive value-function bounds to determine additions to the cache. Another line of work has focused on combining multiple "subtasks" through composition, when a direct relationship over

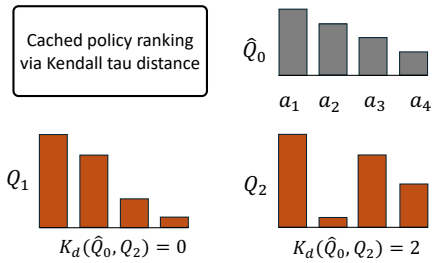



Figure 1: Illustration of similarity ranking of cached policies via Kendall tau distance. Even though both cached policies ($Q_1$, $Q_2$) predict the same optimal action ($a_1$), $Q_1$ is much more similar to the online estimate ($\hat{Q}_0$) when considering the ranking of all action pairs. This is reflected in the smaller Kendall tau distance ($K_d$).

reward functions is known (e.g. linear or Boolean compositions) (Haarnoja et al. 2018; Peng et al. 2019; Hunt et al. 2019; Tasse, James, and Rosman 2020; Saxe, Earle, and Rosman 2017; Adamczyk et al. 2023).

Recently, a distinct notion of "task distance" has emerged, with a pseudo-metric ("EPIC") defined in the space of reward functions (Wulfe et al. 2022; Skalse et al. 2023). However, "similarity" can instead be defined in the space of value functions, which may be particularly useful in the transfer learning setting, where the source solutions have already been computed. Casting the notion of similarity to the space of solutions leads us to a new method for choosing which of the previous source solutions best correspond to the target problem. Before introducing our algorithm, we present some necessary background material.

## Background

### Reinforcement Learning

In this section we briefly introduce the reinforcement learning framework. The reader is directed to standard texts for more information, such as (Sutton and Barto 2018). We consider the RL problem for discrete state-action spaces, and model it as a Markov Decision Process (MDP), represented by the information $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ with state space $\mathcal{S}$; action space $\mathcal{A}$; potentially stochastic transition function (dynamics) $p : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$; bounded, real reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$; and discount factor $\gamma \in [0, 1)$.

The primary objective of RL is to maximize the total discounted reward received under a control policy $\pi$. Specifically, we wish to find $\pi^*$ which maximizes the following expectation, where trajectories $\tau$ are sampled from the Markov chain induced by the fixed dynamics and optimized policy:

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{\tau \sim p, \pi} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \quad (1)$$

In the present work, we consider value-based RL methods, where the solution to the RL problem is equivalently defined by its optimal action-value function, $Q^*(s, a)$. In this case, the optimal policy $\pi^*(a|s)$ is derived from $Q^*$ through a greedy maximization over actions. We choose this problem formulation to allow us to utilize the $Q$-function to calculate the relative ordering of actions, thus defining a distance function between $Q$-functions. The optimal value function can be obtained by iterating the following recursive Bellman equation until convergence:

$$Q^*(s, a) = r(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim p(\cdot|s,a)} \max_{a'} Q^*(s', a'). \quad (2)$$

In the tabular setting, the exact Bellman operator can be applied to any (bounded) initial value until convergence of the above equation. In the function approximation setting (e.g. deep RL), the $Q$ table is replaced by a parameterized function approximator, $Q_\theta$, and the temporal difference (TD) loss calculated over mini-batches is minimized instead. To simplify the discussion, we will neglect the details of precisely how the $Q$-function is obtained, and we focus on generic value functions alongside their approximations. We note that although this initial investigation is presented in the "standard" $Q$-learning setting, extensions to maximum entropy setting may also be possible.

In order to guide source selection, we introduce a novel metric to measure the similarity between tasks. This metric is based on the intuition that similar tasks should have solutions (optimal policies) which agree. This agreement is measured across the entire action space, as opposed to only checking agreement of the optimal action. As solutions, $Q$-values encode a ranking between any two actions. Thus, if two solutions agree, their relative rankings should also agree. This intuition is encapsulated by the following well-established definition, stemming from the bio-informatics literature (Kendall 1938):

**Definition 1** (Kendall Tau Distance Between Value Functions)**.** The (un-normalized) Kendall tau distance between two policies is defined as

$$K_d(Q_1, Q_2; s) = \sum_{\{i,j\} \in P, i<j} \bar{K}_{ij}(Q_1(s, \cdot), Q_2(s, \cdot)), \quad (3)$$

where $P$ is the set of all (unordered) pairs of actions; $\bar{K}_{ij} = 0$ if the rankings $Q_1(s, i) > Q_1(s, j)$ and $Q_2(s, i) > Q_2(s, j)$ agree, and otherwise takes on the value 1 (e.g. if $Q_1(s, i) > Q_1(s, j)$ and $Q_2(s, i) < Q_2(s, j)$).

Here we make the state explicit, but we drop this indexing in subsequent sections for notational clarity. In the following, we propose an algorithm that aligns prior solutions in the cache to the observed data stream, based on this new notion of distance. This definition marks an attempt to capture higher order effects, beyond comparing only the most optimal action between policies (Fig. 1).

Comparison of the action rankings rather than directly comparing $Q$-values has an important advantage in measuring similarity: the Kendall tau distance is invariant to any potential-based reward shaping (Ng, Harada, and Russell 1999) or positive scaling in the reward function. Instead, it directly encodes preferences at the level of action pairs. This definition also allows comparisons between actions in the discrete action case, where directly measuring distance between actions would not be well-defined, as there is (typically) no metric over discrete action spaces (e.g. measuring

the distance between the "LEFT" and "FIRE" actions in the Atari suite (Bellemare et al. 2013)).

**Lemma 1.** *The Kendall tau distance is invariant to potential-based reward shaping (PBRS) (Ng, Harada, and Russell 1999) and positive scaling of the reward function. Let $\alpha > 0$ and $\Phi : \mathcal{S} \to \mathbb{R}$ represent a scale factor and potential function, respectively. Then, consider a cached MDP with shaped reward function $\widetilde{r}_c(s,a) = \alpha\, r_c(s,a) + \gamma\Phi(s') - \Phi(s)$. Denoting the corresponding optimal Q-function as $Q_c \to \widetilde{Q}_c$, we have:*

$$K_d(Q_0, Q_c) = K_d(Q_0, \widetilde{Q}_c). \tag{4}$$

The proof is straightforward given the results of PBRS in (Ng, Harada, and Russell 1999), but is given in the Appendix for completeness. In comparison to the EPIC pseudometric (Gleave et al. 2020), the Kendall-tau distance is a metric (in value space), and its calculation is simpler (does not require samples), while still being equivalent under operations preserving the optimal policy. However, it does not yet directly lead to bounds on regret under transfer. In future work, we foresee the study of such bounds to be a useful step forward.

## CASH Algorithm Description

Next, we describe the CASH algorithm in detail. Denote the "online" value function (being trained on the current task) as $Q_0$ and the current state of the agent as $s_t$. To compute a refined estimate of the online value function, $Q_0^*$, we perform a (parallelized) rollout of the greedy policy derived from $Q_0$ for all possible initial actions. Thus, we consider $|\mathcal{A}|$ trajectories, each for a pre-specified length of $n^*$ steps (we discuss the calculation for choosing the value of $n^*$ shortly).

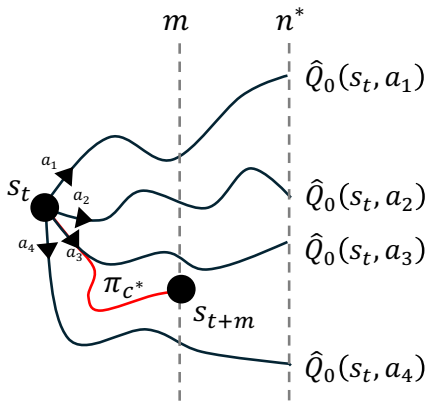
An estimate of the $Q$-function:

$$\widehat{Q}_0(s_t, a) = \sum_{i=0}^{n^*} \gamma^t r_{t+i} + \gamma^{n^*+1} V_0(s_{n^*+1}) \;, \tag{5}$$

at state $s_t$ is then given by the $n^*$-step return and bootstrapped next value. This estimate allows the quality of each action at time $t$ ($a_t$) to be ranked. The most similar cached policy, defined by the minimum Kendall tau distance between the ranked lists of actions, is then identified and used to determine action selection for the next $m$ steps (Fig. 2). Critically, the greedy policy derived from $Q_0$ is also included in the cache, so the algorithm is guaranteed not to perform worse than naïve $Q$-learning. This procedure is then repeated, starting at state $s_{t+m}$.

## Theory

We first motivate the proposed algorithm by providing an intuition on how the sample complexity is improved over a naïve lookahead search method. Considering the initial Monte Carlo rollouts with $|\mathcal{A}|$ trajectories each of length $N$ (where $N$ denotes the maximum episode length before guaranteed termination), completion of an entire episode yields a total computational complexity of $\mathcal{O}(|\mathcal{A}|N^2)$ per training episode. Our first optimization is based on the first




$$c^* = \underset{c}{\operatorname{argmin}}\, K_d(\hat{Q}_0, Q_c)$$

$$\pi_{c^*} = \underset{a}{\operatorname{argmax}}\; Q_{c^*}(s,a)$$

Figure 2: Illustration of the proposed algorithm. As the agent is trained, a trajectory (of length $n^*$) is rolled out using the online estimate of $\pi^*$, starting from each possible action ($a \in \mathcal{A} \doteq \{a_1, a_2, a_3, a_4\}$). The initial actions are then ranked by their values using an estimate of the $Q$-function. This value estimate, $\widehat{Q}_0(s_t, a_t)$, is based on the corresponding $n^*$-step trajectory returns and bootstrapping at the final step (Eq. (5)). Finally, the Kendall tau distance, $K_d$, is used to measure the discrepancy between this ranking, and the action ranking induced by each cached solution at the same state. The most well-aligned policy (as measured here by $\text{argmin}_c K_d$) is used for the next $m$ steps in online environment interaction.

specified horizon ($n^*$) which represents the trajectory length given by optimizing against the cost-accuracy tradeoff (cf. Fig. 3). This leads to a reduced complexity of $\mathcal{O}(|\mathcal{A}|Nn^*)$. Finally, we further reduce the total lookahead ("planning") steps by a factor of $m$, since the cached policy is rolled out online for $m$ steps without repeating the lookahead, giving a final complexity per episode of $\mathcal{O}(|\mathcal{A}|Nn^*m^{-1})$. As $|\mathcal{A}|Nn^*m^{-1} \ll |\mathcal{A}|N^2$, CASH offers significantly reduced scaling in complexity per episode compared to naïve lookahead search.

Next, we derive an error bound on the difference between the online task's action-value function and its estimate,

**Lemma 2** (Error Reduction Property)**.** *Let $R_{max}$ denote the maximal reward in the online MDP. Suppose the online policy $\pi$ (greedy with respect to $Q_0$) is used to collect trajectories of length $n$ to estimate the return beginning from state $s$ and action $a$. Then the error in the value function can be bounded as*

$$\left|Q_0^\pi(s,a) - \widehat{Q}_0^\pi(s,a)\right| \leq \gamma^n \frac{R_{max}}{1-\gamma}.$$

(The proof can be found throughout the literature (Barreto et al. 2017b) and in standard texts on RL (Sutton and

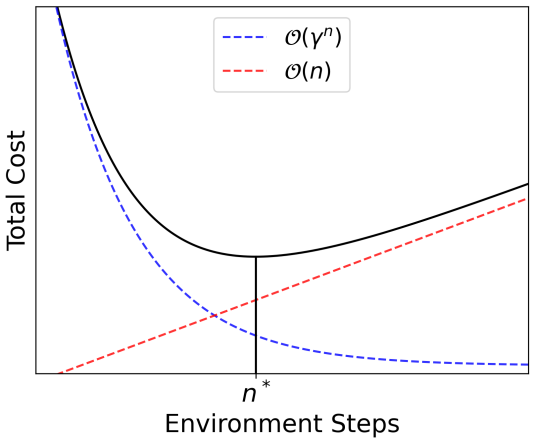



Figure 3: To estimate the $Q$-function, there is a tradeoff between (a) larger $n$-step returns which reduces error exponentially, and (b) linear increases in cost of compute for rollouts of length $n$. The combined cost and error then yield an optimal choice for the rollout trajectory length, which we denoted $n^*$.

Barto 2018).) Note that running the trajectories used for this estimation yield a cost linear in the number of timesteps:

**Assumption 1.** The cost of computing a rollout is linear in the number of timesteps $n$, with coefficient $h > 0$ giving a total simulation cost of $hn$.

Here, we essentially assume the $|\mathcal{A}|$ trajectories are performed in parallel, and are computed sequentially. As seen under different contexts in the literature (cf. discussion in (Dutta et al. 2018)), a cost-accuracy tradeoff is now presented, which can be used to determine an optimal trajectory length, $n^*$, defined below.

**Lemma 3.** *Let $B = 2(1-\gamma)^{-1}R_{max}$ and $h$ denote the per-step cost of the trajectory as in Assumption 1. The trajectory length that minimizes the sum of computational cost and value estimation error is:*

$$n^* = \left\lfloor \frac{1}{\log \gamma^{-1}} \log \frac{B \log \gamma^{-1}}{h} \right\rceil, \tag{6}$$

where $\lfloor \cdot \rceil$ denotes the nearest integer. This result follows immediately from Theorem 2 by adding the rollout cost $hn$ and differentiating with respect to $n$. So long as $n \geq 1$, there is a possible advantage in using this approach. This can be ensured by satisfying the following condition: $B \leq \gamma^{-1} hg$. Using a large horizon $H = (1-\gamma)^{-1}$ for $H \gg 1$, the above expression simplifies to $n^* = \mathcal{O}\left(H \log 2h^{-1}\right)$ (see Appendix), giving a better intuition for the optimal trajectory length as a function of horizon (linear scaling) and rollout cost (logarithmic scaling).

**Lemma 4.** *Let $d^{\pi_c}$ denote the discounted state occupancy under $\pi_c$ and let $\delta$ denote the maximum margin in action-value. That is, $|Q_0^\pi(s,a) - Q_0^\pi(s,a')| \leq \delta$ for all $s, a, a'$. Then, the policy $\pi_c$ can estimate the value of the desired policy $\pi$ with bounded error:*

$$|Q_0^\pi(s,a) - Q_0^{\pi_c}(s,a)| \leq \frac{\gamma\delta}{1-\gamma} \mathop{\mathbb{E}}_{s' \sim d^{\pi_c}} K_d(Q^\pi, Q^{\pi_c}; s).$$

We note the following caveats: (1) The above bound accounts for discrepancies only in the optimal actions, but we hope the Kendall tau introduces "higher order" information and (2) the value estimate of $\pi_c$ in the online MDP may not be readily available. If one operates in the successor feature setting, though, then one can obtain the value function easily through a dot product $Q_c^\pi(s,a) = \psi^{\pi_c}(s,a) \cdot w$, where $w$ represents the weights for the online MDP.

Once the policy with minimal Kendall tau distance has been chosen (cf. Fig 1 and Fig 2), a natural question follows: For how long can the policy $\pi_{c^*}$ be "trusted" to yield similar near-optimal trajectories? When the error in using the "best" cached policy exceeds the next-best cached policy's error, the previous bound becomes void. This allows us to determine the number of steps for which the cached policy can be safely deployed, which we denote as $m$. With a continuous state space and some additional assumptions on the structure of the MDP, we can provide the following lemma, giving insight into this secondary rollout horizon.

**Assumption 2** (Dynamical Locality)**.** There exists a constant $d > 0$ such that

$$|s - s'| < d, \tag{7}$$

for all $s'$ satisfying $p(s'|s,a) > 0$, where $p$ denotes the (stochastic) transition dynamics.

**Lemma 5.** *Suppose Assumption 2 holds for some $d > 0$. Let $Q(s,a)$ be Lipschitz continuous with constant $L_Q$. For all trajectories $\tau = (s_0, a_0, \ldots s_n, a_n)$, the difference in initial and terminal $Q$ values is bounded by:*

$$\left|Q(s_0,a_0) - Q(s_n,a_n)\right| \leq L_Q \cdot d \cdot n. \tag{8}$$

The Lipschitz property of value functions is discussed in detail by (Rachelson and Lagoudakis 2010). The proof of this follows simply from the definitions, but is given in the Appendix for completeness.

**Lemma 6.** *Let the estimate of $Q_0^\pi$ have bounded error: $|Q_0^\pi(s,a) - Q_0^{\pi_c}(s,a)| < \varepsilon_c$. If the $Q$ functions each have Lipschitz constant upper bounded by $L_Q$, and Assumption 2 is satisfied for all $Q$-functions in the cache (including $Q_0$), then the optimized cache policy $c^*$ remains optimal (in the sense of least Kendall tau, e.g.) for at least $m$ steps, with*

$$m = \frac{L_Q d}{4\left(\varepsilon_{c^*} - \varepsilon_{\bar{c}}\right)}, \tag{9}$$

*where $\bar{c}$ denotes the cached value function with next-smallest distance.*

*Proof.* The optimal policy in the cache (denoted $c^*$) is only guaranteed to be optimal when its error, $\varepsilon_{c^*}$, does not exceed that of the second-best policy in the cache (denoted $\bar{c}$), which has an error $\varepsilon_{\bar{c}}$. There are three conflicting accumulations of error: (1) The true $Q$-values of $Q_0$ (which were estimated by $\widehat{Q}_0$), the best cached policy's estimate $Q_{c^*}$ and the second

best cached policy's estimate $Q_{\bar{c}}$ all change at a rate of $L_Q d$. Since (at worst) the best approximation for the true $Q$-values can diverge from $Q_0$ at a rate $2L_Q d$, and the second-best approximation for the $Q$-values can *approach* $Q_0^*$ at a rate $2L_Q d$, the gap in question $\varepsilon_{c^*} - \varepsilon_{\bar{c}}$ can be closed in no fewer than $m$ steps. □

## Conclusions

In this work, we introduced CASH, a new algorithm for reusing old solutions in an online model-based setting. We offer a new similarity metric which is used to compare rankings of simulated rollouts to action-rankings dictated by the cached policies. This notion of similarity is inspired by a relative ranking between actions, instead of focusing only on the greedy maximizing actions. We have established a relationship between cached solutions, value error bounds and compute cost, leading to an optimizable trajectory length. This framework yields a novel method of generalization for planning, and leading to theoretical statements revealing connections between core ideas across RL.

There are several promising directions which can build off of this preliminary theoretical work. First, we aim to provide a practical implementation of the CASH algorithm to study it empirically, comparing its performance to the suggested theoretical results. Second, we intend to explore alternative distance metrics and further extend the theoretical results by providing formal sample complexity analysis and relating these results to model-based techniques. We believe that this algorithm at the intersection of model-based RL and transfer learning can enhance training efficiency in complex problem settings.

## Acknowledgements

JA acknowledges funding support the NSF through Award No. PHY-2425180 and JA from AIVO through UC Davis; the use of the supercomputing facilities managed by the Research Computing Department at UMass Boston; and fruitful discussions with Rahul V. Kulkarni and Stas Tiomkin. JCK acknowledges support from the Computational Biology Department at CMU, and from Shiladitya Banerjee in the Department of Physics at CMU.

## References

Adamczyk, J.; Makarenko, V.; Arriojas, A.; Tiomkin, S.; and Kulkarni, R. V. 2023. Bounding the optimal value function in compositional reinforcement learning. In Evans, R. J.; and Shpitser, I., eds., *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, 22–32. PMLR.

Barreto, A.; Dabney, W.; Munos, R.; Hunt, J. J.; Schaul, T.; van Hasselt, H. P.; and Silver, D. 2017a. Successor features for transfer in reinforcement learning. *Advances in Neural Information Processing Systems*, 30.

Barreto, A.; Dabney, W.; Munos, R.; Hunt, J. J.; Schaul, T.; van Hasselt, H. P.; and Silver, D. 2017b. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279.

Dutta, S.; Joshi, G.; Ghosh, S.; Dube, P.; and Nagpurkar, P. 2018. Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD. In *International conference on artificial intelligence and statistics*, 803–812. PMLR.

Gleave, A.; Dennis, M.; Legg, S.; Russell, S.; and Leike, J. 2020. Quantifying differences in reward functions. *arXiv preprint arXiv:2006.13900*.

Haarnoja, T.; Pong, V.; Zhou, A.; Dalal, M.; Abbeel, P.; and Levine, S. 2018. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE international conference on robotics and automation (ICRA)*, 6244–6251. IEEE.

Hafner, D.; Pasukonis, J.; Ba, J.; and Lillicrap, T. 2023. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*.

Hunt, J.; Barreto, A.; Lillicrap, T.; and Heess, N. 2019. Composing entropic policies using divergence correction. In *International Conference on Machine Learning*, 2911–2920. PMLR.

Kakade, S.; and Langford, J. 2002. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 267–274.

Kendall, M. G. 1938. A New Measure of Rank Correlation. *Biometrika*, 30(1/2): 81–93.

Nemecek, M.; and Parr, R. 2021. Policy caches with successor features. In *International Conference on Machine Learning*, 8025–8033. PMLR.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, volume 99, 278–287.

Peng, X. B.; Chang, M.; Zhang, G.; Abbeel, P.; and Levine, S. 2019. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in Neural Information Processing Systems*, 32.

Rachelson, E.; and Lagoudakis, M. G. 2010. On the Locality of Action Domination in Sequential Decision Making. In *11th International Symposium on Artificial Intelligence and Mathematics (ISIAM 2010)*, 1–8. Fort Lauderdale, US.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Saxe, A. M.; Earle, A. C.; and Rosman, B. 2017. Hierarchy through composition with multitask LMDPs. In *International Conference on Machine Learning*, 3017–3026. PMLR.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. 2020. Mastering atari, go, chess and shogi

by planning with a learned model. *Nature*, 588(7839): 604–609.

Skalse, J.; Farnik, L.; Motwani, S. R.; Jenner, E.; Gleave, A.; and Abate, A. 2023. STARC: A General Framework For Quantifying Differences Between Reward Functions. *arXiv preprint arXiv:2309.15257*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Tasse, G. N.; James, S.; and Rosman, B. 2020. A Boolean task algebra for reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 9497–9507.

Taylor, M. E.; and Stone, P. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research*, 10(56): 1633–1685.

Wulfe, B.; Balakrishna, A.; Ellis, L.; Mercat, J.; McAllister, R.; and Gaidon, A. 2022. Dynamics-aware comparison of learned reward functions. *arXiv preprint arXiv:2201.10081*.

Zhu, Z.; Lin, K.; Jain, A. K.; and Zhou, J. 2023. Transfer Learning in Deep Reinforcement Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11): 13344–13362.

# Proofs

**Proof of Lemma 1**

*Proof.* Since PBRS affects the action-value function only by state-dependent shifts, e.g.: $Q(s,a) \to Q(s,a) - \Phi(s)$, for a fixed state $s$ (as discussed in Definition 1), the ordering of actions remains unchanged. Similarly, when the reward function is scaled by a positive constant, $r(s,a) \to \alpha\, r(s,a)$, the value function similarly scales as $Q(s,a) \to \alpha\, Q(s,a)$. For positive scalings, this merely changes the size of gaps in the action values, and not their rankings. It should be noted, however, that if $\alpha \ll 1$, noise may dominate and the true action rankings may be corrupted. □

**Simplification of Lemma 3**

*Proposition.* The optimal value $n^*$ in Lemma 3 can be simplified as $n = \mathcal{O}\left(H \log h^{-1}\right)$.

*Proof.* Since $H = (1-\gamma)^{-1}$, we equivalently have $\gamma = 1 - H^{-1} = 1 - \epsilon$, where we defined $\epsilon \doteq H^{-1} \ll 1$. Note that $\log \gamma^{-1} = -\log(1-\epsilon) \approx \epsilon$. Thus, setting $R_{\max} = 1$ and ignoring constant factors for simplicity:

$$\begin{aligned}
\frac{1}{\log \gamma^{-1}} \log \frac{B \log \gamma^{-1}}{h} &\approx \frac{1}{\log \gamma^{-1}} \log \frac{\epsilon^{-1} \log \gamma^{-1}}{h} \\
&\approx \frac{1}{\log \gamma^{-1}} \log h^{-1} \\
&\approx \epsilon^{-1} \log h^{-1} \\
&= H \log h^{-1}.
\end{aligned}$$

□

**Proof of Lemma 5**

*Lemma.* Suppose Assumption 2 holds for some $d > 0$. Let $Q(s,a)$ be Lipschitz continuous with constant $L_Q$. For all trajectories $\tau = (s_0, a_0, \ldots s_n, a_n)$, the difference in initial and terminal $Q$ values is bounded by:

$$\left|Q(s_0,a_0) - Q(s_n,a_n)\right| \leq L_Q \cdot d \cdot n. \tag{10}$$

*Proof.* In the discrete action case, we ignore the action distances for simplicity. Rewriting the above difference by introducing consecutive terms in a telescoping sum gives:

$$\begin{aligned}
\left|Q(s_0,a_0) - Q(s_n,a_n)\right| &= \left|Q(s_0,a_0) - Q(s_1,a_1) + Q(s_1,a_1) + \cdots + Q(s_{n-1},a_{n-1}) - Q(s_{n-1},a_{n-1}) + Q(s_n,a_n)\right| \\
&= \left|\sum_{t=0}^{n-1} Q(s_t,a_t) - Q(s_{t+1},a_{t+1})\right| \\
&\leq \sum_{t=0}^{n-1} \left|Q(s_t,a_t) - Q(s_{t+1},a_{t+1})\right| \\
&\leq L_Q \sum_{t=0}^{n-1} \left|d(s_t,s_{t+1})\right| \\
&\leq L_Q d n
\end{aligned}$$

We have used the triangle inequality, followed by the Lipschitz condition and locality assumption at each timestep. □

**Proof of Lemma 4**

*Lemma.* Let $d^{\pi_c}$ denote the discounted state occupancy under $\pi_c$ and let $\delta$ denote the maximum margin in action-value. That is, $|Q_0^\pi(s,a) - Q_0^\pi(s,a')| \leq \delta$ for all $s, a, a'$. Then, the policy $\pi_c$ can estimate the value of the desired policy $\pi$ with bounded error:

$$|Q_0^\pi(s,a) - Q_0^{\pi_c}(s,a)| \leq \frac{\gamma\delta}{1-\gamma} \mathop{\mathbb{E}}_{s' \sim d^{\pi_c}} K_d(Q^\pi, Q^{\pi_c}; s). \tag{11}$$

*Proof.* From the backup equation and the performance difference lemma (Kakade and Langford 2002), we have:

$$
\begin{aligned}
&|Q_0^{\pi}(s,a) - Q_0^{\pi_c}(s,a)| \\
&= \left| r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim p} V_0^{\pi}(s') - \left( r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim p} V_0^{\pi_c}(s') \right) \right| \\
&= \frac{\gamma}{1-\gamma} \left| \mathop{\mathbb{E}}_{s' \sim d^{\pi_c}} \mathop{\mathbb{E}}_{a' \sim \pi_c} (Q_0^{\pi}(s',a') - V_0^{\pi}(s')) \right| \\
&= \frac{\gamma}{1-\gamma} \left| \mathop{\mathbb{E}}_{s' \sim d^{\pi_c}} (Q_0^{\pi}(s',\pi_c(s')) - Q_0^{\pi}(s',\pi(s'))) \right| \\
&\leq \frac{\gamma}{1-\gamma} \mathop{\mathbb{E}}_{s' \sim d^{\pi_c}} K_d(s) \max_{s'} |Q_0^{\pi}(s',\pi_c(s')) - Q_0^{\pi}(s',\pi(s'))| \\
&\leq \frac{\gamma\delta}{1-\gamma} \mathop{\mathbb{E}}_{s' \sim d^{\pi_c}} K_d(s)
\end{aligned}
$$

Where we have used the triangle inequality, positivity of $K_d$, and bounding the action-gap by $\delta$. We assume $\pi_c$ is deterministic, as it represents the greedy solution to some prior task (but this could be generalized). Since $K$ represents an indicator function for whether the same action was chosen over all action-pairs, it is an upper bound on the discrepancy for the optimal action choice alone. □