Leveraging Factored Action Spaces for Efficient Offline Reinforcement Learning in Healthcare

Abstract

Many reinforcement learning (RL) applications have combinatorial action spaces, where each action is a composition of sub-actions. A standard RL approach ignores this inherent factorization structure, resulting in a potential failure to make meaningful inferences about rarely observed sub-action combinations; this is particularly problematic for offline settings, where data may be limited. In this work, we propose a form of linear Q-function decomposition induced by factored action spaces. We study the theoretical properties of our approach, identifying scenarios where it is guaranteed to lead to zero bias when used to approximate the Q-function. Outside the regimes with theoretical guarantees, we show that our approach can still be useful because it leads to better sample efficiency without necessarily sacrificing policy optimality, allowing us to achieve a better bias-variance trade-off. Across several offline RL problems using simulators and real-world datasets motivated by healthcare problems, we demonstrate that incorporating factored action spaces into value-based RL can result in better-performing policies. Our approach can help an agent make more accurate inferences within under-explored regions of the state-action space when applying RL to observational datasets.

1 Introduction

In many real-world decision-making problems, the action space exhibits an inherent combinatorial structure. For example, in healthcare, an action may correspond to a combination of drugs and treatments [1–4]. Past work applying reinforcement learning (RL) typically considers each combination a distinct action, resulting in an exponentially large action space (Figure 1a). This is inefficient as it fails to leverage the potential independence structure among the dimensions of the action space.

To improve learning efficiency, one may incorporate this factorization structure of the action space when designing the architecture of function approximators for RL (Figure 1b). Similar ideas have been used in past work, primarily to improve exploration [5, 6], or to handle multiple agents [7–9] or multiple rewards [10]. However, the applicability of this approach has not been systematically studied when the MDP presents no additional structure, such as factorization of the state space.

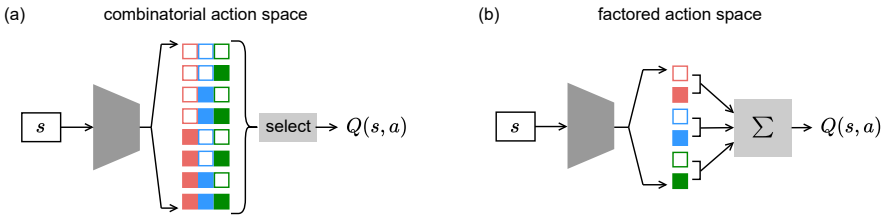


Figure 1: Illustration of Q-network architectures, which take the state s as input and output $Q(s, a)$ for a selected action. In this example, the action space $\mathcal{A}$ consists of $D = 3$ binary sub-action spaces $\{\square, \blacksquare\}$, $\{\square, \blacksquare\}$ and $\{\square, \blacksquare\}$. (a) Learning with the combinatorial action space requires $2^3 = 8$ output heads (exponential in D), one for each combination of sub-actions. (b) Incorporating the linear Q decomposition for the factored action space requires $2 \times 3 = 6$ output heads (linear in D).

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In this work, we develop an approach for offline RL with factored action spaces by learning linearly decomposable Q-functions. First, we study the theoretical properties of this approach, investigating the sufficient and necessary conditions for it to lead to an unbiased estimate of the Q-function (i.e., zero approximation error). Even when the linear decomposition is biased, we note that our approach leads to a reduction of variance, which in turn leads to an improvement in sample efficiency. Lastly, we show that when the sub-actions exhibit certain structures (e.g., when two sub-actions "reinforce" their independent effects), the linear approximation, though biased, can still correspond to the optimal policy. We test our approach in offline RL domains using a simulator [11] and a real clinical dataset [2], where domain knowledge about the relationship among actions suggests our proposed factorization approach applies. Empirically, our approach outperforms a non-factored baseline when the sample size is limited, even when the theoretical assumptions (around the validity of a linear decomposition) are not satisfied perfectly. Qualitatively, in the real-data experiment, our approach learns policies that better capture the effect of less frequently observed treatment combinations.

Our work provides both theoretical insights and empirical evidence for RL practitioners to consider this simple linear decomposition approach. Compatible with any algorithm that has a Q-function component, we expect our approach will lead to the greatest gains for problems with combinatorial action spaces where data are limited in an offline setting, and when domain knowledge can be used to check the validity of theoretical assumptions.

2 Problem Setup

We consider Markov decision processes (MDPs) defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \mu_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $p(s'|s,a)$ and $r(s,a)$ are the transition and instantaneous reward functions, $\mu_0(s)$ is the initial state distribution, and $\gamma \in [0,1]$ is the discount factor. A probabilistic policy $\pi(a|s)$ specifies a mapping from each state to a probability distribution over actions. For a deterministic policy, $\pi(s)$ refers to the action with $\pi(a|s) = 1$. The state-value function is defined as $V^\pi(s) = \mathbb{E}_\pi \mathbb{E}_\mathcal{M} \left[\sum_{t=1}^\infty \gamma^{t-1} r_t \mid s_1 = s\right]$. The action-value function, $Q^\pi(s,a)$, is defined by further restricting the action taken from the starting state. The goal of RL is to find a policy $\pi^* = \arg\max_\pi \mathbb{E}_{s\sim\mu_0}[V^\pi(s)]$ (or an approximation) that has the maximum expected performance.

2.1 Factored Action Spaces

While the standard MDP definition abstracts away the underlying structure within the action space $\mathcal{A}$, in this paper, we explicitly express a factored action space as a Cartesian product of D sub-action spaces, $\mathcal{A} = \bigotimes_{d=1}^D \mathcal{A}_d = \mathcal{A}_1 \times \cdots \times \mathcal{A}_D$. We use $\boldsymbol{a} \in \mathcal{A}$ to denote each action, which can be written as a vector of sub-actions $\boldsymbol{a} = [a_1, \ldots, a_D]$, with each $a_d \in \mathcal{A}_d$. In general, a sub-action space can be discrete or continuous, and the cardinalities of discrete sub-action spaces are not required to be the same. For clarity of analysis and illustration, we consider discrete sub-action spaces in this paper.

2.2 Linear Decomposition of Q Function

The traditional factored MDP literature almost exclusively considers state space factorization [12]. In contrast, here we capitalize on action space factorization. Specifically, our approach considers a linear decomposition of the Q function, as illustrated in Figure 1b:

$$Q^\pi(s, \boldsymbol{a}) = \textstyle\sum_{d=1}^D q_d(s, a_d). \tag{1}$$

Each component $q_d(s, a_d)$ in the summation is allowed to condition on the full state space s and only one sub-action a_d. While similar forms of decomposition have been used in past work, there are key differences in how the summation components are parameterized. In the multi-agent RL literature, each component $q_d(s_d, a_d)$ can only condition on the corresponding state space of the d-th agent [e.g., 7, 8]. The decomposition in Eqn. (1) also differs from a related form of decomposition considered by Juozapaitis et al. [10] where each component $q_d(s, \boldsymbol{a})$ can condition on the full action $\boldsymbol{a}$. To the best of our knowledge, we are the first to consider this specific form of Q-function decomposition backed by both theoretical rigor and empirical evidence; in addition, we are the first to apply this idea to offline RL. Due to space constraints, we discuss other related work in Appendix A.1.

3 Theoretical Analyses

In this section, we study the theoretical properties of the linear Q-function decomposition induced by factored action spaces. We first study the sufficient and necessary conditions for our approach to introduce no bias, and then analyze settings in which our approach can reduce variance without sacrificing policy performance when the conditions are violated. Finally, we discuss how domain knowledge may be used to check the validity of these conditions, providing examples in healthcare.

3.1 Sufficient Conditions for Zero Bias

If we consider the total return of D MDPs running in parallel, where each MDP is defined by their respective state space $\mathcal{S}_d$ and action space $\mathcal{A}_d$, then the desired linear decomposition holds for the MDP defined by the joint state space $\bigotimes_{d=1}^{D} \mathcal{S}_d$ and joint action space $\bigotimes_{d=1}^{D} \mathcal{A}_d$ (formally discussed in Appendix B.1). However, this relies on an explicit, known state space factorization, limiting its applicability. In contrast, we now present a generalization that forgoes the explicit factorization of the state space by making use of state abstractions. Intuitively, the MDP should have some implicit factorization, such that it is homomorphic to D parallel MDPs. It is, however, not a requirement that this factorization is known, as long as it exists.

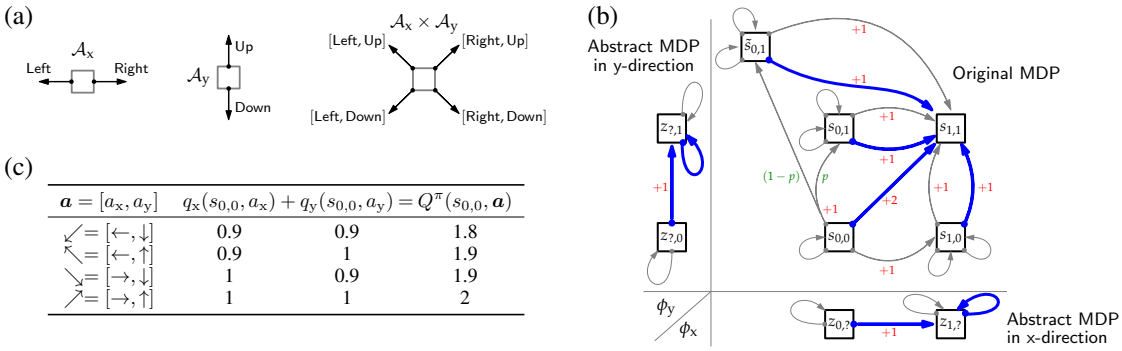
Theorem 1. *Given an MDP defined by $\mathcal{S}, \mathcal{A}, p, r$ and a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, where $\mathcal{A} = \bigotimes_{d=1}^{D} \mathcal{A}_d$ is a factored action space with D sub-action spaces, if there exists D unique corresponding state abstractions $\boldsymbol{\phi} = [\phi_1, \cdots, \phi_D]$ where $\phi_d : \mathcal{S} \to \mathcal{Z}_d$, $z_d = \phi_d(s)$, $z'_d = \phi_d(s')$, such that for all s, a, s' the following holds:*

$$\sum_{\tilde{s} \in \boldsymbol{\phi}^{-1}(\boldsymbol{\phi}(s'))} p(\tilde{s}|s, \boldsymbol{a}) = \prod_{d=1}^{D} p_d(z'_d|z_d, a_d) \tag{2}$$

$$r(s, \boldsymbol{a}) = \sum_{d=1}^{D} r_d(z_d, a_d) \tag{3} \qquad \pi(\boldsymbol{a}|s) = \prod_{d=1}^{D} \pi_d(a_d|z_d) \tag{4}$$

for some $p_d : \mathcal{Z}_d \times \mathcal{A}_d \to \Delta(\mathcal{Z}_d)$, $r_d : \mathcal{Z}_d \times \mathcal{A}_d \to \mathbb{R}$, and $\pi_d : \mathcal{Z}_d \to \Delta(\mathcal{A}_d)$, then the Q-function of π can be expressed as $Q^\pi(s, \boldsymbol{a}) = \sum_{d=1}^{D} q_d(s, a_d)$.

In Appendix B.2, we present an induction-based proof of Theorem 1. Since every assumption is used in key steps of the proof, we conjecture that the sufficient conditions cannot be relaxed in general. Consequently, if the sufficient conditions are satisfied, then using Eqn. (1) to parameterize the Q-function leads to zero approximation error and results in an unbiased estimator. Note that this does not require knowledge of $\boldsymbol{\phi}$. To highlight the significance of Theorem 1, we present the following example, in which the state space cannot be explicitly factored, yet the linear decomposition exists (additional examples probing the sufficient conditions can be found in Appendix C).



(c)

$\boldsymbol{a} = [a_x, a_y]$	$q_x(s_{0,0}, a_x)$	$+\ q_y(s_{0,0}, a_y)$	$= Q^\pi(s_{0,0}, \boldsymbol{a})$
$\swarrow = [\leftarrow, \downarrow]$	0.9	0.9	1.8
$\nwarrow = [\leftarrow, \uparrow]$	0.9	1	1.9
$\searrow = [\rightarrow, \downarrow]$	1	0.9	1.9
$\nearrow = [\rightarrow, \uparrow]$	1	1	2

Figure 2: (a) The composition of sub-action spaces $\mathcal{A}_x$ and $\mathcal{A}_y$ results in $\mathcal{A} = \mathcal{A}_x \times \mathcal{A}_y$ depicted by outgoing arrows exiting the corners of each state (denoted by $\square$). ***The corner from which the action exits encodes the direction.*** (b) An MDP with 5 states and 4 actions of the factored action space $\mathcal{A}$. For example, action $\nearrow = [\rightarrow, \uparrow]$ from $s_{0,0}$ moves the agent both right ($\rightarrow$) and up ($\uparrow$), to $s_{1,1}$. Under abstractions $\boldsymbol{\phi} = [\phi_x, \phi_y]$, this MDP can be mapped to two abstract MDPs (with action spaces $\mathcal{A}_x$ and $\mathcal{A}_y$, respectively). The abstract state spaces are $\mathcal{Z}_x = \{z_{0,?}, z_{1,?}\}$ and $\mathcal{Z}_y = \{z_{?,0}, z_{?,1}\}$, respectively, where ? indicates the coordinate ignored by the abstraction. $s_{1,1}$ is an absorbing state whose outgoing transition arrows are not shown. Taking action $\nwarrow = [\leftarrow, \uparrow]$ from $s_{0,0}$ leads to $s_{0,1}$ with probability p and to $\tilde{s}_{0,1}$ with probability $(1-p)$ (denoted in green). Actions taken by a determinisitic policy π are denoted by **bold blue** arrows. π takes the same action $\searrow = [\rightarrow, \downarrow]$ from $s_{0,1}$ and $\tilde{s}_{0,1}$. Nonzero rewards are denoted in red. (c) Linear decomposition of Q^π for $s_{0,0}$ with respect to the factored action space ($\gamma = 0.9$). Similar decompositions for other states can be found (omitted for space).

Example 1 (Two-dimensional chains with abstractions)**.** The factored action space shown in Figure 2a, $\mathcal{A} = \mathcal{A}_x \times \mathcal{A}_y$, is the composition of two binary sub-action spaces: $\mathcal{A}_x = \{\leftarrow, \rightarrow\}$ leading the agent to move left or right, and $\mathcal{A}_y = \{\downarrow, \uparrow\}$ leading the agent to move down or up. Thus, $\mathcal{A}$ consists of four actions, where each action $\boldsymbol{a} = [a_x, a_y]$ leads the agent to move ***diagonally***.

Consider the MDP in Figure 2b with action space $\mathcal{A}$. $\mathcal{S} = \{s_{0,0}, s_{0,1}, \tilde{s}_{0,1}, s_{1,0}, s_{1,1}\}$ contains 5 different states; subscripts indicate the abstract state vector under $\boldsymbol{\phi} = [\phi_{\text{x}}, \phi_{\text{y}}]$ (e.g., $s_{0,1}$ and $\tilde{s}_{0,1}$ are two different raw states but are identical under the abstraction, $\boldsymbol{\phi}(s_{0,1}) = \boldsymbol{\phi}(\tilde{s}_{0,1}) = [z_{0,?}, z_{?,1}]$). There is no explicit state space factorization. One can check Eqn. (2) and (3) are satisfied by comparing the raw transitions and rewards against the abstracted version (e.g., action $\nearrow$ from $s_{0,0}$ moves both $\rightarrow$ (under ϕ_{x}) and $\uparrow$ (under ϕ_{y}) to $s_{1,1}$ and receives the sum of the two rewards, $1+1=2$). For Eqn. (4) to hold, the policy must take the same action from $s_{0,1}$ and $\tilde{s}_{0,1}$. In Figure 2c, we show the linear decomposition of the Q-function for one such policy where Theorem 1 applies, under which the evolution of the MDP can be seen as two chain MDPs running in parallel (also in Figure 2b). $\triangleleft$

3.2 Necessary Conditions for Zero Bias

In Appendix B.5, we derive a necessary condition for the linear parameterization to be unbiased. Unfortunately, the condition is not verifiable unless the exact MDP parameters are known; this highlights the non-trivial nature of the problem. One may naturally question whether the sufficient conditions (which are arguably more verifiable in practice) must hold (i.e., are necessary) for the linear parameterization to be unbiased. Perhaps surprisingly, ***none*** of the conditions are necessary. We state the following propositions and provide justifications through a set of counterexamples below and in Appendix C.

Proposition 2. *There exist an MDP $\mathcal{M}$ and a policy π for which $Q^{\pi}_{\mathcal{M}}$ decomposes as Eqn. (1) but the transition function p of $\mathcal{M}$ does not satisfy Eqn. (2).*

Proposition 3. *There exist an MDP $\mathcal{M}$ and a policy π for which $Q^{\pi}_{\mathcal{M}}$ decomposes as Eqn. (1) but the reward function r of $\mathcal{M}$ does not satisfy Eqn. (3).*

Proposition 4. *There exist an MDP $\mathcal{M}$ and a policy π for which $Q^{\pi}_{\mathcal{M}}$ decomposes as Eqn. (1) but the policy π does not satisfy Eqn. (4).*

Example 2 (Modified two-dimensional chains)**.** In Figure 3, all conditions in Theorem 1 are violated, yet for each state, there exists a linear decomposition of Q-values (see Appendix C). $\triangleleft$

Therefore, while Theorem 1 imposes a rather stringent set of assumptions on the MDP structure (transitions, rewards) and the policy, violations of these conditions do not preclude the linear parameterization of the Q-function from being an unbiased estimator.

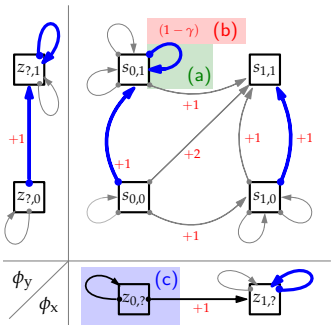


Figure 3: This MDP is similar to Example 1 (except it does not have state $\tilde{s}_{0,1}$) and we consider the same abstractions $\boldsymbol{\phi} = [\phi_{\text{x}}, \phi_{\text{y}}]$. The Q-function and decomposition are exactly the same as in the previous example. However, none of the conditions in Theorem 1 are satisfied. (a) The transition function does not satisfy Eqn. (2) because action $\nearrow = [\rightarrow, \uparrow]$ from $s_{0,1}$ does not move right ($\rightarrow$ under ϕ_{x}) to $s_{1,1}$ and instead moves back to state $s_{0,1}$. (b) The reward function does not satisfy Eqn. (3) as the reward of $(1-\gamma)$ for action $\nearrow = [\rightarrow, \uparrow]$ from $s_{0,1}$ is not the sum of $+1$ ($\rightarrow$ from $z_{0,?}$ under ϕ_{x}) and 0 ($\uparrow$ from $z_{?,1}$ under ϕ_{y}). (c) The policy does not satisfy Eqn. (4) as it takes different sub-actions from $z_{0,?}$ under ϕ_{x} ($\nwarrow$ from $s_{0,0}$ specifies $\leftarrow$, whereas $\nearrow$ from $s_{0,1}$ specifies $\rightarrow$).

3.3 How Does Bias Affect Policy Learning?

When the sufficient conditions do not hold perfectly, using the linear parameterization in Eqn. (1) to fit the Q-function may incur nonzero approximation error (bias). This can affect the performance of the learned policy; in Appendix B.3, we derive error bounds based on the extent of bias relative to the sufficient conditions in Theorem 1. Despite this bias, our approach always leads to a reduction in the variance of the estimator. This gives us an opportunity to achieve a better bias-variance trade-off, especially given limited historical data in the offline setting. In addition, as we will demonstrate, biased Q-values does not always result in suboptimal policy performance, and we identify the characteristics of problems where this happens under our proposed linear decomposition.

3.3.1 Bias-Variance Trade-off

While the amount of bias incurred depends on the problem structure, the benefit of variance reduction is immediate. Intuitively, to learn the Q-function of a tabular MDP with state space $\mathcal{S}$ and action space $\mathcal{A} = \bigotimes_{d=1}^{D} \mathcal{A}_d$, the linear parameterization reduces the number of free parameters from $|\mathcal{S}||\mathcal{A}| = |\mathcal{S}|(\prod_{d=1}^{D} |\mathcal{A}_d|)$ to $|\mathcal{S}|(\sum_{d=1}^{D} |\mathcal{A}_d| - D + 1)$ (see Appendix B.4). This reduces the hypothesis class from exponential in D to linear in D. To analyze variance reduction, we compare the bounds on Rademacher complexity [13–15] of the Q-function approximator using the factored action space with that of the full combinatorial action space (formally discussed in Appendix B.6).

Proposition 5. *Using the linear Q-function decomposition for the factored action space in Eqn. (1) has a smaller lower bound on the empirical Rademacher complexity compared to learning the Q-function in the combinatorial action space.*

Proposition 5 shows that our linear Q-function parameterization leads to a smaller function space, which implies a lower-variance estimator. Hence, our factored-action approach can make more efficient use of limited samples, leading to an interesting bias-variance trade-off that is especially beneficial for offline settings with limited data.

3.3.2 Bias $\not\Rightarrow$ Suboptimal Performance

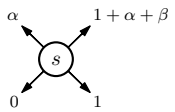
Even in the presence of bias, an inaccurate Q-function may still correctly identify the value-maximizing action (Proposition 6). While this statement is generally true, in this section, we identify ***when*** this occurs ***specifically given*** our linear decomposition based on factored action spaces. To focus the analysis on the most interesting aspects unique to our approach, we consider a bandit setting; extensions to the sequential RL setting are possible under additional technical assumptions [16, 17, 14].

Proposition 6. *There exists an MDP with the optimal Q^* and its approximation $\hat{Q}$ in Eqn. (1), such that $\hat{Q} \neq Q^*$ and yet $\arg\max_{\boldsymbol{a}} \hat{Q}(\boldsymbol{a}) = \arg\max_{\boldsymbol{a}} Q^*(\boldsymbol{a})$.*

Justification. Consider a 1-step bandit problem with a single state and the same action space as before, $\mathcal{A} = \mathcal{A}_{\mathrm{x}} \times \mathcal{A}_{\mathrm{y}}$. Taking an action $\boldsymbol{a} = [a_{\mathrm{x}}, a_{\mathrm{y}}]$ leads the agent to move diagonally and terminate immediately. Since there are no transitions, the Q-values of any policy are simply the immediate reward from each action, $Q(\boldsymbol{a}) = r(\boldsymbol{a})$. We assume the reward function is defined as in Figure 4a (Appendix B.7 describes a procedure to standardize an arbitrary reward function). Applying our approach amounts to solving for the parameters $r_{\text{Left}}, r_{\text{Right}}, r_{\text{Down}}, r_{\text{Up}}$ of the linear system in Figure 4b, while dropping the interaction term r_{Interact}, resulting in a form of omitted-variable bias [18]. Solving the system gives the approximate value function where the interaction term β appears in the approximation $\hat{Q}$ for all arms (Figure 4c, details in Appendix B.8).

Note that $\hat{Q} = Q^*$ only when $\beta = 0$, i.e., there is no interaction between the two sub-actions. We first consider the family of problems with $\alpha = 1$ and $\beta \in [-4, 4]$. In Figure 5a, we measure the value approximation error $\mathrm{RMSE}(Q^*, \hat{Q})$, as well as the suboptimality $V^{\pi^*} - V^{\hat{\pi}} = \max_{\boldsymbol{a}} Q^*(\boldsymbol{a}) - Q^*(\arg\max_{\boldsymbol{a}} \hat{Q}(\boldsymbol{a}))$ of the greedy policy $\hat{Q}$ compared to π^*. As expected, when $\beta = 0$, $\hat{Q}$ is unbiased and has zero approximation error. When $\beta \neq 0$, $\hat{Q}$ is biased and RMSE > 0; however, for $\beta \geq -1$, $\hat{Q}$ corresponds to a policy that correctly identifies the optimal action.

(a)



(b)

$$\begin{aligned} r_{\text{Left}} + r_{\text{Down}} &= 0 \\ r_{\text{Left}} + r_{\text{Up}} &= \alpha \\ r_{\text{Right}} + r_{\text{Down}} &= 1 \\ r_{\text{Right}} + r_{\text{Up}} + \cancel{r_{\text{Interact}}} &= 1 + \alpha + \beta \end{aligned}$$

(c)

True Value Function

$$\begin{bmatrix} Q^*(\swarrow) \\ Q^*(\nwarrow) \\ Q^*(\searrow) \\ Q^*(\nearrow) \end{bmatrix} = \begin{bmatrix} 0 \\ \alpha \\ 1 \\ 1 + \alpha + \beta \end{bmatrix}$$

Linear Approximation

$$\begin{bmatrix} \hat{Q}(\swarrow) \\ \hat{Q}(\nwarrow) \\ \hat{Q}(\searrow) \\ \hat{Q}(\nearrow) \end{bmatrix} = \begin{bmatrix} -\frac{1}{4}\beta \\ \alpha + \frac{1}{4}\beta \\ 1 + \frac{1}{4}\beta \\ 1 + \alpha + \frac{3}{4}\beta \end{bmatrix}$$

Figure 4: (a) A two-dimensional bandit problem with action space $\mathcal{A}$. Rewards are denoted for each arm. (b) Learning using the linear Q decomposition approach corresponds to a system of linear equations that relates the reward of each arm. The parameter r_{Interact} is dropped in our linear approximation, leading to omitted-variable bias. (c) Solving the system results in an approximate value function $\hat{Q}$, which does not equal to the true value function Q^* unless $\beta = 0$.

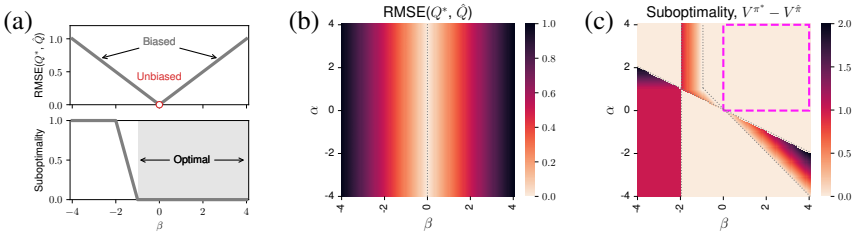


Figure 5: (a) The approximation error and policy suboptimality of our approach for the bandit problem in Figure 4a, for different settings of β when $\alpha = 1$. The Q-value approximation is unbiased only when $\beta = 0$, but the corresponding approximate policy is optimal for a wider range of $\beta \geq -1$. (b-c) The approximation error and policy suboptimality of our approach for the bandit problem in Figure 4a, for different settings of α and β. The Q-value approximation is unbiased only when $\beta = 0$, but the corresponding approximate policy is optimal for a wider range of α and β values. The highlighted region of zero suboptimality corresponds to $\alpha \geq 0$ and $\beta \geq 0$.

We further investigate this phenomenon considering both $\alpha, \beta \in [-4, 4]$ (to show all regions with interesting trends), measuring RMSE and suboptimality as above. As shown in Figure 5b, the approximation error is zero only when $\beta = 0$, regardless of α. However, in Figure 5c, for a wide range of α and β settings, suboptimality is zero; this suggests that in those regions, even in the presence of bias (non-zero approximation error), our approach leads to an approximate value function that correctly identifies the optimal action. The irregular contour outlines multiple regions where this happens; one key region is when the two sub-actions affect the reward in the same direction (i.e., $\alpha \geq 0$) and their interaction effects also affect the reward in the same direction (i.e., $\beta \geq 0$).

3.4 Practical Considerations: Are these Assumptions too Strong?

Based on our theoretical analysis, strong assumptions (Section 3.1) on the problem structure (though not necessary, Section 3.2) are the only known way to guarantee the unbiasedness of our proposed linear approximation. It is thus crucial to understand the applicability (and inapplicability) of our approach in real-world scenarios. Exploring to what extent these assumptions hold in practice is especially important for safety-critical domains such as healthcare where incorrect actions (treatments) can have devastating consequences. Fortunately, RL tasks for healthcare are often equipped with significant domain knowledge, which serves as a better guide to inform the algorithm design than heuristics-driven reasoning alone [19, 5, 8].

Oftentimes, when clinicians treat conditions using multiple medications at the same time (giving rise to the factored action space), it is because each medication has a different "mechanism of action," resulting in negligible or limited interactions. For example, several classes of medications are used in the management of chronic heart failure, and each has a unique and incremental benefits on patient outcomes [20]. Problems such as this satisfy the sufficient conditions in Section 3.1 in spite of a non-factorized state space. Moreover, any small interactions would have a bounded effect on RL policy performance (according to Appendix B.3).

Similarly, in the management of sepsis (which we consider in Section 4.2), fluids and vasopressors affect blood pressure to correct hypotension via different mechanisms [21]. Fluid infusion increases "preload" by increasing the blood return to the heart to make sure the heart has enough blood to pump out [22]. In contrast, common vasopressors (e.g., norepinephrine) increase "inotropy" by stimulating the heart muscle and increase peripheral vascular resistance to maintain perfusion to organs [23, 24]. Therefore, while the two treatments may appear to operate on the same part of the state space (e.g., they both increase blood pressure), in general they are not expected to interfere with each other. Recently, there has also been evidence suggesting that their combination can better correct hypotension [25], which places this problem approximately in the regime discussed in Section 3.3.2.

Given limited historical data in the offline setting, the reduction in variance by our approach can outweigh any potential small bias incurred in the scenarios above and lead to overall performance improvement (Section 3.3.1). However, our approach is not suitable if the interaction is counter to the effect of the sub-actions (e.g., two drugs that raise blood pressure individually, but when combined lead to a decrease). In such scenarios, the resulting bias will likely lead to suboptimal performance (Section 3.3.2). However, many drug-drug interactions are known and/or predictable [26–29]. In such cases, one can either explicitly encode the interaction terms or resort back to a combinatorial action space. While we focus on healthcare, there are other domains (e.g., education) in which significant domain knowledge regarding the interactions among sub-actions is available. In such scenarios, this knowledge can and should be leveraged.

4 Experimental Evaluations

We apply our approach to two offline RL problems from healthcare: a simulated and a real-data problem, both having an action space that is composed of several sub-action spaces. These problems correspond to settings discussed in Section 3.4 where we expect our proposed approach to perform well. In the following experiments, we compare our proposed approach (Figure 1b), which makes a simplifying assumption regarding the effect of sub-actions in combination with other sub-actions, against a common baseline that considers a combinatorial action space (Figure 1a).

4.1 Simulated Domain: Sepsis Simulator

Rationale. First, we apply our approach to a simulated domain modeled after the physiology of patients with sepsis [11]. Although the policies are learned "offline," a simulated setting allows us to evaluate the learned policies in an "online" fashion without requiring offline policy evaluation (OPE).

Setup. Following prior work [30], a state is represented by a feature vector $\mathbf{x}(s) \in \{0,1\}^{21}$ that uses a one-hot encoding for each underlying variable (diabetes status, heart rate, blood pressure, oxygen concentration, glucose: all of which are discrete). The action space is composed of 3 binary treatments: antibiotics, vasopressors, and mechanical ventilation, such that $\mathcal{A} = \mathcal{A}_{\text{abx}} \times \mathcal{A}_{\text{vaso}} \times \mathcal{A}_{\text{mv}}$, with $\mathcal{A}_{\text{abx}} = \mathcal{A}_{\text{vaso}} = \mathcal{A}_{\text{mv}} = \{0,1\}$ and $|\mathcal{A}| = 2^3 = 8$. Each treatment affects certain vital signs and may raise or lower their values with pre-specified probabilities (precise definition in [30]). A patient is discharged alive when all vitals are normal and all treatments have been withdrawn; death occurs if 3 or more vitals are abnormal. Rewards are sparse and only assigned at the end of each episode ($+1$ for survival and -1 for death), after which the system transitions into the respective absorbing state. Episodes are truncated at a maximum length of 20 following [11] (where no terminal reward is assigned). Here, the MDP partly satisfies the sufficient conditions outlined in Section 3. For example, oxygen saturation (which can be seen as a state abstraction) is only affected by mechanical ventilation, whereas heart rate is only affected by antibiotics. However, blood pressure is affected by both antibiotics and vasopressors, meaning the effects of these two sub-actions are *not* independent.

Offline learning. First, we generated datasets with different sample sizes following different behavior policies. We ran fitted Q-iteration for up to 50 iterations using a neural network function approximator, selecting the early-stopping iteration based on ground-truth policy performance. Each setting of sample size and behavior policy was repeated 10 times with different random seeds. Additional details are described in Appendix D.1.

Results. Figure 6 compares median performance of the proposed approach vs. the baseline over the 10 runs (error bars are interquartile ranges). We consider behavior policies that take the optimal action with probability ρ and select randomly among non-optimal actions with probability $1-\rho$.

How does sample size affect performance? We first look at a uniformly random behavior policy ($\rho = 1/|\mathcal{A}| = 0.125$, Figure 6 middle). As expected, larger sample sizes lead to better policy performance for both the baseline and proposed approaches. For lower sample sizes (< 5000), the proposed approach consistently outperforms the baseline. As sample size increases further, the performance gap shrinks and eventually the baseline overtakes our proposed approach. This is because variance decreases with sample size but the bias incurred by the factored approximation does not change. Once there are enough samples, reductions in variance are no longer advantageous and the incurred bias dominates the performance. Overall, this shows that our approach is promising especially for datasets with limited sample size.

How does behavior policy affect performance? As we anneal the behavior policy closer to the optimal policy ($\rho > 0.125$, Figure 6 left two), we reduce the randomness in the behavior policy and limit the amount of exploration possible at the same sample size. The same overall trend largely holds. On the other hand, when the probability of taking the optimal action is lower than random ($\rho < 0.125$, Figure 6 right two), the proposed approach achieves better performance than the baseline with an even larger gap for limited sample sizes ($\leq 10^3$). Without observing the optimal actions ($\rho = 0$), the baseline performs poorly, even for large sample sizes. In comparison, our approach accounts for relationships among actions to some extent and is able to better generalize to the unobserved/under-explored optimal actions, thereby outperforming the baseline.

Takeaways. Under a challenging situation where our theoretical assumptions do not hold perfectly, our proposed approach matches or outperforms the baseline, especially for smaller sample sizes.

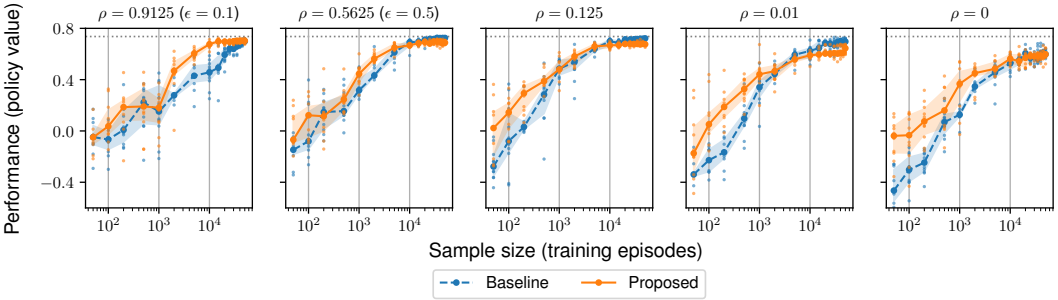


Figure 6: Performance on the sepsis simulator across sample sizes and behavior policies. Plots display the performance over 10 runs, with the trend lines showing medians and error bars showing interquartile ranges. ρ is the probability of taking the optimal action under the behavior policies used to generate offline datasets from the simulator. The left two plots show two ϵ-greedy policies ($\rho > 0.125$; conversion: $\rho = (1-\epsilon) + \epsilon/|\mathcal{A}|$); the middle plot shows a uniformly random policy ($\rho = 0.125$); the right two plots show two policies that undersample the optimal action, $\rho < 0.125$; from left to right, ρ decreases. Across different data distributions, our proposed approach outperforms the baseline at small sample sizes, and closely matches baseline performance at large sample sizes.

4.2 Real Healthcare Data: Sepsis Treatment in MIMIC-III

Rationale. We apply our method to a real-world example of learning optimal sepsis treatment policies for patients in the intensive care unit. Despite the challenging nature of OPE for quantitative comparisons, here we can qualitatively inspect the learned policies using clinical domain knowledge.

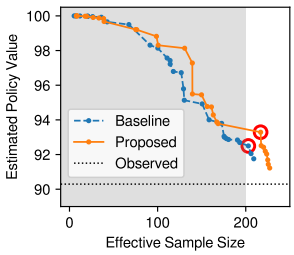
Setup. Originally introduced by [2], we use the improved formulations of this task as per [31] and [32]. After applying the specified inclusion and exclusion criteria to the de-identified MIMIC-III database [33], we obtained a cohort of 19,287 patients and performed a 70/15/15 split for training, validation and testing. For each patient, their data include 10 time-invariant demographic and contextual features and a 33-dimensional time series collected at 4h intervals, consisting of measurements from up to 24h prior until up to 48h after sepsis onset. We used a recurrent neural network (RNN) with long short-term memory (LSTM) cells to create an approximate information state [34] to summarize the history into a $d_{\mathcal{S}}$-dimensional embedding vector. A terminal reward of 100 is assigned for 48h survival and 0 otherwise. Intermediate rewards are all 0. γ for learning is 0.99 and for evaluation is 1. Actions pertain to treatment decisions in each 4h interval, representing total volume of intravenous (IV) fluids and amount of vasopressors administered, resulting in a 5×5 factored action space.

Offline learning. After learning the state representations, we apply variants of discrete-action batch-constrained Q-learning (BCQ) [35, 36], where the baseline uses the combinatorial action space and the proposed approach incorporates the linear decomposition induced by the factored action space. The Q-networks were trained for a maximum of $10,000$ iterations, with checkpoints saved every 100 iterations. We perform model selection [30] over the saved checkpoints (candidate policies) by evaluating policy performance using the validation set with OPE. Specifically, we estimated policy value using weighted importance sampling (WIS) and measured effective sample size (ESS), where the behavior policy is estimated using k nearest neighbors in the embedding space. Following previous work [37], the final policies were selected by maximizing validation WIS with ESS of ≥ 200 (we consider other thresholds in Appendix D.3), for which we report results on the test set.

Results. We visualize the validation performance over all candidate policies. Figure 7-left shows the proposed approach achieves a better Pareto frontier in terms of WIS and ESS compared to baseline.

Quantitative comparisons. Evaluating the final selected policies on the test set (Figure 7-right) shows that the proposed factored BCQ achieves a higher policy value (estimated using WIS) than baseline BCQ at the same level of ESS. In addition, both policies have a similar level of agreement with the clinician policy, comparable to the average agreement among clinicians.

Qualitative comparisons. In Figure 8a, we compare the distributions of recommended actions by the clinician behavior policy, baseline BCQ and factored BCQ, as evaluated on the test set. While overall the policies look rather similar, in that the most frequently recommended action corresponds to low doses of IV fluids <500mL with no vasopressors, there are notable differences for key parts of the action space. In particular, baseline BCQ almost never recommends higher doses of IV fluids



Policy	Baseline BCQ	Factored BCQ	Clinician
Test WIS Test ESS	90.44 ± 2.44 178.32 ± 11.42	91.62 ± 2.12 178.32 ± 11.96	90.29 ± 0.51 2894
% agreement with clinician	62.42%	62.37%	57.16%

Figure 7: Left - Pareto frontiers of validation performance for the candidate policies (all points plotted in Figure 16). The shaded region does not meet the ESS cutoff of ≥ 200. The red circles indicate the selected models (based on best validation WIS) for baseline and proposed (both have a BCQ threshold of $\tau = 0.5$). Right - Performance on test set, $\pm$ standard errors from 100 bootstraps.

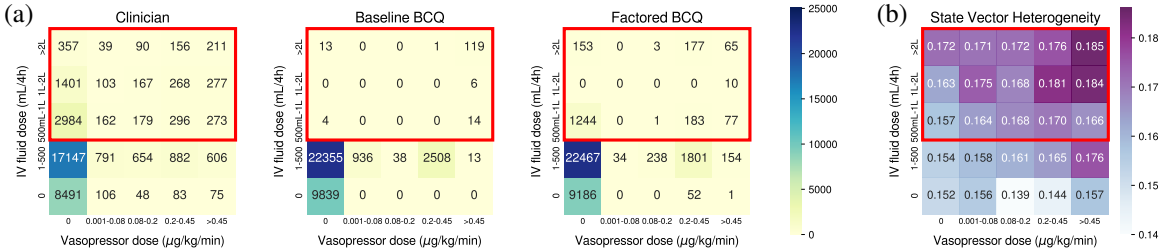


Figure 8: (a) Qualitative comparison of policies. (b) Per-action state heterogeneity, measured as the standard deviation of all state embeddings from which a particular action is observed in the dataset, averaged over state embedding dimensions. Actions with higher IV fluid doses exhibited greater heterogeneity in the observed states from which those actions were taken (by the clinician policy).

>500 mL, either alone or in combination with vasopressors, whereas both clinician and factored BCQ recommend IV fluids >500 mL more frequently. These actions are typically used for critically ill patients, for whom the Surviving Sepsis Campaign guidelines recommends up to >2L of fluids [38]. We hypothesize that this difference is due to a higher level of heterogeneity in the patient states for which actions with high IV fluid doses were observed, compared to the remaining actions with lower doses of IV fluids. To further understand this phenomenon, we measure the per-action state heterogeneity in the test set by computing, for each action, the standard deviation (averaged over the embedding dimensions) of all RNN state embeddings from which that action is taken according to the behavior policy. As shown in Figure 8b, actions with higher IV fluids generally have larger standard deviations, supporting our hypothesis. The larger heterogeneity combined with lower sample sizes makes it difficult for baseline BCQ to correctly infer the effects of these actions, as it does not leverage the relationship among actions. In contrast, our approach leverages the factored action space and can thus make better inferences about these actions.

Takeaways. Applied to real clinical data, our proposed approach outperforms the baseline quantitatively and is able to recommend treatments that align better with clinical knowledge. While promising, these results are based on OPE that has many issues [39]. Further investigation and close collaboration with clinicians are essential before such RL algorithms are deployed in practice.

5 Conclusion

To better leverage factored action spaces in RL, we developed an approach to learning policies that incorporates a simple linear decomposition of the Q-function. As part of the theoretical analysis, we discuss the sufficient and necessary conditions for this parameterization to be unbiased, study its effect on variance reduction, and identify scenarios when this bias does not lead to suboptimal performance. We also note how domain knowledge may be used to inform the applicability of our approach in practice, for problems where any possible bias is negligible or does not affect optimality. Through empirical experiments on two offline RL problems involving a simulator and real clinical data, we demonstrate the advantage of our approach especially with limited sample sizes. Due to space constraints, we provide a further discussion on limitations, ethical considerations and societal impacts of this work in Appendix A. Though motivated by healthcare, our approach could apply more broadly to scale RL to other applications (e.g., education) involving combinatorial action spaces where domain knowledge may be used to verify the theoretical conditions.

References

[1] Damien Ernst, Guy-Bart Stan, Jorge Goncalves, and Louis Wehenkel. Clinical data based optimal STI strategies for HIV: a reinforcement learning approach. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 667–672. IEEE, 2006.

[2] Matthieu Komorowski, Leo A Celi, Omar Badawi, Anthony C Gordon, and A Aldo Faisal. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature medicine*, 24(11):1716–1720, 2018.

[3] Niranjani Prasad, Li Fang Cheng, Corey Chivers, Michael Draugelis, and Barbara E Engelhardt. A reinforcement learning approach to weaning of mechanical ventilation in intensive care units. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

[4] Sonali Parbhoo, Mario Wieser, Volker Roth, and Finale Doshi-Velez. Transfer learning from well-curated to less-resourced populations with HIV. In *Machine Learning for Healthcare Conference*, pages 589–609. PMLR, 2020.

[5] Arash Tavakoli, Fabio Pardo, and Petar Kormushev. Action branching architectures for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[6] Arash Tavakoli, Mehdi Fatemi, and Petar Kormushev. Learning to represent action values as a hypergraph on the action vertices. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Xv_s64FiXTv.

[7] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS, 2018.

[8] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018.

[9] Ming Zhou, Yong Chen, Ying Wen, Yaodong Yang, Yufeng Su, Weinan Zhang, Dell Zhang, and Jun Wang. Factorized Q-learning for large-scale multi-agent systems. In *Proceedings of the First International Conference on Distributed Artificial Intelligence*, pages 1–7, 2019.

[10] Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on Explainable Artificial Intelligence*, 2019.

[11] Michael Oberst and David Sontag. Counterfactual off-policy evaluation with Gumbel-max structural causal models. In *International Conference on Machine Learning*, pages 4881–4890, 2019.

[12] Daphne Koller and Ronald Parr. Computing factored value functions for policies in structured MDPs. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence-Volume 2*, pages 1332–1339, 1999.

[13] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[14] Yaqi Duan, Chi Jin, and Zhiyuan Li. Risk bounds and Rademacher complexity in batch reinforcement learning. *arXiv preprint arXiv:2103.13883*, 2021.

[15] Maggie Makar, Ben Packer, Dan Moldovan, Davis Blalock, Yoni Halpern, and Alexander D'Amour. Causally motivated shortcut removal using auxiliary labels. In *International Conference on Artificial Intelligence and Statistics*, pages 739–766. PMLR, 2022.

[16] Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pages 1042–1051, 2019.

[17] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. In *Advances in Neural Information Processing Systems*, 2020.

[18] Jeffrey M Wooldridge. *Introductory econometrics: A modern approach*. Cengage learning, 2015.

[19] Sahil Sharma, Aravind Suresh, Rahul Ramesh, and Balaraman Ravindran. Learning to factor policies and action-value functions: Factored action space representations for deep reinforcement learning. *arXiv preprint arXiv:1705.07269*, 2017.

[20] Michel Komajda, Michael Boehm, Jeffrey S Borer, Ian Ford, Luigi Tavazzi, Matthieu Pannaux, and Karl Swedberg. Incremental benefit of drug therapies for chronic heart failure with reduced ejection fraction: a network meta-analysis. *European journal of heart failure*, 20(9):1315–1322, 2018.

[21] Jeffrey E Gotts and Michael A Matthay. Sepsis: pathophysiology and clinical management. *Bmj*, 353, 2016.

[22] Laurent Guérin, Jean-Louis Teboul, Romain Persichini, Martin Dres, Christian Richard, and Xavier Monnet. Effects of passive leg raising and volume expansion on mean systemic pressure and venous return in shock in humans. *Critical Care*, 19(1):1–9, 2015.

[23] Olfa Hamzaoui, Jean-François Georger, Xavier Monnet, Hatem Ksouri, Julien Maizel, Christian Richard, and Jean-Louis Teboul. Early administration of norepinephrine increases cardiac preload and cardiac output in septic patients with life-threatening hypotension. *Critical care*, 14 (4):1–9, 2010.

[24] Xavier Monnet, Julien Jabot, Julien Maizel, Christian Richard, and Jean-Louis Teboul. Norepinephrine increases cardiac preload and reduces preload dependency assessed by passive leg raising in septic shock patients. *Critical care medicine*, 39(4):689–694, 2011.

[25] Olfa Hamzaoui. Combining fluids and vasopressors: A magic potion? *Journal of Intensive Medicine*, 2021. ISSN 2667-100X. doi: https://doi.org/10.1016/j.jointm.2021.09.004.

[26] Teijo I Saari, Kari Laine, Mikko Neuvonen, Pertti J Neuvonen, and Klaus T Olkkola. Effect of voriconazole and fluconazole on the pharmacokinetics of intravenous fentanyl. *European journal of clinical pharmacology*, 64(1):25–30, 2008.

[27] Pamela L Smithburger, Sandra L Kane-Gill, and Amy L Seybert. Drug–drug interactions in the medical intensive care unit: an assessment of frequency, severity and the medications involved. *International Journal of Pharmacy Practice*, 20(6):402–408, 2012.

[28] WebMD. Drugs interaction checker - find interactions between medications. https://www.webmd.com/interaction-checker/default.htm.

[29] Epic. Lexicomp and medi-span for epic - drug reference information. https://www.wolterskluwer.com/en/solutions/lexicomp/about/epic/drug-reference.

[30] Shengpu Tang and Jenna Wiens. Model selection for offline reinforcement learning: Practical considerations for healthcare settings. In *Machine Learning for Healthcare Conference*, pages 2–35. PMLR, 2021.

[31] Shengpu Tang, Aditya Modi, Michael Sjoding, and Jenna Wiens. Clinician-in-the-loop decision making: Reinforcement learning with near-optimal set-valued policies. In *International Conference on Machine Learning*, pages 9387–9396. PMLR, 2020.

[32] Taylor W Killian, Haoran Zhang, Jayakumar Subramanian, Mehdi Fatemi, and Marzyeh Ghassemi. An empirical study of representation learning for reinforcement learning in healthcare. In *Machine Learning for Health NeurIPS Workshop*, 2020.

[33] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[34] Jayakumar Subramanian and Aditya Mahajan. Approximate information state for partially observed systems. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1629–1636. IEEE, 2019.

[35] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint arXiv:1910.01708*, 2019.

[36] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[37] Yao Liu and Emma Brunskill. Avoiding overfitting to the importance weights in offline policy optimization, 2022. URL https://openreview.net/forum?id=dLTXoSIcrik.

[38] Laura Evans, Andrew Rhodes, Waleed Alhazzani, Massimo Antonelli, Craig M Coopersmith, Craig French, Flávia R Machado, Lauralyn Mcintyre, Marlies Ostermann, Hallie C Prescott, et al. Surviving sepsis campaign: international guidelines for management of sepsis and septic shock 2021. *Intensive care medicine*, 47(11):1181–1247, 2021.

[39] Omer Gottesman, Fredrik Johansson, Joshua Meier, Jack Dent, Donghun Lee, Srivatsan Srinivasan, Linying Zhang, Yi Ding, David Wihl, Xuefeng Peng, et al. Evaluating reinforcement learning algorithms in observational health settings. *arXiv preprint arXiv:1805.12298*, 2018.

[40] Tengyang Xie and Nan Jiang. Batch value-function approximation with only realizability. In *International Conference on Machine Learning*, pages 11404–11413. PMLR, 2021.

[41] Wenhao Zhan, Baihe Huang, Audrey Huang, Nan Jiang, and Jason D Lee. Offline reinforcement learning with realizability and single-policy concentrability. *arXiv preprint arXiv:2202.04634*, 2022.

[42] Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. Guidelines for reinforcement learning in healthcare. *Nature medicine*, 25(1):16–18, 2019.

[43] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[44] Alexander L Strehl, Carlos Diuk, and Michael L Littman. Efficient structure learning in factored-state MDPs. In *Proceedings of the 22nd national conference on Artificial intelligence-Volume 1*, pages 645–650, 2007.

[45] Karina Valdivia Delgado, Scott Sanner, and Leliane Nunes De Barros. Efficient solutions to factored MDPs with imprecise transition probabilities. *Artificial Intelligence*, 175(9-10): 1498–1527, 2011.

[46] Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored MDPs. *Advances in Neural Information Processing Systems*, 27:604–612, 2014.

[47] Yangyi Lu, Amirhossein Meisami, and Ambuj Tewari. Causal Markov decision processes: Learning good interventions efficiently. *arXiv preprint arXiv:2102.07663*, 2021.

[48] Brian Sallans and Geoffrey E Hinton. Reinforcement learning with factored states and actions. *The Journal of Machine Learning Research*, 5:1063–1088, 2004.

[49] Tom Van de Wiele, David Warde-Farley, Andriy Mnih, and Volodymyr Mnih. Q-learning in enormous action spaces via amortized approximate maximization. *arXiv preprint arXiv:2001.08116*, 2020.

[50] Thomas Pierrot, Valentin Macé, Jean-Baptiste Sevestre, Louis Monier, Alexandre Laterre, Nicolas Perrin, Karim Beguir, and Olivier Sigaud. Factored action spaces in deep reinforcement learning, 2021. URL https://openreview.net/forum?id=naSAkn2Xo46.

[51] Thomas Spooner, Nelson Vadori, and Sumitra Ganesh. Factored policy gradients: Leveraging structure for efficient learning in MOMDPs. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[52] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[53] Laetitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31, 2012.

[54] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PLOS ONE*, 12(4):1–15, 04 2017. doi: 10.1371/journal.pone.0172395. URL https://doi.org/10.1371/journal.pone.0172395.

[55] Harm Van Seijen, Mehdi Fatemi, Joshua Romoff, Romain Laroche, Tavian Barnes, and Jeffrey Tsang. Hybrid reward architecture for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

[56] Luke Metz, Julian Ibarz, Navdeep Jaitly, and James Davidson. Discrete sequential prediction of continuous actions for deep RL, 2018. URL https://openreview.net/forum?id=r1SuFjkRW.

[57] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for MDPs. *Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics*, 4, 2006.

[58] Satinder P Singh and Richard C Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233, 1994.

[59] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2):209–232, 2002.

[60] Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

[61] Pranjal Awasthi, Natalie Frank, and Mehryar Mohri. On the Rademacher complexity of linear hypothesis sets. *arXiv preprint arXiv:2007.11045*, 2020.

[62] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[63] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Section 3.4, Appendix A
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Appendix A
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Sec 3, Appendix B
   (b) Did you include complete proofs of all theoretical results? [Yes] Appendix B

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Sec 4.1-4.2, Appendix D.1-D.2
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Fig 6
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] our experiments only involved small neural networks
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] MIMIC-III by Johnson et al. [33], sepsisSim by Oberst and Sontag [11]
   (b) Did you mention the license of the assets? [No] please refer to the website of the creator of MIMIC-III https://physionet.org/content/mimiciii/1.4/
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] please refer to the website of the creator of MIMIC-III https://physionet.org/content/mimiciii/1.4/
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Sec 4.2, MIMIC-III is de-identified
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Data and Code Availability

The code to reproduce all experiments is provided in the supplementary material and will be released via a public GitHub repository upon publication of this paper. The sepsis simulator is based on prior work with public implementation at https://github.com/clinicalml/gumbel-max-scm. The MIMIC-III database used in the real-data experiments of this paper is publicly available through the PhysioNet website: https://physionet.org/content/mimiciii/1.4/. The cohort definition, extraction and preprocessing code are based on prior work with publicly available implementation at https://github.com/microsoft/mimic_sepsis.

A Additional Discussion

Limitations. Our theoretical analysis in Section 3 focuses on the "realizability" condition of the linear function class [16], where we are interested in guarantees of zero approximation error, i.e., whether the true Q^* lies within the linear function class. In principle, it is possible to find Q^* given a realizable function class (e.g., by enumerating all member functions). However, when Q-learning-style iterative algorithms are used in practice, its convergence relies on a stronger "completeness" condition, as discussed in [16, 40, 41]. We did not investigate how our proposed form of parameterization (and the specific shape of bias introduced) interacts with the learning procedure, and this is an interesting direction for future work.

Ethical Considerations and Societal Impact. In general, policies computationally derived using RL must be carefully validated before they can be used in high-stakes domains such as healthcare. Our linear parameterization implicitly makes an independence assumption with respect to the sub-actions, allowing the Q-function to generalize to sub-action combinations that are under-explored (and even unexplored) in the offline data (as shown in Section 4.1). When the independence assumptions are valid (according to domain knowledge), this is a case of "free lunch" as we can reduce variance without introducing any bias. However, inaccurate or incomplete domain knowledge may render the independence assumptions invalid and cause the agent to incorrectly generalize to dangerous actions (e.g., learned policy recommends drug combinations with adverse side effects, see Section 3.4). This misuse may be alleviated by incorporating additional offline RL safeguards to constrain the learned policy (e.g., BCQ was used in Section 4.2 to restrict the learned policy to not take rarely observed sub-action combinations). Still, to apply RL in safety-critical domains, it is important to consult and closely collaborate with domain experts (e.g., clinicians for healthcare problems) to come up with meaningful tasks and informed assumptions, and perform thorough evaluations involving both the quantitative and qualitative aspects [39, 42].

A.1 Related Work

For many years, the factored RL literature focused exclusively on state space factorization [12, 43–45]. However, in recent years, action space factorization has attracted a growing interest as RL is applied in increasingly more complex planning and control tasks. In particular, researchers have previously considered the model-based setting with a known MDP factorization in which both state and action spaces factor [46, 47]. Within the model-free context, others have studied methods for factored actions with a policy component (i.e., policy-based or actor-critic) [48, 19, 49–51]. In contrast, our work considers value-based methods as those have seen the most success in the offline setting [52].

Among prior work with a value-based component (e.g., Q-network), the majority pertains to multi-agent or multi-objective problems that impose known, explicit assumptions on the state space or the reward function [53, 54, 7, 8]. For single-agent deep online RL, Sharma et al. [19] and Tavakoli et al. [5] incorporated factored action spaces into Q-network architecture designs, but did not provide a formal justification for the linear decomposition. Others have empirically compared various "mixing" functions to combine the values of sub-actions [19, 8]. In contrast, while our work only considers the linear function, we examine its theoretical properties and provide justifications for using this approach in practical problems.

Finally, the sufficient conditions we establish are related to, but different from, those identified by Van Seijen et al. [55] and Juozapaitis et al. [10] who considered reward decompositions in the absence of factored actions. Related, Metz et al. [56] propose an approach that sequentially predicts values for discrete dimensions of a transformed continuous action space, but assume an *a priori* ordering of action dimensions, which we do not. Complementary to our work, Tavakoli et al. [6] proposed to organize the sub-actions and interactions as a hypergraph and linearly combining the values; our theoretical results on the linear decomposition nonetheless applies to their setting where the sub-action interactions are explicitly identified and encoded.

B Detailed Theoretical Analyses

B.1 Sufficient Condition: The Trivial Setting - D Parallel MDPs

To build intuition, we first consider a related setting where D MDPs are running in parallel. If every MDP evolves independently as controlled by its respective policy, then the total return from all MDPs should naturally be the sum of the returns from individual MDPs. Formally, we state the following proposition involving fully factored MDPs and factored policies. Here, we use the vector notation $\boldsymbol{s} = [s_1, \cdots, s_D]$ to clarify the explicit state space factorization.

Definition 1. Given MDPs $\mathcal{M}_1 \cdots \mathcal{M}_D$ where each $\mathcal{M}_d$ is defined by $(\mathcal{S}_d, \mathcal{A}_d, p_d, r_d)$, a fully factored MDP $\mathcal{M} = \bigotimes_{d=1}^{D} \mathcal{M}_d$ is defined by $\mathcal{S}, \mathcal{A}, p, r$ such that $\mathcal{S} = \bigotimes_{d=1}^{D} \mathcal{S}_d$, $\mathcal{A} = \bigotimes_{d=1}^{D} \mathcal{A}_d$, $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) = \prod_{d=1}^{D} p_d(s'_d|s_d, a_d)$, and $r(\boldsymbol{s}, \boldsymbol{a}) = \sum_{d=1}^{D} r_d(s_d, a_d)$.

Definition 2. Given MDPs $\mathcal{M}_1 \cdots \mathcal{M}_d$ and policies $\pi_1 \cdots \pi_D$ where each $\pi_d : \mathcal{S}_d \to \Delta(\mathcal{A}_d)$, then a factored policy $\pi = \bigotimes_{d=1}^{D} \pi_j$ for the MDP $\mathcal{M} = \bigotimes_{d=1}^{D} \mathcal{M}_j$ is $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ such that $\pi(\boldsymbol{a}|\boldsymbol{s}) = \prod_{d=1}^{D} \pi_d(a_d|s_d)$.

Proposition 7. *The Q-function of policy $\pi = \bigotimes_{d=1}^{D} \pi_j$ for MDP $\mathcal{M} = \bigotimes_{d=1}^{D} \mathcal{M}_d$ can be expressed as $Q^{\pi}_{\mathcal{M}}(\boldsymbol{s}, \boldsymbol{a}) = \sum_{d=1}^{D} Q^{\pi_d}_{\mathcal{M}_d}(s_d, a_d)$.*

To match the form in Eqn. (1), we can set $q_d(\boldsymbol{s}, a_d) = Q^{\pi_d}_{\mathcal{M}_d}(s_d, a_d)$. Importantly, each $Q^{\pi_d}_{\mathcal{M}_d}$ does not depend on any $a_{d'}$ where $d' \neq d$. Note that although our definition of q_d is allowed to condition on the entire state space $\boldsymbol{s}$, each $Q^{\pi_d}_{\mathcal{M}_d}$ only depends on s_d. Note that Proposition 7 can be seen as a corollary to Theorem 1 where the abstractions are defined using the sub-state spaces, such that $\phi_d : \mathcal{S} \to \mathcal{S}_d$.

Proof of Proposition 7. Without loss of generality, we consider the setting with $D = 2$ so $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$; extension to $D > 2$ is straightforward. The proof is based on mathematical induction on a sequence of h-step Q-function of π defined as $Q^{\pi,(h)}_{\mathcal{M}}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}[\sum_{t=1}^{h} \gamma^{t-1} r_t | \boldsymbol{s}_1 = \boldsymbol{s}, \boldsymbol{a}_1 = \boldsymbol{a}, \boldsymbol{a}_t \sim \pi]$.

Base case. For $h = 1$, the one-step Q-function is simply the reward, which by assumption $r(\boldsymbol{s}, \boldsymbol{a}) = r_1(s_1, a_1) + r_2(s_2, a_2)$. Therefore, $Q^{\pi,(1)}_{\mathcal{M}}(s, \boldsymbol{a}) = Q^{\pi_1,(1)}_{\mathcal{M}_1}(s_1, a_1) + Q^{\pi_2,(1)}_{\mathcal{M}_2}(s_2, a_2)$.

Inductive step. Suppose $Q^{\pi,(h)}_{\mathcal{M}}(\boldsymbol{s}, \boldsymbol{a}) = Q^{\pi_1,(h)}_{\mathcal{M}_1}(s_1, a_1) + Q^{\pi_2,(h)}_{\mathcal{M}2}(s_2, a_2)$ holds. We can express $Q^{\pi,(h+1)}_{\mathcal{M}}$ in terms of $Q^{\pi,(h)}_{\mathcal{M}}$ using the Bellman equation:

$$Q^{\pi,(h+1)}_{\mathcal{M}}(\boldsymbol{s}, \boldsymbol{a}) = \underbrace{r(\boldsymbol{s}, \boldsymbol{a})}_{①} + \gamma \underbrace{\sum_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) V^{\pi,(h)}_{\mathcal{M}}(\boldsymbol{s}')}_{②}$$

where $V^{\pi,(h)}_{\mathcal{M}}(\boldsymbol{s}') = \sum_{\boldsymbol{a}'} \pi(\boldsymbol{a}'|\boldsymbol{s}') Q^{\pi,h}_{\mathcal{M}}(\boldsymbol{s}', \boldsymbol{a}')$.

By Definition 1, ① can be written as a sum $r(\boldsymbol{s}, \boldsymbol{a}) = r_1(s_1, a_1) + r_2(s_2, a_2)$ where each summand depends on only either a_1 or a_2 but not both. Next we show ② also decomposes in a similar manner.

For a given $\boldsymbol{s}$ we have:

$$\begin{aligned}
V^{\pi,(h)}_{\mathcal{M}}(\boldsymbol{s}) &= \sum_{\boldsymbol{a}} \pi(\boldsymbol{a}|\boldsymbol{s}) Q^{\pi,(h)}_{\mathcal{M}}(\boldsymbol{s}, \boldsymbol{a}) \\
&= \sum_{a_1, a_2} \pi_1(a_1|s_1) \pi_2(a_2|s_2) \Big(Q^{\pi_1,(h)}_{\mathcal{M}_1}(s_1, a_1) + Q^{\pi_2,(h)}_{\mathcal{M}_2}(s_2, a_2) \Big) \\
&= \Big(\cancelto{1}{\textstyle\sum_{a_2} \pi_2(a_2|s_2)} \Big) \textstyle\sum_{a_1} \pi_1(a_1|s_1) Q^{\pi_1,(h)}_{\mathcal{M}_1}(s_1, a_1) + \Big(\cancelto{1}{\textstyle\sum_{a_1} \pi_1(a_1|s_1)} \Big) \textstyle\sum_{a_2} \pi_2(a_2|s_2) Q^{\pi_2,(h)}_{\mathcal{M}_2}(s_2, a_2) \\
&= \Big(\sum_{a_1} \pi_1(a_1|s_1) Q^{\pi_1,(h)}_{\mathcal{M}_1}(s_1, a_1) \Big) + \Big(\sum_{a_2} \pi_2(a_2|s_2) Q^{\pi_2,(h)}_{\mathcal{M}_2}(s_2, a_2) \Big) ,
\end{aligned}$$

where we used the fact that $\pi_1(a_1|s_1)Q^{\pi_1,(h)}_{\mathcal{M}_1}(z_1,a_1)$ is independent of $\pi_2(a_2|s_2)$ (and vice versa), and that $\pi_d(\cdot|s_d)$ is a probability simplex. Letting $V^{\pi_d,(h)}_{\mathcal{M}_d}(s_d) = \sum_{a_d}\pi_1(a_d|s_d)Q^{\pi_d,h}_{\mathcal{M}_d}(s_d,a_d)$, then $V^{\pi,(h)}_{\mathcal{M}}(s') = V^{\pi_1,(h)}_{\mathcal{M}_1}(s'_1) + V^{\pi_2,(h)}_{\mathcal{M}_2}(s'_2)$.

Substituting into ②, we have:

$$
\begin{aligned}
&\sum_{\boldsymbol{s}'} p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})V^{\pi,(h)}_{\mathcal{M}}(\boldsymbol{s}') \\
&= \sum_{s'_1,s'_2} p_1(s'_1|s_1,a_1)p_2(s'_2|s_2,a_2)\left(V^{\pi_1,(h)}_{\mathcal{M}_1}(s'_1) + V^{\pi_2,(h)}_{\mathcal{M}_2}(s'_2)\right) \\
&= \Big(\cancelto{1}{\textstyle\sum_{s'_2} p_2(s'_2|s_2,a_2)}\Big)\textstyle\sum_{s'_1} p_1(s'_1|s_1,a_1)V^{\pi_1,(h)}_{\mathcal{M}_1}(s'_1) + \Big(\cancelto{1}{\textstyle\sum_{s'_1} p_1(s'_1|s_1,a_1)}\Big)\textstyle\sum_{s'_2} p_2(s'_2|s_2,a_2)V^{\pi_2,(h)}_{\mathcal{M}_2}(s'_2) \\
&= \Big(\sum_{s'_1} p_1(s'_1|s_1,a_1)V^{\pi_1,(h)}_{\mathcal{M}_1}(s'_1)\Big) + \Big(\sum_{s'_2} p_2(s'_2|s_2,a_2)V^{\pi_2,(h)}_{\mathcal{M}_2}(s'_2)\Big)
\end{aligned}
$$

where we used the same independence-like property as above and that $p_d(\cdot|s_d,a_d)$ is a probability simplex.

Therefore, we have $Q^{\pi,(h+1)}_{\mathcal{M}}(\boldsymbol{s},\boldsymbol{a}) = Q^{\pi_1,(h+1)}_{\mathcal{M}_1}(s_1,a_1) + Q^{\pi_2,(h+1)}_{\mathcal{M}_2}(s_2,a_2)$ as desired, where $Q^{\pi_d,(h+1)}_{\mathcal{M}_d}(s_d,a_d) = r_d(s_d,a_d) + \gamma\sum_{s'_d} p_d(s'_d|s_d,a_d)\sum_{a'_d}\pi_j(a'_d|s'_d)Q^{\pi_d,(h)}_{\mathcal{M}_j}(s'_d,a'_d)$.

By mathematical induction, this decomposition holds for any h-step Q-function. Letting $h\to\infty$ shows that this holds for the full Q-function. □

B.2 Sufficient Condition: The Abstraction Setting

We first review important background knowledge on state abstractions. Using the properties of state abstractions, we can prove the main sufficient condition in Theorem 1. This proof follows largely from the techniques used in proving Proposition 7, with the exception of how marginalization over the state space is handled.

Background on State Abstractions. A state abstraction (also known as state aggregation) [57], is a mapping $\phi:\mathcal{S}\to\mathcal{Z}$ that converts each element of the primitive state space $\mathcal{S}$ to an element of the abstract state space $\mathcal{Z}$. Intuitively, if two states s_1 and s_2 are mapped to the same element under ϕ, i.e., $\phi(s_1)=\phi(s_2)$, then they are treated as the same (abstract) state under the abstraction. Therefore, we can view an abstraction as a partitioning of the primitive state space into non-overlapping subsets. Since a state abstraction is a many-to-one mapping, we define its inverse as $\phi^{-1}(z) = \{\tilde{s}:\phi(\tilde{s})=z\}$, a set containing all primitive states that are mapped to the abstract state z.

We have the following property of summations involving state abstractions: for any function $f:\mathcal{S}\to\mathbb{R}$,

$$\sum_{s\in\mathcal{S}} f(s) = \sum_{z\in\mathcal{Z}}\sum_{\tilde{s}\in\phi^{-1}(z)} f(\tilde{s})$$

In other words, the sum of $f(s)$ for all states in $\mathcal{S}$ can be obtained in two different ways: i) directly iterate through the elements of $\mathcal{S}$, ii) first iterate through the partitions of $\mathcal{S}$ (induced by the abstraction), and then iterate through the elements in each partition, giving rise to the double summation. This property allows us to change the index of summation from primitive states to abstract states. For multiple abstractions $\boldsymbol{\phi} = [\phi_1,\cdots,\phi_D]$ where $\phi_d\neq\phi_{d'}$ if $d\neq d'$, denoting $\boldsymbol{z}=\boldsymbol{\phi}(s)=[z_1,\ldots,z_D]$, we can similarly define the inverse abstraction $\boldsymbol{\phi}^{-1}(\boldsymbol{z}) = \{\tilde{s}:\boldsymbol{\phi}(\tilde{s})=\boldsymbol{z}\}$, and the summation property similarly applies.

Proof of Theorem 1. Without loss of generality, we consider the setting with $D=2$ so $\mathcal{A}=\mathcal{A}_1\times\mathcal{A}_2$; extension to $D>2$ is straightforward. The proof is based on mathematical induction on a sequence of h-step Q-function of π denoted by $Q^{(h)}(s,\boldsymbol{a}) = \mathbb{E}[\sum_{t=1}^{h}\gamma^{t-1}r_t|s_1=s,\boldsymbol{a}_1=\boldsymbol{a},\boldsymbol{a}_t\sim\pi]$.

Base case. For $h=1$, the one-step Q-function is simply the reward, which by assumption $r(s,\boldsymbol{a}) = r_1(z_1,a_1)+r_2(z_2,a_2)$. We can trivially set $q^{(1)}_d(z_d,a_d) = r_d(z_d,a_d)$ such that $Q^{(1)}(s,\boldsymbol{a}) = q^{(1)}_1(z_1,a_1)+q^{(1)}_2(z_2,a_2)$.

Inductive step. Suppose $Q^{(h)}(s, \boldsymbol{a}) = q_1^{(h)}(z_1, a_1) + q_2^{(h)}(z_2, a_2)$ holds. We can express $Q^{(h+1)}$ in terms of $Q^{(h)}$ using the Bellman equation:

$$Q^{(h+1)}(s, \boldsymbol{a}) = \underbrace{r(s, \boldsymbol{a})}_{①} + \gamma \underbrace{\sum_{s'} p(s'|s, \boldsymbol{a}) V^{(h)}(s')}_{②}$$

where $V^{(h)}(s') = \sum_{\boldsymbol{a}'} \pi(\boldsymbol{a}'|s') Q^{(h)}(s', \boldsymbol{a}')$.

① can be written as a sum $r(s, \boldsymbol{a}) = r_1(z_1, a_1) + r_2(z_2, a_2)$ where each summand depends on only either a_1 or a_2 but not both. Next we show ② also decomposes in a similar manner.

For a given s we have:

$$\begin{aligned}
&V^{(h)}(s) = \sum_{\boldsymbol{a}} \pi(\boldsymbol{a}|s) Q^{(h)}(s, \boldsymbol{a}) \\
&= \sum_{a_1, a_2} \pi_1(a_1|z_1)\pi_2(a_2|z_2)\Big(q_1^{(h)}(z_1, a_1) + q_2^{(h)}(z_2, a_2)\Big) \\
&= \Big(\cancelto{1}{\textstyle\sum_{a_2} \pi_2(a_2|z_2)}\Big) \textstyle\sum_{a_1} \pi_1(a_1|z_1) q_1^{(h)}(z_1, a_1) + \Big(\cancelto{1}{\textstyle\sum_{a_1} \pi_1(a_1|z_1)}\Big) \textstyle\sum_{a_2} \pi_2(a_2|z_2) q_2^{(h)}(z_2, a_2) \\
&= \sum_{a_1} \pi_1(a_1|z_1) q_1^{(h)}(z_1, a_1) + \sum_{a_2} \pi_2(a_2|z_2) q_2^{(h)}(z_2, a_2)\ ,
\end{aligned}$$

where we used the property that $\pi_1(a_1|z_1) q_1^{(h)}(z_1, a_1)$ is independent of $\pi_2(a_2|z_2)$ (and vice versa), and that $\pi_d(\cdot|z_d)$ is a probability simplex. Letting $v_d^{(h)}(z_d) = \sum_{a_d} \pi_d(a_d|z_d) q_d^{(h)}(z_d, a_d)$, then we can write $V^{(h)}(s') = v_1^{(h)}(z_1') + v_2^{(h)}(z_2')$.

Substituting into ②, we have:

$$\begin{aligned}
&\sum_{s'} p(s'|s, \boldsymbol{a}) V^{(h)}(s') = \sum_{\boldsymbol{z}'} \sum_{\tilde{s} \in \boldsymbol{\phi}^{-1}(\boldsymbol{z}')} p(\tilde{s}|s, \boldsymbol{a}) V^{(h)}(\tilde{s}) \\
&= \sum_{\boldsymbol{z}'} \sum_{\tilde{s} \in \boldsymbol{\phi}^{-1}(\boldsymbol{z}')} p(\tilde{s}|s, \boldsymbol{a}) \Big(v_1^{(h)}(z_1') + v_2^{(h)}(z_2')\Big) \\
&= \sum_{\boldsymbol{z}'} \Big(\sum_{\tilde{s} \in \boldsymbol{\phi}^{-1}(\boldsymbol{z}')} p(\tilde{s}|s, \boldsymbol{a})\Big) \Big(v_1^{(h)}(z_1') + v_2^{(h)}(z_2')\Big) \\
&= \sum_{z_1', z_2'} p_1(z_1'|z_1, a_1) p_2(z_2'|z_2, a_2) \Big(v_1^{(h)}(z_1') + v_2^{(h)}(z_2')\Big) \\
&= \Big(\cancelto{1}{\textstyle\sum_{z_2'} p_2(z_2'|z_2, a_2)}\Big) \textstyle\sum_{z_1'} p_1(z_1'|z_1, a_1) v_1^{(h)}(z_1') + \Big(\cancelto{1}{\textstyle\sum_{z_1'} p_1(z_1'|z_1, a_1)}\Big) \textstyle\sum_{z_2'} p_2(z_2'|z_2, a_2) v_2^{(h)}(z_2') \\
&= \Big(\sum_{z_1'} p_1(z_1'|z_1, a_1) v_1^{(h)}(z_1')\Big) + \Big(\sum_{z_2'} p_2(z_2'|z_2, a_2) v_2^{(h)}(z_2')\Big)
\end{aligned}$$

where on the first line we used the property of state abstractions to replace the index of summation, and from the second to the third line we used the fact that for all $\tilde{s} \in \boldsymbol{\phi}^{-1}(\boldsymbol{z}')$ that have the same abstract state vector $\boldsymbol{z}'$, their value $V^{(h)}(s') = v_1^{(h)}(z_1') + v_2^{(h)}(z_2')$ are equal; this allows us to directly sum their transition probabilities $p(\tilde{s}|s, \boldsymbol{a})$. Following that, we substitute in Eqn. (2), and then use the same independence-like property as above and that $p_d(\cdot|z_d, a_d)$ is a probability simplex.

Therefore, we have $Q^{(h+1)}(s, \boldsymbol{a}) = q_1^{(h+1)}(z_1, a_1) + q_2^{(h+1)}(z_2, a_2)$ as desired where $q_d^{(h+1)}(z_d, a_d) = r_d(z_d, a_d) + \gamma \sum_{z_d'} p_d(z_d'|z_d, a_d) \sum_{a_d'} \pi(a_d'|z_d') q_d^{(h)}(z_d', a_d')$.

By mathematical induction, this decomposition holds for any h-step Q-function. Letting $h \to \infty$ shows that this holds for the full Q-function. □

B.3 Policy Learning with Bias - Performance bounds

Consider a particular model-based procedure for approximating the optimal Q-function using Eqn. (1): i) find approximations $\hat{p}$, $\hat{r}$ that are close to the true transition/reward functions p, r such that there exists some state abstraction set $\boldsymbol{\phi}$ with $\hat{p}$, $\hat{r}$ satisfying (2) and (3) with respect to $\boldsymbol{\phi}$, ii) do planning (e.g., dynamic programming) using the approximate MDP parameters $\hat{p}$ and $\hat{r}$. We can show the following performance bounds; note that these upper bounds are loose and information-theoretic (in that they require knowledge of the implicit factorization).

Proposition 8. *If the approximation errors in $\hat{p}$ and $\hat{r}$ are upper bounded by ϵ_p and ϵ_r for all $s \in \mathcal{S}, \boldsymbol{a} \in \mathcal{A}$:*

$$\sum_{s'} \left| p(s'|s,\boldsymbol{a}) - \hat{p}(s'|s,\boldsymbol{a}) \right| \leq \epsilon_p,$$
$$\left| r(s,\boldsymbol{a}) - \hat{r}(s,\boldsymbol{a}) \right| \leq \epsilon_r,$$

then the above model-based procedure leads to an approximate Q-function $\hat{Q}$ and an approximate policy $\hat{\pi}$ that satisfy:

$$\|Q^*_{\mathcal{M}} - Q^*_{\widehat{\mathcal{M}}}\|_\infty \leq \frac{\epsilon_r}{1-\gamma} + \frac{\gamma \epsilon_p R_{\max}}{2(1-\gamma)^2},$$
$$\|V^*_{\mathcal{M}} - V^{\hat{\pi}}_{\mathcal{M}}\|_\infty \leq \frac{2\epsilon_r}{1-\gamma} + \frac{\gamma \epsilon_p R_{\max}}{(1-\gamma)^2}.$$

Proof. See classical results by Singh and Yee [58] and Kearns and Singh [59] (the simulation lemma). □

B.4 Subspace of Representable Q Functions

To help understand how the linear parameterization of Q-function Eqn. (1) affects the representation power of the function class, we first define the following matrices for action space featurization.

Definition 3. The *sub-action mapping matrix for sub-action space $\mathcal{A}_d$*, $\boldsymbol{\Psi}_d$, is defined as

$$\boldsymbol{\Psi}_j = \begin{bmatrix} — & \boldsymbol{\psi}_d(\boldsymbol{a}^1)^\intercal & — \\ & \vdots & \\ — & \boldsymbol{\psi}_d(\boldsymbol{a}^{|\mathcal{A}|})^\intercal & — \end{bmatrix} \in \{0,1\}^{|\mathcal{A}| \times |\mathcal{A}_d|}$$

where each row $\boldsymbol{\psi}_d(\boldsymbol{a}^i)^\intercal \in \{0,1\}^{1 \times |\mathcal{A}_d|}$ is a one-hot vector with value 1 in column $\mathrm{proj}_{\mathcal{A} \to \mathcal{A}_d}(\boldsymbol{a}^i)$.

Remark. The i-th row of $\boldsymbol{\Psi}_d$ corresponds to an action $\boldsymbol{a}^i \in \mathcal{A}$, and the j-th column corresponds to a particular element of the sub-action space $a^j_d \in \mathcal{A}_d$. The (i,j)-entry of $\boldsymbol{\Psi}_d$ is 1 if and only if the projection of $\boldsymbol{a}^i$ onto the sub-action space $\mathcal{A}_d$ is a^j_d. Since each row is a one-hot vector, the sum of elements in each row is exactly 1, i.e., $\boldsymbol{\psi}_d(\boldsymbol{a}^i)^\intercal \mathbf{1} = 1$.

Definition 4. The *sub-action mapping matrix*, $\boldsymbol{\Psi}$, is defined by a horizontal concatenation of $\boldsymbol{\Psi}_d$ for $d = 1 \ldots D$

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\Psi}_1 & \cdots & \boldsymbol{\Psi}_D \end{bmatrix} \in \{0,1\}^{|\mathcal{A}| \times (\sum_d |\mathcal{A}_d|)}$$

Remark. $\boldsymbol{\Psi}$ describes how to map each action $\boldsymbol{a}^i \in \mathcal{A}$ to its corresponding sub-actions. Therefore, the sum of elements in each row is exactly D, the number of sub-action spaces; $\boldsymbol{\psi}(\boldsymbol{a}^i)^\intercal \mathbf{1} = D$.

Definition 5. The *condensed sub-action mapping matrix*, $\tilde{\boldsymbol{\Psi}}$, is

$$\tilde{\boldsymbol{\Psi}} = \left[\begin{array}{c|ccc} \mathbf{1} & \tilde{\boldsymbol{\Psi}}_1 & \cdots & \tilde{\boldsymbol{\Psi}}_D \end{array} \right] \in \{0,1\}^{|\mathcal{A}| \times \left(1 + \sum_d (|\mathcal{A}_d| - 1)\right)}$$

where the first column contains all 1's, and $\tilde{\boldsymbol{\Psi}}_d$ denotes $\boldsymbol{\Psi}_d$ with the first column removed.

Proposition 9. $\text{colspace}(\boldsymbol{\Psi}) = \text{colspace}(\tilde{\boldsymbol{\Psi}})$ *and* $\text{rank}(\boldsymbol{\Psi}) = \text{rank}(\tilde{\boldsymbol{\Psi}}) = \text{ncols}(\tilde{\boldsymbol{\Psi}})$ *(i.e., matrix $\tilde{\boldsymbol{\Psi}}$ has full column rank). Consequently, $\boldsymbol{\Psi}\boldsymbol{\Psi}^+ = \tilde{\boldsymbol{\Psi}}\tilde{\boldsymbol{\Psi}}^+$.*

Corollary 10. *Suppose the Q-function Q of a policy π at state s is linearly decomposable with respect to the sub-actions, i.e., we can write $Q(s,a) = \sum_{d=1}^{D} q_d(s, a_d)$ for all $a_d \in \mathcal{A}_d$, then there exists $\boldsymbol{w}$ and $\tilde{\boldsymbol{w}}$ such that the column vector containing the Q-values for all actions at state s can be expressed as $\boldsymbol{Q}(s,\mathcal{A}) = \boldsymbol{\Psi}\boldsymbol{w} = \tilde{\boldsymbol{\Psi}}\tilde{\boldsymbol{w}}$. In other words, Eqn. (1) is equivalent to $\boldsymbol{Q}(s,\mathcal{A}) \in \text{colspace}(\tilde{\boldsymbol{\Psi}})$.*

Corollary 11. *Suppose $\boldsymbol{Q}(s,\mathcal{A}) \notin \text{colspace}(\tilde{\boldsymbol{\Psi}})$. Let $\hat{\boldsymbol{w}} = \boldsymbol{\Psi}^+\boldsymbol{Q}(s,\mathcal{A})$ and $\hat{\tilde{\boldsymbol{w}}} = \tilde{\boldsymbol{\Psi}}^+\boldsymbol{Q}(s,\mathcal{A})$ be the least-squares solutions of the respective linear equations. Then $\boldsymbol{\Psi}\hat{\boldsymbol{w}} = \tilde{\boldsymbol{\Psi}}\hat{\tilde{\boldsymbol{w}}}$.*

Remark. Corollaries 10 and 11 imply there are two possible implementations, regardless of whether the true Q-function can be represented by the linear parameterization. Intuitively, both versions try to project the true Q-value vector $\boldsymbol{Q}(s,\mathcal{A})$ for a particular state s onto the subspace spanned by the columns of $\boldsymbol{\Psi}$ or $\tilde{\boldsymbol{\Psi}}$. Since the two matrices have the same column space, the results of the projections are equal. This does not imply $\hat{\boldsymbol{w}}$ and $\hat{\tilde{\boldsymbol{w}}}$ are equal (they cannot be as they have different dimensions), but rather the resultant Q-value estimates are equal, $\hat{\boldsymbol{Q}}(s,\mathcal{A}) = \boldsymbol{\Psi}\hat{\boldsymbol{w}} = \tilde{\boldsymbol{\Psi}}\hat{\tilde{\boldsymbol{w}}}$.

To make the theorem statements more concrete, we inspect a simple numerical example and verify the theoretical properties.

Example 3. Consider an MDP with $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$, where $\mathcal{A}_1 = \{0,1\}$ and $\mathcal{A}_2 = \{0,1\}$. Consequently, $|\mathcal{A}_1| = |\mathcal{A}_2| = 2$ and $|\mathcal{A}| = 2^2 = 4$.

Suppose for state s we can write $Q(s,a) = Q(s,[a_1,a_2]) = q_1(s,a_1) + q_2(s,a_2)$ for all $a_1 \in \mathcal{A}_1, a_2 \in \mathcal{A}_2$. Then

$$\boldsymbol{Q}(s,\mathcal{A}) = \begin{bmatrix} Q(s,a_1=0,a_2=0) \\ Q(s,a_1=0,a_2=1) \\ Q(s,a_1=1,a_2=0) \\ Q(s,a_1=1,a_2=1) \end{bmatrix} = \begin{bmatrix} q_1(s,0)+q_2(s,0) \\ q_1(s,0)+q_2(s,1) \\ q_1(s,1)+q_2(s,0) \\ q_1(s,1)+q_2(s,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_1(s,0) \\ q_1(s,1) \\ q_2(s,0) \\ q_2(s,1) \end{bmatrix}$$

$$= \begin{bmatrix} | \\ - \boldsymbol{\Psi} - \\ | \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{w} \\ | \end{bmatrix} \quad \text{where } \boldsymbol{\Psi} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} q_1(s,0) \\ q_1(s,1) \\ q_2(s,0) \\ q_2(s,1) \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\end{matrix}}\right\} \boldsymbol{w}_1 \\ \left.\vphantom{\begin{matrix}a\\a\end{matrix}}\right\} \boldsymbol{w}_2 \end{matrix}$$

$$\boldsymbol{\Psi} = [\underbrace{\begin{matrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{matrix}}_{\boldsymbol{\Psi}_1} \; \underbrace{\begin{matrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{matrix}}_{\boldsymbol{\Psi}_2}]$$

We can also write

$$\boldsymbol{Q}(s,\mathcal{A}) = \tilde{\boldsymbol{\Psi}}\tilde{\boldsymbol{w}}, \quad \text{where } \tilde{\boldsymbol{\Psi}} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad \tilde{\boldsymbol{w}} = \begin{bmatrix} v_0(s) \\ u_1(s) \\ u_2(s) \end{bmatrix} = \begin{bmatrix} q_1(s,0)+q_2(s,0) \\ q_1(s,1)-q_1(s,0) \\ q_2(s,1)-q_2(s,0) \end{bmatrix}$$

One can verify that $\text{rank}(\boldsymbol{\Psi}) = \text{rank}(\tilde{\boldsymbol{\Psi}}) = 3$ and $\text{colspace}(\boldsymbol{\Psi}) = \text{colspace}(\tilde{\boldsymbol{\Psi}})$, because the columns of $\tilde{\boldsymbol{\Psi}}$ are linearly independent, but the columns of $\boldsymbol{\Psi}$ are not linearly independent:

$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

Furthermore,

$$\boldsymbol{\Psi}^+ = \begin{bmatrix} 3/8 & 3/8 & -1/8 & -1/8 \\ -1/8 & -1/8 & 3/8 & 3/8 \\ 3/8 & -1/8 & 3/8 & -1/8 \\ -1/8 & 3/8 & -1/8 & 3/8 \end{bmatrix}, \quad \tilde{\boldsymbol{\Psi}}^+ = \begin{bmatrix} 3/4 & 1/4 & 1/4 & -1/4 \\ -1/2 & -1/2 & 1/2 & 1/2 \\ -1/2 & 1/2 & -1/2 & 1/2 \end{bmatrix}$$

and

$$\boldsymbol{\Psi}\boldsymbol{\Psi}^+ = \tilde{\boldsymbol{\Psi}}\tilde{\boldsymbol{\Psi}}^+ = \begin{bmatrix} 3/4 & 1/4 & 1/4 & -1/4 \\ 1/4 & 3/4 & -1/4 & 1/4 \\ 1/4 & -1/4 & 3/4 & 1/4 \\ -1/4 & 1/4 & 1/4 & 3/4 \end{bmatrix}.$$

◁

Proof of Proposition 9.

First note that $\boldsymbol{\Psi}$ is a tall matrix for non-trivial cases, with more rows than columns, because $|\mathcal{A}| = \prod_d |\mathcal{A}_d| \geq \sum_d |\mathcal{A}_d|$ if $|\mathcal{A}_d| \geq 2$ for all d (see proof). Therefore, the rank of $\boldsymbol{\Psi}$ is the number of linear independent columns of $\boldsymbol{\Psi}$.

We use the following notation to write matrix $\boldsymbol{\Psi}_d$ in terms of its columns:

$$\boldsymbol{\Psi}_d = \begin{bmatrix} | & & | \\ \boldsymbol{c}_{d,1} & \cdots & \boldsymbol{c}_{d,|\mathcal{A}_d|} \\ | & & | \end{bmatrix}.$$

The following statements are true:

Claim 1: The columns of $\boldsymbol{\Psi}_d$ are pairwise orthogonal, $\boldsymbol{c}_{d,j}{}^\intercal \boldsymbol{c}_{d,j'} = 0, \forall j \neq j'$, and they form an orthogonal basis. This is because each row $\boldsymbol{\psi}_d(\boldsymbol{a}^i)^\intercal$ is a one-hot vector, containing only one 1; this implies that out of the two entries in row i of $\boldsymbol{c}_{d,j}$ and $\boldsymbol{c}_{d,j'}$, at least one entry is 0, and their product must be 0.

Claim 2: The sum of entries in each row of $\boldsymbol{\Psi}_d$ is 1, and $\sum_{j=1}^{|\mathcal{A}_d|} \boldsymbol{c}_{d,j} = \mathbf{1}$ a column vector of 1's with matching size. This is a direct consequence of each row $\boldsymbol{\psi}_d(\boldsymbol{a}^i)^\intercal$ being a one-hot vector. In other words, $\mathbf{1} \in \mathrm{colspace}(\boldsymbol{\Psi}_d)$.

Claim 3: The columns of $\boldsymbol{\Psi}$ are not linearly independent. This is because there is not a unique way to write $\mathbf{1}$ as a linear combination of the columns of $\boldsymbol{\Psi}$. For example, $\sum_{j=1}^{|\mathcal{A}_d|} \boldsymbol{c}_{d,j} = \sum_{j=1}^{|\mathcal{A}_{d'}|} \boldsymbol{c}_{d',j} = \mathbf{1}$ for some $d' \neq d$, where we used the columns of $\boldsymbol{\Psi}_d$ and $\boldsymbol{\Psi}_{d'}$.

Claim 4: $\mathbf{1} \notin \mathrm{colspace}(\tilde{\boldsymbol{\Psi}}_1 \cdots \tilde{\boldsymbol{\Psi}}_D)$ because the first entry of every column vector in any $\tilde{\boldsymbol{\Psi}}_d$ is 0 and no linear combination of them can result in a 1. Consequently, $\mathbf{1} \notin \mathrm{colspace}(\tilde{\boldsymbol{\Psi}}_d)$ for any d.

Claim 5: $\boldsymbol{c}_{d,1} \notin \mathrm{colspace}(\mathbf{1}, \tilde{\boldsymbol{\Psi}}_{d'} : d' \neq d)$, where $\boldsymbol{c}_{d,1}$ is the column removed from $\boldsymbol{\Psi}_d$ to construct $\tilde{\boldsymbol{\Psi}}_d$. This can also be seen from the first entry of the column vector: the first entry of $\boldsymbol{c}_{d,1}$ is 1, and all columns of $\tilde{\boldsymbol{\Psi}}_{d'} : d' \neq d$ have the first entry being 0.

Claim 6: $\boldsymbol{c}_{d,j} \notin \mathrm{colspace}(\mathbf{1}, \tilde{\boldsymbol{\Psi}}_1 \cdots \tilde{\boldsymbol{\Psi}}_D \setminus \{\boldsymbol{c}_{d,j}\})$ for $j > 1$. By expressing $\boldsymbol{c}_{d,j} = (\mathbf{1} - \sum_{j'=2, j' \neq j}^{|\mathcal{A}_d|} \boldsymbol{c}_{d,j'}) + (-\boldsymbol{c}_{d,1})$, we observe that the first part of the sum lies in the column space, while the second part does not (from the previous claim, $\boldsymbol{c}_{d,1}$ is not in the column space of $\tilde{\boldsymbol{\Psi}}_{d'}$ where $d' \neq d$; this is because within $\tilde{\boldsymbol{\Psi}}_d$, the only way is $\boldsymbol{c}_{d,1} = \mathbf{1} - \sum_{j'=2}^{|\mathcal{A}_d|} \boldsymbol{c}_{d,j'}$ and we have excluded one of the columns $\boldsymbol{c}_{d,j}$ from the column space).

Combining these claims implies that each column of $\tilde{\boldsymbol{\Psi}}$ cannot be expressed as a linear combination of all other columns, and thus $\tilde{\boldsymbol{\Psi}}$ has full column rank, $\mathrm{rank}(\tilde{\boldsymbol{\Psi}}) = \mathrm{ncols}(\tilde{\boldsymbol{\Psi}}) = 1 + \sum_{d=1}^{D} (|\mathcal{A}_d| - 1)$. It follows that $\tilde{\boldsymbol{\Psi}}$ contains the linearly independent subset of columns from $\boldsymbol{\Psi}$, and their column spaces and ranks are equal.

$\boldsymbol{\Psi}\boldsymbol{\Psi}^+$ and $\tilde{\boldsymbol{\Psi}}\tilde{\boldsymbol{\Psi}}^+$ are orthogonal projection matrices onto the column space of $\boldsymbol{\Psi}$ and $\tilde{\boldsymbol{\Psi}}$, respectively. Since $\mathrm{colspace}(\boldsymbol{\Phi}) = \mathrm{colspace}(\tilde{\boldsymbol{\Psi}})$, it follows that $\boldsymbol{\Psi}\boldsymbol{\Psi}^+ = \tilde{\boldsymbol{\Psi}}\tilde{\boldsymbol{\Psi}}^+$. □

B.5 A Necessary Condition

Consider the matrix form of the Bellman equation (cf. Sec 2 of Lagoudakis and Parr [60]):

$$\boldsymbol{Q} = \boldsymbol{R} + \gamma \boldsymbol{P}^\pi \boldsymbol{Q}$$

where $\boldsymbol{Q} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ is a vector containing the Q-values for all state-action pairs, $\boldsymbol{R} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, and $\boldsymbol{P}^\pi \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}||\mathcal{A}|}$ is the (s, a)-transition matrix induced by the MDP and policy π. Solving this

equation gives us the Q-function in closed form:

$$\boldsymbol{Q} = (\boldsymbol{I} - \gamma \boldsymbol{P}^{\pi})^{-1} \boldsymbol{R} \tag{5}$$

where $\boldsymbol{I} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}||\mathcal{A}|}$.

To derive a necessary condition, we start by assuming that the Q-function is representable by the linear parameterization, i.e., there exists $\boldsymbol{W} \in \mathbb{R}^{(\sum_{d=1}^{D} |\mathcal{A}_d|) \times |\mathcal{S}|}$ such that $\mathrm{vec}^{-1}_{|\mathcal{A}| \times |\mathcal{S}|}(\boldsymbol{Q}) = \boldsymbol{\Psi} \boldsymbol{W}$. Here, $\mathrm{vec}^{-1}_{|\mathcal{A}| \times |\mathcal{S}|}$ is the inverse vectorization operator that reshapes the vector of all Q-values into a matrix of size $|\mathcal{A}| \times |\mathcal{S}|$, and $\boldsymbol{\Psi} \in \{0,1\}^{|\mathcal{A}| \times (\sum_{d=1}^{D} |\mathcal{A}_d|)}$ is defined in Appendix B.4. Substituting Eqn. (5) into the premise gives us a necessary condition: if there exists $\boldsymbol{W} \in \mathbb{R}^{(\sum_{d=1}^{D} |\mathcal{A}_d|) \times |\mathcal{S}|}$ such that

$$\mathrm{vec}^{-1}_{|\mathcal{A}| \times |\mathcal{S}|} \left((\boldsymbol{I} - \gamma \boldsymbol{P}^{\pi})^{-1} \boldsymbol{R} \right) = \boldsymbol{\Psi} \boldsymbol{W}$$

Unfortunately, unlike the sufficient conditions in Theorem 1 (and Proposition 7), this necessary condition is not as clean and likely not verifiable in most settings. The matrix inverse and vec^{-1} reshaping operation make it challenging to further manipulate the expression. This highlights the non-trivial nature of the problem.

B.6 Variance Reduction in the Bandit Setting

Background on Rademacher complexity. Let $\mathcal{F}$ be a family of functions mapping from $\mathbb{R}^d$ to $\mathbb{R}$. The empirical Rademacher complexity of $\mathcal{F}$ for a sample $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ is defined by

$$\widehat{\mathfrak{R}}_{\mathcal{S}}(\mathcal{F}) = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i f(\mathbf{x}_i) \right],$$

where $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_m]$ is a vector of i.i.d. Rademacher variables, i.e., independent uniform r.v.s taking values in $\{-1, +1\}$.

For a matrix $\mathbf{M} \in \mathbb{R}^{m \times D}$, define the (p,q)-group norm as the q-norm of the p-norm of the columns of $\mathbf{M}$, that is $\|\mathbf{M}\|_{p,q} = \|[\|\mathbf{M}_1\|_p, \cdots, \|\mathbf{M}_D\|_p]\|_q$, where $\mathbf{M}_j$ is the j-th column of $\mathbf{M}$.

In Awasthi et al. [61], Theorem 2 stated that: let $\mathcal{F} = \{f = \mathbf{w}^{\intercal}\mathbf{x} : \|\mathbf{w}\|_p \leq A\}$ be a family of linear functions defined over $\mathbb{R}^d$ with bounded weight in ℓ_2-norm, then the empirical Rademacher complexity of $\mathcal{F}$ for a sample $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ satisfies the following lower bound (where $\mathbf{X} = [\mathbf{x}_1 \ldots \mathbf{x}_m]^{\intercal}$):

$$\widehat{\mathfrak{R}}_{\mathcal{S}}(\mathcal{F}) \geq \frac{A}{\sqrt{2}m} \|\mathbf{X}\|_{2,2}.$$

Proof for Proposition 5. For the sake of argument, we consider the one-timestep bandit setting; extension to the sequential setting can be similarly derived following Chen and Jiang [16], Duan et al. [14]. Let the true generative model be $\boldsymbol{Q}^* = \boldsymbol{\Psi}\boldsymbol{r} + \boldsymbol{\psi}_{\text{Interact}} r_{\text{Interact}}$ (details in Appendix B.8). We formally show the reduction in the variance of the estimators, by comparing the lower bound of their respective empirical Rademacher complexities. A smaller Rademacher complexity translates into lower variance estimators.

Suppose we obtain a sample of m actions and apply the linear approximation. Our approach for factored action space corresponds to the matrix $\boldsymbol{X} \in \{0,1\}^{m \times (\sum_d |\mathcal{A}_d|)}$, obtained by stacking the corresponding rows of $\boldsymbol{\Psi}$ (recall Definition 4). The complete, combinatorial action space corresponds to the matrix $\boldsymbol{X}' = [\boldsymbol{X}, \boldsymbol{x}_{\text{Interact}}] \in \{0,1\}^{m \times (1 + \sum_d |\mathcal{A}_d|)}$ by adding the corresponding rows of $\boldsymbol{\psi}_{\text{Interact}}$. By definition, $\|\boldsymbol{X}\|_{p,q} < \|\boldsymbol{X}'\|_{p,q}$, since the former drops a column with non-zero norm that exists in the latter.

Consider the following two function families, for the factored action space and the complete action space respectively:

$$\mathcal{F}_{\text{F}} = \{f = \mathbf{w}_{\text{F}}^{\intercal}\mathbf{x} : \|\mathbf{w}_{\text{F}}\|_2 \leq A\}$$
$$\mathcal{F}_{\text{C}} = \{f = \mathbf{w}_{\text{C}}^{\intercal}\mathbf{x}' : \|\mathbf{w}_{\text{C}}\|_2 \leq A\},$$

for some $A > 0$. A straightforward application of Theorem 2 of Awasthi et al. [61] shows that the lower bound on the Rademacher complexity of the of the factored action space is smaller than that of the complete action space, which completes our argument. □

B.7 Standardization of Rewards for the Bandit Setting (Proposition 6)

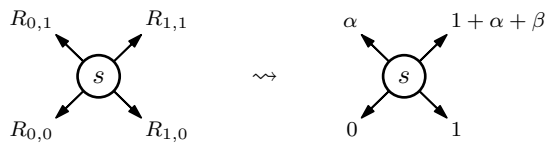


Figure 9: Standardization of rewards.

Suppose the rewards of the four arms are $[R_{0,0}, R_{0,1}, R_{1,0}, R_{1,1}]$. We can apply the following transformations to reduce any reward function to the form of $[0, \alpha, 1, 1+\alpha+\beta]$, and these transformations do not affect the least-squares solution:

- If $R_{0,0} = R_{1,0}$ and $R_{0,1} = R_{1,1}$, we can ignore x-axis sub-action as setting it to either 0 ($\leftarrow$) or 1 ($\rightarrow$) does not affect the reward. Similarly, if $R_{0,0} = R_{0,1}$ and $R_{1,0} = R_{1,1}$, we can ignore y-axis sub-action. In both cases, this reduces to a one-dimensional action space which we do not discuss further.
- Now at least one of the following is false: $R_{0,0} = R_{1,0}$ or $R_{0,1} = R_{1,1}$. If $R_{0,0} \neq R_{1,0}$, skip this step. Otherwise, it must be that $R_{0,0} = R_{1,0}$ and $R_{0,1} \neq R_{1,1}$. Swap the role of down vs. up such that the new $R_{0,0} \neq R_{1,0}$.
- If $R_{0,0} < R_{1,0}$, skip this step. Otherwise it must be that $R_{0,0} > R_{1,0}$. Swap the role of left vs. right so that $R_{0,0} < R_{1,0}$.
- If $R_{0,0} \neq 0$, subtract $R_{0,0}$ from all rewards so that the new $R_{0,0} = 0$.
- Now $R_{1,0} > R_{0,0} > 0$ must be positive. If $R_{1,0} \neq 1$, divide all rewards by $R_{1,0}$ so that the new $R_{1,0} = 1$.
- Lastly, we should have $R_{0,0} = 0$ and $R_{1,0} = 1$. Set $\alpha = R_{0,1}$ and $\beta = R_{1,1} - R_{1,0} - R_{0,1}$.

B.8 Omitted-Variable Bias in the Bandit Setting (Proposition 6)

Suppose the true generative model is

$$Q^*(\boldsymbol{a}) = \mathbb{1}_{(a_{\mathrm{x}}=\text{Left})} r_{\text{Left}} + \mathbb{1}_{(a_{\mathrm{x}}=\text{Right})} r_{\text{Right}} + \mathbb{1}_{(a_{\mathrm{y}}=\text{Down})} r_{\text{Down}} + \mathbb{1}_{(a_{\mathrm{y}}=\text{Up})} r_{\text{Up}} + \mathbb{1}_{(\boldsymbol{a}=\text{Right,Up})} r_{\text{Interact}}$$

In other words,

$$\begin{bmatrix} Q^*(\swarrow) \\ Q^*(\nwarrow) \\ Q^*(\searrow) \\ Q^*(\nearrow) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{\text{Left}} \\ r_{\text{Right}} \\ r_{\text{Down}} \\ r_{\text{Up}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} r_{\text{Interact}} \quad \rightsquigarrow \quad \boldsymbol{Q}^* = \boldsymbol{\Psi}\boldsymbol{r} + \boldsymbol{\psi}_{\text{Interact}} r_{\text{Interact}}$$

Here, $r_{\text{Left}}, r_{\text{Right}}, r_{\text{Down}}, r_{\text{Up}}, r_{\text{Interact}}$ are parameters of the generative model. Note that the matrix $[\boldsymbol{\Psi}, \boldsymbol{\psi}_{\text{Interact}}]$ has a column space of $\mathbb{R}^4$, i.e., this generative model captures every possible reward configuration of the four actions.

Applying our proposed linear approximation translates to "dropping" the interaction parameter, r_{Interact}, and estimate the remaining four parameters. This leads to a form of omitted-variable bias, which can be computed as:

$$\begin{aligned} \boldsymbol{\Psi}^+ \boldsymbol{\psi}_{\text{Interact}} r_{\text{Interact}} &= \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}^+ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} r_{\text{Interact}} \\ &= \begin{bmatrix} 3/8 & 3/8 & -1/8 & -1/8 \\ -1/8 & -1/8 & 3/8 & 3/8 \\ 3/8 & -1/8 & 3/8 & -1/8 \\ -1/8 & 3/8 & -1/8 & 3/8 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} r_{\text{Interact}} = \begin{bmatrix} -1/8 \\ 3/8 \\ -1/8 \\ 3/8 \end{bmatrix} r_{\text{Interact}} \end{aligned}$$

The biased estimate of the four parameters are:

$$\hat{\boldsymbol{r}} = \boldsymbol{r} + \boldsymbol{\Psi}^{+}\boldsymbol{\psi}_{\text{Interact}} r_{\text{Interact}} \quad \rightsquigarrow \quad \begin{bmatrix} \hat{r}_{\text{Left}} \\ \hat{r}_{\text{Right}} \\ \hat{r}_{\text{Down}} \\ \hat{r}_{\text{Up}} \end{bmatrix} = \begin{bmatrix} r_{\text{Left}} - \frac{1}{8} r_{\text{Interact}} \\ r_{\text{Right}} + \frac{3}{8} r_{\text{Interact}} \\ r_{\text{Down}} - \frac{1}{8} r_{\text{Interact}} \\ r_{\text{Up}} + \frac{3}{8} r_{\text{Interact}} \end{bmatrix}$$

and the estimated Q-values are:

$$\hat{\boldsymbol{Q}} = \begin{bmatrix} \hat{Q}(\swarrow) \\ \hat{Q}(\nwarrow) \\ \hat{Q}(\searrow) \\ \hat{Q}(\nearrow) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{\text{Left}} - \frac{1}{8} r_{\text{Interact}} \\ r_{\text{Right}} + \frac{3}{8} r_{\text{Interact}} \\ r_{\text{Down}} - \frac{1}{8} r_{\text{Interact}} \\ r_{\text{Up}} + \frac{3}{8} r_{\text{Interact}} \end{bmatrix} = \begin{bmatrix} r_{\text{Left}} + r_{\text{Down}} - \frac{1}{4} r_{\text{Interact}} \\ r_{\text{Left}} + r_{\text{Up}} + \frac{1}{4} r_{\text{Interact}} \\ r_{\text{Right}} + r_{\text{Down}} + \frac{1}{4} r_{\text{Interact}} \\ r_{\text{Right}} + r_{\text{Up}} + \frac{3}{4} r_{\text{Interact}} \end{bmatrix}$$

For the bandit problem in Figure 4a, substituting $r_{\text{Left}} + r_{\text{Down}} = 0$, $r_{\text{Left}} + r_{\text{Up}} = \alpha$, $r_{\text{Right}} + r_{\text{Down}} = 1$, and $r_{\text{Interact}} = \beta$ gives

$$\begin{bmatrix} \hat{Q}(\swarrow) \\ \hat{Q}(\nwarrow) \\ \hat{Q}(\searrow) \\ \hat{Q}(\nearrow) \end{bmatrix} = \begin{bmatrix} -\frac{1}{4}\beta \\ \alpha + \frac{1}{4}\beta \\ 1 + \frac{1}{4}\beta \\ 1 + \alpha + \frac{3}{4}\beta \end{bmatrix}$$

which is the solution we presented in Figure 4c.

C More Illustrative Examples

In this appendix, we discuss the building blocks of the examples used in the main paper and provide additional examples to support the theoretical properties presented in Section 3.

One-dimensional Chain. First, consider the chain problem depicted in Figure 10a. The agent always starts in the initial state s_0 and can take one of two possible actions: left ($a = 0$), which leads the agent to stay at s_0, or right ($a = 1$), which leads the agent to transition into s_1 and receive a reward of $+1$. After reaching the absorbing state s_1, both $a = 0$ and $a = 1$ lead the agent to stay at s_1 with zero reward. For $\gamma < 1$, a (deterministic) optimal policy is $\pi^*(s_0) = 1$, and either action can be taken in s_1. Next, we use this MDP to construct a two-dimensional problem.

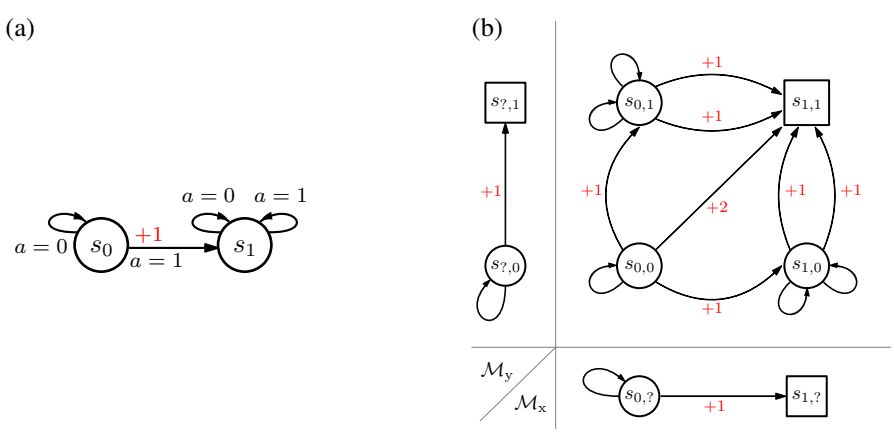


Figure 10: (a) A one-dimensional chain MDP, with an initial state s_0 and an absorbing state s_1, and two actions $a = 0$ (left) and $a = 1$ (right). (b) A two-dimensional chain MDP shown together with the component chains $\mathcal{M}_x$ and $\mathcal{M}_y$. Rewards are denoted in red. Squares $\square$ indicate absorbing states whose outgoing transition arrows are omitted. For readability, in the diagram, the states and actions are laid out following a convention similar to the Cartesian coordinate system so that the bottom left state has index $(0, 0)$, and right and up both increase the corresponding coordinate by 1.

Two-dimensional Chain. Following the construction used in Definition 1, we consider an MDP $\mathcal{M} = \mathcal{M}_{\mathrm{x}} \times \mathcal{M}_{\mathrm{y}}$ consisting of two chains (the horizontal chain $\mathcal{M}_{\mathrm{x}}$ and the vertical chain $\mathcal{M}_{\mathrm{y}}$) running in parallel, as shown in Figure 10b. Their corresponding state spaces are $\mathcal{S}_{\mathrm{x}} = \{s_{0,?}, s_{1,?}\}$ and $\mathcal{S}_{\mathrm{y}} = \{s_{?,0}, s_{?,1}\}$, which indicate the x- and y-coordinates respectively. There are 4 actions from each state, depicted by diagonal arrows $\{\swarrow, \nwarrow, \searrow, \nearrow\}$; each action $\boldsymbol{a} = [a_{\mathrm{x}}, a_{\mathrm{y}}]$ effectively leads the agent to perform a_{x} in $\mathcal{M}_{\mathrm{x}}$ and a_{y} in $\mathcal{M}_{\mathrm{y}}$. For example, taking action $\nearrow = [\rightarrow, \uparrow]$ from state $s_{0,0}$ leads the agent to transition into state $s_{1,1}$ and receive a reward of $+2$ (the sum of $+1$ from $\mathcal{M}_{\mathrm{x}}$ and $+1$ from $\mathcal{M}_{\mathrm{y}}$). For $\gamma < 1$, an optimal policy for this MDP is to always move up and right, $\pi^*(\cdot) = \nearrow = [\rightarrow, \uparrow]$, regardless of which state the agent is in.

Satisfying the Sufficient Conditions. Let $\phi_{\mathrm{x}} : \mathcal{S} \rightarrow \mathcal{S}_{\mathrm{x}}$ and $\phi_{\mathrm{y}} : \mathcal{S} \rightarrow \mathcal{S}_{\mathrm{y}}$ be the abstractions. By construction, the transition and reward functions of this MDP satisfy Eqn. (3) and (4). To apply Theorem 1, the policy must satisfy Eqn. (4). In Figure 11, we show three such policies (other policies in this category are omitted due to symmetry and transitions that have the same outcome), together with the true Q-functions (with $\gamma = 0.9$) and their decompositions in the form of Eqn. (1).

Violating the Sufficient Conditions.

- **Policy violates Eqn. (4) - Nonzero bias.** For this setting, we hold the MDP (transitions and rewards) unchanged. In Figure 12, we show seven policies that do not satisfy Eqn. (4), together with the resultant Q-function and the biased linear approximation with the non-zero approximation error.
- **Transition violates Eqn. (2) - Nonzero Bias.** Figure 13 shows an example where one transition has been modified.
- **Reward violates Eqn. (2) - Nonzero Bias.** Figure 14 shows an example where one reward has been modified.
- **Transition violates Eqn. (2), or policy violates Eqn. (4) - Zero Bias.** If $\gamma = 0$, then the Q-function is simply the immediate reward, and any conditions on the transition or policy can be forgone.
- **Reward violates Eqn. (3) - Zero Bias.** It is possible to construct reward functions adversarially such that r itself does not satisfy the condition, and yet Q can be linearly decomposed. See Figure 15 for an example.

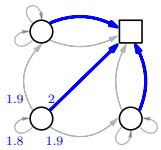
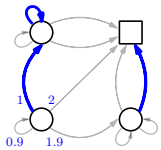
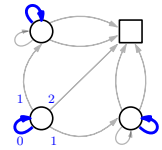
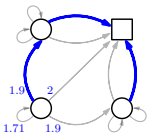
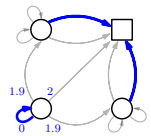
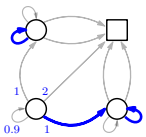
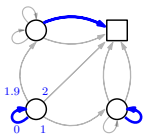
Policy π	MDP diagram	Q^π	$=$	Q_x	$+$	Q_y
Optimal policy π^*						
$\begin{matrix} s_{0,0} \\ s_{0,1} \\ s_{1,0} \\ s_{1,1} \end{matrix} \begin{bmatrix} \nearrow \\ \nearrow \\ \nearrow \\ \nearrow \end{bmatrix} = \begin{bmatrix} \rightarrow,\uparrow \\ \rightarrow,\uparrow \\ \rightarrow,\uparrow \\ \rightarrow,\uparrow \end{bmatrix}$		$\begin{matrix} & \swarrow & \nwarrow & \searrow & \nearrow \\ s_{0,0} & 1.8 & 1.9 & 1.9 & 2 \\ s_{0,1} & 0.9 & 0.9 & 1 & 1 \\ s_{1,0} & 0.9 & 1 & 0.9 & 1 \\ s_{1,1} & 0 & 0 & 0 & 0 \end{matrix}$		$\begin{matrix} & \leftarrow & \leftarrow & \rightarrow & \rightarrow \\ s_{0,?} & 0.9 & 0.9 & 1 & 1 \\ s_{0,?} & 0.9 & 0.9 & 1 & 1 \\ s_{1,?} & 0 & 0 & 0 & 0 \\ s_{1,?} & 0 & 0 & 0 & 0 \end{matrix}$		$\begin{matrix} & \downarrow & \uparrow & \downarrow & \uparrow \\ s_{?,0} & 0.9 & 1 & 0.9 & 1 \\ s_{?,1} & 0 & 0 & 0 & 0 \\ s_{?,0} & 0.9 & 1 & 0.9 & 1 \\ s_{?,1} & 0 & 0 & 0 & 0 \end{matrix}$
A non-optimal policy						
$\begin{matrix} s_{0,0} \\ s_{0,1} \\ s_{1,0} \\ s_{1,1} \end{matrix} \begin{bmatrix} \nwarrow \\ \nwarrow \\ \nearrow \\ \nearrow \end{bmatrix} = \begin{bmatrix} \leftarrow,\uparrow \\ \leftarrow,\uparrow \\ \rightarrow,\uparrow \\ \rightarrow,\uparrow \end{bmatrix}$		$\begin{matrix} & \swarrow & \nwarrow & \searrow & \nearrow \\ s_{0,0} & 0.9 & 1 & 1.9 & 2 \\ s_{0,1} & 0 & 0 & 1 & 1 \\ s_{1,0} & 0.9 & 1 & 0.9 & 1 \\ s_{1,1} & 0 & 0 & 0 & 0 \end{matrix}$		$\begin{matrix} & \leftarrow & \leftarrow & \rightarrow & \rightarrow \\ s_{0,?} & 0 & 0 & 1 & 1 \\ s_{0,?} & 0 & 0 & 1 & 1 \\ s_{1,?} & 0 & 0 & 0 & 0 \\ s_{1,?} & 0 & 0 & 0 & 0 \end{matrix}$		$\begin{matrix} & \downarrow & \uparrow & \downarrow & \uparrow \\ s_{?,0} & 0.9 & 1 & 0.9 & 1 \\ s_{?,1} & 0 & 0 & 0 & 0 \\ s_{?,0} & 0.9 & 1 & 0.9 & 1 \\ s_{?,1} & 0 & 0 & 0 & 0 \end{matrix}$
Another non-optimal policy						
$\begin{matrix} s_{0,0} \\ s_{0,1} \\ s_{1,0} \\ s_{1,1} \end{matrix} \begin{bmatrix} \swarrow \\ \nwarrow \\ \searrow \\ \nearrow \end{bmatrix} = \begin{bmatrix} \leftarrow,\downarrow \\ \leftarrow,\uparrow \\ \rightarrow,\downarrow \\ \rightarrow,\uparrow \end{bmatrix}$		$\begin{matrix} & \swarrow & \nwarrow & \searrow & \nearrow \\ s_{0,0} & 0 & 1 & 1 & 2 \\ s_{0,1} & 0 & 0 & 1 & 1 \\ s_{1,0} & 0 & 1 & 0 & 1 \\ s_{1,1} & 0 & 0 & 0 & 0 \end{matrix}$		$\begin{matrix} & \leftarrow & \leftarrow & \rightarrow & \rightarrow \\ s_{0,?} & 0 & 0 & 1 & 1 \\ s_{0,?} & 0 & 0 & 1 & 1 \\ s_{1,?} & 0 & 0 & 0 & 0 \\ s_{1,?} & 0 & 0 & 0 & 0 \end{matrix}$		$\begin{matrix} & \downarrow & \uparrow & \downarrow & \uparrow \\ s_{?,0} & 0 & 1 & 0 & 1 \\ s_{?,1} & 0 & 0 & 0 & 0 \\ s_{?,0} & 0 & 1 & 0 & 1 \\ s_{?,1} & 0 & 0 & 0 & 0 \end{matrix}$

Figure 11: Example MDPs and policies where Proposition 7 applies, for the optimal policy and two particular non-optimal policies. $\gamma = 0.9$. We show the linear decomposition of the Q-function into Q_x and Q_y. Q_x only depends on the x-coordinate of state and the sub-action that moves $\leftarrow$ or $\rightarrow$; Q_y only depends on the y-coordinate of state and the sub-action that moves $\downarrow$ or $\uparrow$.

$\pi(\mathcal{S})$	MDP diagram	$Q^\pi(s_{0,0}, \mathcal{A})$	$\hat{Q}(s_{0,0}, \mathcal{A})$
$\begin{matrix} s_{0,0} \\ s_{0,1} \\ s_{1,0} \\ s_{1,1} \end{matrix} \begin{bmatrix} \nwarrow \\ \nearrow \\ \nearrow \\ \nearrow \end{bmatrix} = \begin{bmatrix} \leftarrow,\uparrow \\ \rightarrow,\uparrow \\ \rightarrow,\uparrow \\ \rightarrow,\uparrow \end{bmatrix}$		$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 1.71 \\ 1.9 \\ 1.9 \\ 2 \end{bmatrix}$	$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 1.7325 \\ 1.8775 \\ 1.8775 \\ 2.0225 \end{bmatrix}$
$\begin{matrix} s_{0,0} \\ s_{0,1} \\ s_{1,0} \\ s_{1,1} \end{matrix} \begin{bmatrix} \swarrow \\ \nearrow \\ \nearrow \\ \nearrow \end{bmatrix} = \begin{bmatrix} \leftarrow,\downarrow \\ \rightarrow,\uparrow \\ \rightarrow,\uparrow \\ \rightarrow,\uparrow \end{bmatrix}$		$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 0 \\ 1.9 \\ 1.9 \\ 2 \end{bmatrix}$	$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 0.45 \\ 1.45 \\ 1.45 \\ 2.45 \end{bmatrix}$
$\begin{matrix} s_{0,0} \\ s_{0,1} \\ s_{1,0} \\ s_{1,1} \end{matrix} \begin{bmatrix} \searrow \\ \swarrow \\ \searrow \\ \searrow \end{bmatrix} = \begin{bmatrix} \rightarrow,\downarrow \\ \leftarrow,\downarrow \\ \rightarrow,\downarrow \\ \rightarrow,\downarrow \end{bmatrix}$		$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 0.9 \\ 1 \\ 1 \\ 2 \end{bmatrix}$	$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 0.675 \\ 1.225 \\ 1.225 \\ 1.775 \end{bmatrix}$
$\begin{matrix} s_{0,0} \\ s_{0,1} \\ s_{1,0} \\ s_{1,1} \end{matrix} \begin{bmatrix} \searrow \\ \swarrow \\ \searrow \\ \searrow \end{bmatrix} = \begin{bmatrix} \rightarrow,\downarrow \\ \leftarrow,\downarrow \\ \rightarrow,\downarrow \\ \rightarrow,\downarrow \end{bmatrix}$		$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 0 \\ 1.9 \\ 1 \\ 2 \end{bmatrix}$	$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 0.225 \\ 1.675 \\ 0.775 \\ 2.225 \end{bmatrix}$

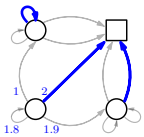
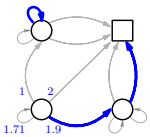
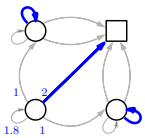
$\pi(\mathcal{S})$	MDP diagram	$Q^\pi(s_{0,0}, \mathcal{A})$	$\hat{Q}(s_{0,0}, \mathcal{A})$
$\begin{matrix} s_{0,0} \\ s_{0,1} \\ s_{1,0} \\ s_{1,1} \end{matrix} \begin{bmatrix} \nearrow \\ \nwarrow \\ \nearrow \\ \nearrow \end{bmatrix} = \begin{bmatrix} \rightarrow,\uparrow \\ \leftarrow,\uparrow \\ \rightarrow,\uparrow \\ \rightarrow,\uparrow \end{bmatrix}$		$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 1.8 \\ 1 \\ 1.9 \\ 2 \end{bmatrix}$	$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 1.575 \\ 1.225 \\ 2.125 \\ 1.775 \end{bmatrix}$
$\begin{matrix} s_{0,0} \\ s_{0,1} \\ s_{1,0} \\ s_{1,1} \end{matrix} \begin{bmatrix} \searrow \\ \nearrow \\ \nearrow \\ \nearrow \end{bmatrix} = \begin{bmatrix} \rightarrow,\downarrow \\ \leftarrow,\uparrow \\ \rightarrow,\uparrow \\ \rightarrow,\uparrow \end{bmatrix}$		$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 1.71 \\ 1 \\ 1.9 \\ 2 \end{bmatrix}$	$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 1.5075 \\ 1.2025 \\ 2.1025 \\ 1.7975 \end{bmatrix}$
$\begin{matrix} s_{0,0} \\ s_{0,1} \\ s_{1,0} \\ s_{1,1} \end{matrix} \begin{bmatrix} \nearrow \\ \nearrow \\ \searrow \\ \searrow \end{bmatrix} = \begin{bmatrix} \rightarrow,\uparrow \\ \leftarrow,\downarrow \\ \rightarrow,\downarrow \\ \rightarrow,\downarrow \end{bmatrix}$		$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 1.8 \\ 1 \\ 1 \\ 2 \end{bmatrix}$	$\begin{matrix} \nwarrow \\ \nearrow \\ \swarrow \\ \searrow \end{matrix} \begin{bmatrix} 1.35 \\ 1.45 \\ 1.45 \\ 1.55 \end{bmatrix}$

Figure 12: Example MDPs and policies where Proposition 7 does not apply because the policy violates Eqn. (4) (violations are highlighted). $\gamma = 0.9$. For example, in the first case, the policy does not take the same sub-action from $s_{0,0}$ and $s_{0,1}$ with respect to the horizontal chain $\mathcal{M}_\mathrm{x}$. Applying the linear approximation produces biased estimates $\hat{Q}$ of the true Q-function, Q^π.

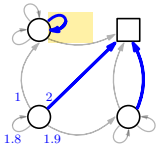
$\pi(\mathcal{S})$	MDP diagram	$Q^\pi(s_{0,0}, \mathcal{A})$	$\hat{Q}(s_{0,0}, \mathcal{A})$
$\begin{array}{c} s_{0,0} \\ s_{0,1} \\ s_{1,0} \\ s_{1,1} \end{array} \begin{bmatrix} \nearrow \\ \nearrow \\ \nearrow \\ \nearrow \end{bmatrix} = \begin{bmatrix} \rightarrow, \uparrow \\ \rightarrow, \uparrow \\ \rightarrow, \uparrow \\ \rightarrow, \uparrow \end{bmatrix}$		$\begin{bmatrix} 1.8 \\ 1.9 \\ 1 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 1.575 \\ 2.125 \\ 1.225 \\ 1.775 \end{bmatrix}$

Figure 13: Example MDPs and policies where Theorem 1 does not apply because the transition function violates Eqn. (2). $\gamma = 0.9$. In this example, the highlighted transition corresponding to the action $\nearrow = [\rightarrow, \uparrow]$ from $s_{0,1}$ does not move right ($\rightarrow$ under $\mathcal{M}_{\text{x}}$) to $s_{1,1}$ and instead moves back to state $s_{0,1}$. Applying the linear approximation produces biased estimates $\hat{Q}$ of the true Q-function, Q^π.

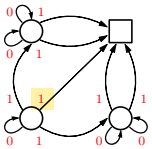
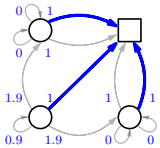
Reward function	Q-function	$Q^\pi(s_{0,0}, \mathcal{A})$	$\hat{Q}(s_{0,0}, \mathcal{A})$
		$\begin{bmatrix} 0.9 \\ 1.9 \\ 1.9 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1.375 \\ 1.425 \\ 1.425 \\ 1.475 \end{bmatrix}$

Figure 14: Example MDPs and policies where Theorem 1 does not apply because the reward function violates Eqn. (3). $\gamma = 0.9$. In this example, the reward function of the bottom left state $s_{0,0}$ does not satisfy the condition because the reward of $\nearrow$ is $1 \neq 2 = 1 + 1$. Applying the linear approximation produces biased estimates $\hat{Q}$ of the true Q-function, Q^π.

D Experiments

D.1 Sepsis Simulator - Implementation Details

When generating the datasets, we follow the default initial state distribution specified in the original implementation.

By default, we used neural networks consisting of one hidden layer with 1,000 neurons and ReLU activation to allow for function approximators with sufficient expressivity. We trained these networks using the Adam optimizer (default settings) [62] with a batch size of 64 for a maximum of 100 epochs, applying early stopping on 10% "validation data" (specific to each supervised task) with a patience of 10 epochs. We minimized the mean squared error (MSE) for regression tasks (each iteration of FQI). For FQI, we also added value clipping (to be within the range of possible returns $[-1, 1]$) when computing bootstrapping targets to ensure a bounded function class and encourage better convergence behavior [63].

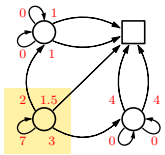
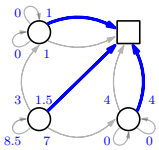
Reward function	Q-function	Q^π	$=$	Q_{x}	$+$	Q_{y}
		$\begin{array}{c} \\ s_{0,0} \\ s_{0,1} \\ s_{1,0} \\ s_{1,1} \end{array} \begin{array}{c} \begin{array}{cccc} \swarrow & \nwarrow & \searrow & \nearrow \end{array} \\ \begin{bmatrix} 8.5 & 3 & 7 & 1.5 \\ 0 & 0 & 1 & 1 \\ 0 & 4 & 0 & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$		$\begin{array}{c} \\ s_{0,?} \\ s_{0,?} \\ s_{1,?} \\ s_{1,?} \end{array} \begin{array}{c} \begin{array}{cccc} \leftarrow & \leftarrow & \rightarrow & \rightarrow \end{array} \\ \begin{bmatrix} 1.5 & 1.5 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$		$\begin{array}{c} \\ s_{?,0} \\ s_{?,1} \\ s_{?,0} \\ s_{?,1} \end{array} \begin{array}{c} \begin{array}{cccc} \downarrow & \uparrow & \downarrow & \uparrow \end{array} \\ \begin{bmatrix} 7 & 1.5 & 7 & 1.5 \\ 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$

Figure 15: Example MDPs and policies where Theorem 1 does not apply because the reward function violates Eqn. (3). $\gamma = 1$. In this example, the reward function of the bottom left state $s_{0,0}$ does not satisfy the condition because $7 + 1.5 \neq 2 + 3$. However, there exists a linear decomposition of the true Q-function, Q^π, for a particular policy denoted by bold blue arrows.

D.2 MIMIC Sepsis - Implementation Details

The RNN was trained to predict the mean of a unit-variance multivariate Gaussian that outputs the observation at subsequent timesteps, conditioned on the subsequent actions.

Hyperparameter	**Searched Settings**
RNN:	
- Embedding dimension, d_S	$\{8, 16, 32, 64, 128\}$
- Learning rate	{ 1e-5, 5e-4, 1e-4, 5e-3, 1e-3 }
BCQ (with 5 random restarts):	
- Threshold, τ	$\{0, 0.01, 0.05, 0.1, 0.3, 0.5, 0.75, 0.999\}$
- Learning rate	3e-4
- Weight decay	1e-3
- Hidden layer size	256

Table 1: Hyperparameter values used for training the RNN approximate information state as well as BCQ for offline RL. Discrete BCQ for both the baseline and factored implementation are identical except for the final Q-network layer.

D.3 MIMIC Sepsis results

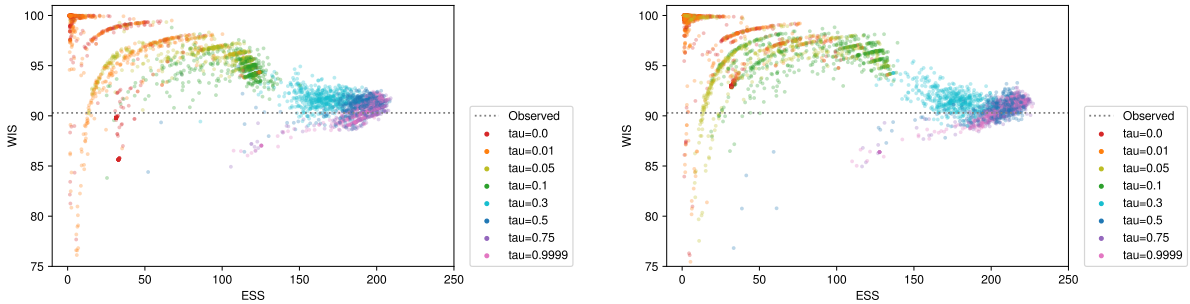


Figure 16: Validation performance (in terms of WIS and ESS) for all hyperparameter settings and all checkpoints considered during model selection. Left - baseline, Right - proposed.

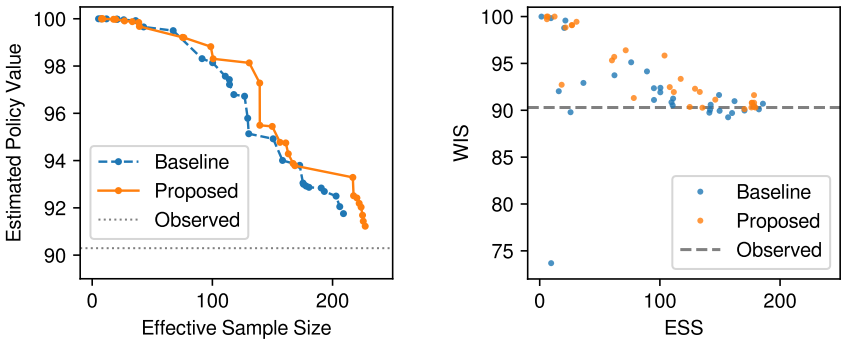


Figure 17: Left - Pareto frontiers of validation performance for the baseline and proposed approaches; Right - test performance of the candidate models that lie on the validation Pareto frontier. The validation performance largely reflects the test performance, and proposed approach outperforms the baseline in terms of test performance albeit with a bit more overlap.

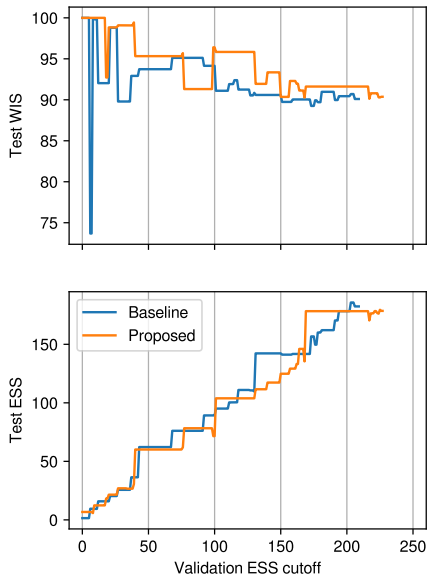


Figure 18: Model selection with different minimum ESS cutoffs. In the main paper we used ESS ≥ 200; here we sweep this threshold and compare the resultant selected policies for both the baseline and proposed approach (only using candidate models that lie on the validation Pareto frontier). In general, across the ESS cutoffs, the proposed approach outperforms the baseline in terms of test set WIS value, with comparable or slightly lower ESS.