Model-Based Reinforcement Learning in Discrete-Action Non-Markovian Reward Decision Processes

Alessandro Trapasso[1,2] , **Luca Iocchi**[2] , **Fabio Patrizi**[2]
[1]Fondazione Bruno Kessler, Trento, Italy
[2]Sapienza University of Rome, Rome, Italy
atrapasso@fbk.eu, {trapasso, iocchi, patrizi}@diag.uniroma1.it

Abstract

Many practical decision-making problems involve tasks whose success depends on the entire system history, rather than on achieving a state with desired properties. Markovian Reinforcement Learning (RL) can be inadequate for such tasks, while modeling them as non-Markovian reward decision processes (NMRDPs) enables agents to handle temporal dependencies. Existing approaches offer limited formal guarantees on both (near-)optimality and sample efficiency. We address both issues with QR-MAX, a novel model-based algorithm for NMRDPs with discrete actions that factorizes Markovian transition learning from non-Markovian reward handling via reward machines. To our knowledge, this is the first model-based RL algorithm for discrete-action NMRDPs that leverages this factorization to obtain PAC convergence to ε-optimal policies with polynomial sample complexity. We then extend QR-MAX to continuous state spaces with BUCKET-QR-MAX, a SimHash-based discretizer that preserves the same factorized structure and achieves fast and stable learning without manual gridding or function approximation. We experimentally compare our method with state-of-the-art model-based RL approaches on environments of increasing complexity, showing substantially improved sample efficiency and greater robustness in finding optimal policies.

1 Introduction

Reinforcement Learning (RL) has achieved striking success in domains that range from game playing to robotics, thanks to its ability to shape behaviour from sparse feedback. Designing such feedback, however, is difficult when the objective spans multiple temporally ordered sub-goals: specifying such tasks with a Markovian reward often yields brittle or misleading signals. Non-Markovian Reward Decision Processes (NMRDPs) address this difficulty by allowing the reward to depend on the environment history; compact logical or automata formalisms, including Reward Machines and LTL_f/LDL_f specifications, allow for encoding these rewards succinctly [Bacchus *et al.*, 1996; Thiébaux *et al.*, 2006; Brafman *et al.*, 2018; Icarte *et al.*, 2022]. Any NMRDP can be reduced to an ordinary Markov Decision Process (MDP) whose states are pairs (s, q) of environment state and automaton state [Brafman *et al.*, 2018]. Model-free agents operate in $S \times Q$ without storing a transition model [De Giacomo *et al.*, 2020; Icarte *et al.*, 2022], while model-based algorithms such as R-MAX [Brafman and Tennenholtz, 2003] can in principle learn a PAC-optimal policy on the same space [Gaon and Brafman, 2020]. Despite this theoretical effectiveness, treating (s, q) as an opaque state discards the fact that the transition kernel is still Markovian and that only the reward component is history-dependent, leading to redundant exploration.

We show that preserving this structural distinction unlocks substantial gains. By factorizing the dynamics into the Markovian environment transition $P(s'|s, a)$ and the deterministic automaton update $q' = \eta(q, s')$, we devise QR-MAX, a model-based algorithm that reuses each learned environment transition for every automaton state and retains separate optimistic bonuses. We show that QR-MAX is PAC–MDP: with high probability, all but a polynomial number of interaction steps are ε-optimal. Its sample bound has the factorized unknown-component term $U = |S||A|m_E + |S||Q|m_Q$, where m_E, m_Q are the environment and automaton knownness thresholds, instead of the product-space count $|S||Q||A|m$ incurred by applying R-MAX directly on $S \times Q$. Thus factorization avoids relearning the environment transition model separately for every automaton state. To the best of our knowledge, this is the first model-based RL algorithm for discrete-action NMRDPs that explicitly leverages the decoupling between environment and automaton dynamics to obtain PAC guarantees with improved sample-complexity bounds. Building on the same idea, we also introduce BUCKET-QR-MAX, a variant of QR-MAX combining SimHash discretisation [Charikar, 2002] with the factorised counts, which extends the approach to continuous state spaces without manual gridding and, under a standard bounded-discretisation assumption, admits a conditional PAC–MDP bound.

We evaluate QR-MAX on five maps, each combined with seven distinct Reward Machines of increasing spatial and temporal complexity, and we also include a continuous-state variant. On the benchmarks we consider, our approach is one to two orders of magnitude more sample-efficient than standard R-MAX, the reward-machine-aware QRM, and optimistic explorers such as UCBVI, PSRL and OPSRL, while

retaining formal PAC guarantees. These results suggest that QR-MAX and its continuous-state extension are practical model-based algorithms with formal sample-complexity bounds for discrete-action NMRDPs. Moreover, optimal discrete-action solutions can drive effective and efficient training in continuous-actions MDPs using hierarchical RL approaches (e.g., [Cipollone *et al.*, 2025]).

2 Related Work

RL with non-Markovian rewards has been studied via NMRDPs, Reward Machines (RMs), and temporal-logic specifications. Two strands are most relevant here: model-based RL for NMRDPs and structure-aware methods for non-Markovian tasks. Prior work applies R-MAX to NMRDPs by incrementally extending an MDP with reward-automaton states, i.e., learning in a representation akin to the R-MAX baseline we use [Gaon and Brafman, 2020]. In contrast, our approach improves sample efficiency by factorizing environment dynamics and reusing learned environment transitions across automaton states.

We also benchmark against state-of-the-art model-based exploration methods for finite MDPs. UCBVI [Azar *et al.*, 2017] matches minimax-optimal regret using empirical counts and confidence bonuses, with memory $\mathcal{O}(S^2A)$. PSRL [Osband *et al.*, 2013] and its optimistic variant OPSRL [Tiapkin *et al.*, 2022] maintain Dirichlet posteriors for transitions, yielding memory $\mathcal{O}(H, S^2A)$ (see Strens [2000], §5.1); this quadratic dependence on S makes them impractical on our largest grids, hence we omit PSRL/OPSRL when the state space is very large. On the 15×15 grid with $H{=}250$, PSRL/OPSRL need $\approx$ 50.6M entries vs. 0.204M for QR-MAX, exceeding our budget (see extended version).

A recent line of work develops structure-aware algorithms for non-Markovian tasks. Exploration in *deterministic* RMs has been studied with regret bounds exploiting automaton structure [Bourel *et al.*, 2023], and extensions to Probabilistic RMs (PRMs) have also been proposed [Bourel *et al.*, 2024; Lin and Zhang, 2025]. These methods typically optimize regret (often average-reward or finite-horizon) under a known RM/PRM, and are therefore not directly comparable to our discounted PAC-style evaluation; in our setting we provide a tabular model-based baseline and improve over R-MAX on $S \times Q$ by reusing environment transitions across automaton states. Our factorization is orthogonal to factored MDPs, where the state is represented by variables and the transition model factorises as a dynamic Bayesian network [Kearns and Koller, 1999; Guestrin *et al.*, 2003; Strehl, 2007; Osband and Roy, 2014]. Here, the underlying environment need not be factored: the gain comes from separating Markovian environment dynamics from the non-Markovian reward automaton. Other NMRDP and temporal-logic approaches either estimate alternative dynamics or focus mostly on model-free learning [Gupta *et al.*, 2021; Icarte *et al.*, 2022; De Giacomo *et al.*, 2020], whereas we focus on model-based PAC learning. Among model-free methods, one approach maximizes the probability of satisfying an LTL formula [Shao and Kwiatkowska, 2023]; it differs from our setting in focusing on infinite traces, while we use finite-trace automata (regular languages). Nonetheless, on finite traces such approaches reduce to Q-learning and QRM, which we compare against, so our experiments indirectly account for [Shao and Kwiatkowska, 2023].

Finally, a substantial body of work aims to learn the NMR model itself, e.g., [Gaon and Brafman, 2020]; this is outside our scope since we assume a given NMR specification. To the best of our knowledge, QR-MAX is the first tabular model-based RL algorithm for discounted NMRDPs that factorizes environment and reward dynamics to obtain PAC-MDP guarantees with improved sample-complexity bounds over R-MAX on $S \times Q$.

3 Background

A *Markov Decision Process* (MDP) is a tuple $\mathcal{M} = (S, A, P, R_E)$ with finite states S, actions A, transition kernel $P(s'|s,a)$, and reward function $R_E : S \times A \times S \to \mathbb{R}$. A *policy* $\rho : S \to A$ induces the *return* $v^\rho(s) = E_\rho[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots]$, where $r_i = R_E(s_i, \rho(s_i), s_{i+1})$, $s_0 = s$, and $\gamma \in [0,1]$ is the *discount factor*. The goal is to find an *optimal policy* ρ^* maximizing $v^*(s)$ for all $s \in S$. In RL, the functions P and R_E are unknown and must be learned through interaction.

RL methods are typically *model-free* or *model-based*: the former seek ρ^* without explicitly estimating P and R_E, while the latter learn both. Two representative algorithms are Q-LEARNING [Watkins and Dayan, 1992] and R-MAX [Brafman and Tennenholtz, 2003]. A central notion is the optimal state-action value function $Q^\star : S \times A \to \mathbb{R}$, where $Q^\star(s,a)$ is the expected discounted return obtained by executing a in s and then acting optimally; an optimal policy is $\rho^\star(s) \in \arg\max_a Q^\star(s,a)$. While $Q^\star$ can be computed from P and R_E (e.g., via *Value Iteration* [Sutton and Barto, 2018]), this is not directly possible in RL since P and R_E are unknown.

Q-LEARNING maintains an estimate $\widehat{Q}$, initialized arbitrarily and updated from experience using an update rule based on *Bellman's optimality condition*; with sufficient exploration, $\widehat{Q}$ approximates $Q^\star$ and induces a near-optimal policy. In contrast, R-MAX estimates P and R_E by collecting samples for state-action pairs (s,a) until they are deemed *known*; it then plans in the learned model (typically via value iteration), using optimism to drive exploration. In standard RL, R_E is *Markovian*, depending only on the current (and possibly next) state; here we consider *non-Markovian* rewards.

A *Non-Markovian Reward Decision Process* (NMRDP) is a tuple $\mathcal{N} = (\mathcal{M}, R_\mathcal{A})$, where $R_\mathcal{A} : S^+ \to \mathbb{R}$ is a *Non-Markovian Reward Function* (NMR) depending on state histories. NMRs can be specified compactly, e.g., by Reward Machines [Icarte *et al.*, 2022] or Restraining Bolts [De Giacomo *et al.*, 2020]. We unify them using a *Deterministic Finite-state Automaton* (DFA) $\mathcal{A} = (S, Q, q_0, \eta, F)$, where S is the input alphabet, Q the automaton states, q_0 the initial state, $\eta : Q \times S \to Q$ the transition function, and F the final states. We assume full observability of both the environment state and the automaton state: after executing an action, the agent observes (s', q') (equivalently, it can update q' deterministically when $\mathcal{A}$ is known). The reward is given by $R_\mathcal{A}(q, s', q')$ for the last automaton transition (q, s', q'). In

this notation, a reward machine is a deterministic automaton endowed with transition rewards, while a restraining bolt is the DFA induced by the temporal specification; acceptance can be encoded through $R_{\mathcal{A}}$, e.g., by assigning positive reward to transitions entering F. Thus F has no separate algorithmic role unless acceptance itself is part of the reward specification.

$\mathcal{A}$ compactly captures temporal properties of the history; in practice $|Q| \ll |S|$. In an NMRDP $\mathcal{N} = (\mathcal{M}, R_{\mathcal{A}})$ with $\mathcal{M} = (S, A, P, R_E)$, after a trajectory $s_0 a_0 \cdots s a s'$, the reward is $R_E(s, a, s') + R_{\mathcal{A}}(q, s', q')$, where q (resp. q') is the DFA state after consuming $s_0 \cdots s$ (resp. $s_0 \cdots s s'$). Although this makes the reward *stateful*, NMRDPs can be reduced to Markovian RL by augmenting the state with the automaton component. As shown in [Brafman *et al.*, 2018], $\mathcal{N}$ is equivalent to an MDP $\mathcal{M}' = (S', A, P', R')$ over $S' = S \times Q$, with P' capturing synchronous transitions and R' Markovian. Standard RL methods can then be applied (see, e.g., [De Giacomo *et al.*, 2020; Icarte *et al.*, 2022]).

However, this reduction treats (s, q) as an atomic state, losing the distinction between the MDP dynamics ($\mathcal{M}$) and the DFA dynamics ($\mathcal{A}$). For instance, learning transitions from (s, q) on action a does not help with (s, q'), even though the environment dynamics from s are identical. Since $\mathcal{A}$ is deterministic (and typically easier to learn), this coupling can yield redundant exploration. We investigate how to retain this opportunity by factorizing $\mathcal{N}$'s dynamics into $\mathcal{M}$'s and $\mathcal{A}$'s.

4 Decoupling Environment and Reward Dynamics

We model an NMRDP $\mathcal{N} = (\mathcal{M}, R_{\mathcal{A}})$, by combining the dynamics of the environment $\mathcal{M}$ with that of the NMR $R_{\mathcal{A}}$ and the DFA $\mathcal{A}$ it is based on, as follows: (1) when the system is reset, the environment is set to an initial state s_0, $\mathcal{A}$ is set to its initial state q_0, and a dummy DFA transition $q_0 = \eta(q_0, s_0)$ is performed, s.t. $\mathcal{A}$ consumes s_0; (2) at each step, given the current state (s, q) and the chosen action a, the environment progresses to a state s' (with probability $P(s'|s, a)$), $\mathcal{A}$ progresses to $q' = \eta(q, s')$ (with probability 1), and the returned total reward is $R_E(s, a, s') + R_{\mathcal{A}}(q, s', q')$.

The resulting transition function is captured as follows:

$$P(s', q'|s, q, a) = P(q'|s', q, s, a)\, P(s'|s, q, a),$$

where $P(s', q'|s, q, a)$, $P(q'|s', q, s, a)$, $P(s'|s, q, a)$ are the transition probability distributions of $\mathcal{N}$, $R_{\mathcal{A}}$'s automaton $\mathcal{A}$, and the environment $\mathcal{M}$, respectively. However, since $\mathcal{M}$ is Markovian: $P(s'|s, q, a) = P(s'|s, a)$, i.e., given s and a, s' is independent of the history leading $\mathcal{A}$ to q. Also, since q' depends only on q and s': $P(q'|s', q, s, a) = P(q'|s', q)$. Thus, $\mathcal{N}$'s transition model is:

$$P(s', q'|s, q, a) = P(s'|s, a) P(q'|s', q). \tag{1}$$

As for the reward function, we simply have:

$$R(s, q, a, s', q') = R_E(s, a, s') + R_{\mathcal{A}}(q, s', q'). \tag{2}$$

Note that $R_{\mathcal{A}}(q, s', q')$ does not depend on s or a, i.e., on how state (s', q') has been reached from any state (s, q).

Equations 1 and 2 describe a factorization of the system dynamics that decouples the Markovian environment dynamics ($P(s'|s, a)$, $R_E(s, a, s')$) from the non-Markovian reward expressed by the DFA ($P(q'|s', q)$, $R_{\mathcal{A}}(q, s', q')$).

Observability and RM Knowledge. We assume the agent observes the current automaton state q (as standard in RM settings via a monitor over observable propositions) and receives the RM reward component $r_{\mathcal{A}}$ from the RM monitor/wrapper. We consider two settings: QR-MAX does *not* assume knowledge of the automaton transition function η (it only receives the $q \to q'$ signal), while QR-MAXRM assumes the RM is given (including η), enabling counterfactual updates across automaton states. If q is not observable, the problem becomes partially observable and is outside our scope.

Learning Algorithms. Discrete model-based RL can be applied in the joint state space $S \times Q$ to estimate the transition and reward functions $P(s', q'|s, q, a)$ and $R(s, q, a, s', q')$. In this paper, we compare our method based on the above factorization with different model-based RL approaches that use the joint space $S \times Q$, such as that in [Brafman *et al.*, 2018; Gaon and Brafman, 2020]. As shown in the experiments, the use of the joint space does not scale well with the task's complexity, since it does not exploit the Markovian nature of the environment transitions.

4.1 QR-MAX Algorithm

In this section, we describe QR-MAX, a novel algorithm extending R-MAX to non-Markovian rewards and Markovian transitions, by exploiting the factorization described above. As shown later on, QR-MAX provides the same guarantees for near-optimal solutions as R-MAX, but exhibits a significantly higher sample efficiency. The main advantage of QR-MAX is that the terms of Equations 1 and 2, i.e., $P(s'|s, a)$, $P(q'|s', q)$, $R_E(s, a, s')$, and $R_{\mathcal{A}}(q, s', q')$, can be estimated in parallel by setting different thresholds, t_E and t_Q, to determine when they are *known*. For $n_E(s, a)$ and $n_Q(q, s')$ the number of visits to the environment's and DFA's states, respectively, we use the following conditions to label states as known: $n_E(s, a) \geq t_E \implies \textit{Known}(s, a)$ and $n_Q(q, s') \geq t_Q \implies \textit{Known}(q, s')$.

The experience for $n_E(s, a)$ is collected independently of the state of $\mathcal{A}$ and once $\textit{Known}(s, a)$ holds and the environment transition and reward functions are estimated, respectively as $\hat{P}(s'|s, a)$ and $\hat{R}_E(s, a, s')$, these values are used also in other unseen states of $\mathcal{A}$. This decoupling yields a significant improvement in sample efficiency with a standard, monolithic formulation of the joint space. Observe, in particular, that for deterministic $\mathcal{A}$ (as is the case here), $t_Q = 1$ is sufficient to know all $\mathcal{A}$'s states, thus yielding a significant saving in time for learning $\mathcal{A}$. We keep the threshold t_Q for modularity and generality, to make the algorithm applicable also to more general machines, such as *stochastic* RMs [Corazza *et al.*, 2022].

Following R-MAX's approach, QR-MAX (Algorithm 1) initializes the Q-table QT with a maximum value (line 1) for every state-action pair (notice states now consist of s and q) and selects the next action to execute as the argmax of the current Q-table (line 8). In lines 11–26, environment transitions and rewards (τ_E, ρ_E, n_E) and automaton transitions and

Algorithm 1. QR-MAX

Input: S, Q, A states and actions, E environment,
γ discount factor, $R_{\max}$ max expected reward,
t_E, t_Q thresholds, T value iteration horizon
Output: QT optimal Q-table

$QT(s,q,a) \leftarrow R_{\max}/(1-\gamma), \forall s \in S, q \in Q, a \in A$
$\tau_E(s,a,s') \leftarrow 0, \rho_E(s,a,s') \leftarrow 0, \forall s, s' \in S, a \in A$
$n_E(s,a) \leftarrow 0, \forall s \in S, a \in A$
$\tau_Q(q,s',q') \leftarrow 0, \rho_Q(q,s',q') \leftarrow 0, \forall s' \in S, q, q' \in Q$
$n_Q(q,s') \leftarrow 0, \forall s' \in S, q \in Q$
$s, q \leftarrow E.reset()$
for $t = 1, \ldots, N_{steps}$ **do**
  $\bar{a} \leftarrow \text{argmax}_a\, QT(s,q,a)$
  $s', q', r_E, r_{\mathcal{A}}, done = E.step(\bar{a})$
  doVI = False
  if $n_E(s,\bar{a}) < t_E$ **then**
    $\tau_E(s,\bar{a},s') += 1$
    $\rho_E(s,\bar{a},s') += r_E$
    $n_E(s,\bar{a}) += 1$
    if $n_E(s,\bar{a}) = t_E$ **then**
      doVI = True
  if $n_Q(q,s') < t_Q$ **then**
    $\tau_Q(q,s',q') += 1$
    $\rho_Q(q,s',q') += r_{\mathcal{A}}$
    $n_Q(q,s') += 1$
    if $n_Q(q,s') = t_Q$ **then**
      doVI = True
  if *done* **then**
    $QT(s',q',a') \leftarrow 0, \forall a' \in A$
  if doVI **then**
    $QT \leftarrow VI(QT, \tau_E, \rho_E, n_E, \tau_Q, \rho_Q, n_Q, \gamma, T)$
  if *done* **then**
    $s, q \leftarrow E.reset()$
  else
    $s, q \leftarrow s', q'$
return QT

rewards (τ_Q, ρ_Q, n_Q) are accumulated separately, using different thresholds, t_E, t_Q. If any pair (s,a) or (q,s') reaches the corresponding target threshold (lines 15-17 and 23-25), a value iteration step (VI) is run for T iterations (line 31) to update the Q-table values. When an episode terminates, the Q-table values for the final states are set to zero (line 28) and the environment is reset (line 34). For value iteration, a standard algorithm is adopted, which uses Equations 1 and 2 to describe the process and estimate the model functions as follows, considering only *known* pairs:

$$\hat{P}(s'|s,a) = \frac{\tau_E(s,a,s')}{n_E(s,a)}, \quad \hat{P}(q'|s',q) = \frac{\tau_Q(q,s',q')}{n_Q(q,s')},$$

$$\hat{R}_E(s,a,s') = \frac{\rho_E(s,a,s')}{n_E(s,a)}, \quad \hat{R}_Q(q,s',q') = \frac{\rho_Q(q,s',q')}{n_Q(q,s')}.$$

Bellman Backup Used in VI. For known components, VI:

$$Q_{k+1}(s,q,a) = \sum_{s'} \hat{R}_E(s,a,s') + \sum_{s'} \hat{P}(s'|s,a)$$
$$\Big(\sum_{q'} \hat{R}_Q(q,s',q') + \gamma \sum_{q'} \hat{P}(q'|s',q) \max_{a'} Q_k(s',q',a')\Big). \quad (3)$$

Here $\hat{R}_E(s,a,s')$ and $\hat{R}_Q(q,s',q')$ are *reward-mass* estimates: $\hat{R}_E(s,a,s') = \hat{P}(s'|s,a)\,\hat{r}_E(s,a,s')$ and $\hat{R}_Q(q,s',q') = \hat{P}(q'|s',q)\,\hat{r}_Q(q,s',q')$. Thus Eq. (3) is identical to the canonical expected-reward backup, but written in mass form to avoid conditioning on rare successor events.

During value iteration we update $QT(s,q,a)$ only when $\text{KNOWN}_E(s,a)$ holds and, for every successor state s' with $\tau_E(s,a,s') > 0$, also $\text{KNOWN}_Q(q,s')$ holds. All other entries retain their initial optimistic value $R_{\max}/(1-\gamma)$; equivalently, any unobserved successor probability mass is assigned to an absorbing optimistic state with value $R_{\max}/(1-\gamma)$ (standard R-MAX), underpinning the PAC–MDP analysis in Section 4.2.

In contrast to RM-aware methods that assume direct access to the automaton transition function η, QR-MAX does not require η as an input: it only assumes access to the automaton-state signal, i.e., the agent observes the current automaton state q and, after each action, the updated state q' together with the reward, as formalized above. If q is not observable and the RM is unknown, the problem becomes partially observable; we leave automaton-state inference to future work.

When the DFA/RM is available, we can additionally generate counterfactual experience across automaton states and apply the same factorized updates to it, obtaining the structure-aware variant QR-MAXRM, which we also evaluate experimentally.

4.2 Theoretical Guarantees of QR-MAX

Let $X = S \times Q$, $V_{\max} = R_{\max}/(1-\gamma)$, and $U = |S||A|m_E + |S||Q|m_Q$, where U is the number of factorized components that can be unknown before becoming known. We assume one-step rewards are bounded in $[0, R_{\max}]$. For the PAC analysis, we denote the knownness thresholds by $m_E = t_E$ and $m_Q = t_Q$. The thresholds m_E, m_Q are chosen by standard concentration bounds and a union bound over the events for $R_E, P_E, R_{\mathcal{A}}$, and P_Q, so that, with probability at least $1-\delta$, every known component induces reward error at most ξ_R, environment-transition error at most ξ_E, and automaton-transition error at most ξ_Q. They are chosen so that $2(\xi_R + \gamma V_{\max}(\xi_E + \xi_Q))/(1-\gamma) + \eta_{\text{VI}} \le \epsilon$, where η_{VI} is the value-iteration truncation error. For deterministic reward machines with deterministic transition rewards, one observation suffices for each observed (q,s'), hence $m_Q = 1$ and $\xi_Q = 0$ on known automaton components.

Lemma 1 (Factored Accuracy)**.** *On the above event, every known product component satisfies $|\widehat{R} - R| \le \xi_R$ and $\|\widehat{P}(\cdot \mid s,q,a) - P(\cdot \mid s,q,a)\|_1 \le \xi_E + \xi_Q$. Indeed, from Eq.* (1)*, $\|\widehat{P} - P\|_1 \le \|\widehat{P}_E(\cdot \mid s,a) - P_E(\cdot \mid s,a)\|_1 + \sum_{s'} \widehat{P}_E(s' \mid s,a)\|\widehat{P}_Q(\cdot \mid q,s') - P_Q(\cdot \mid q,s')\|_1 \le \xi_E + \xi_Q$.*

Lemma 2 (Near-Optimality in the Known Model)**.** *Conditioned on Lemma 1, the greedy policy computed in the optimistic model is ϵ-optimal whenever no unknown component is reached within the effective horizon. This follows from the discounted simulation bound $\|\widehat{\mathcal{T}}V - \mathcal{T}V\|_\infty \le \xi_R + \gamma V_{\max}(\xi_E + \xi_Q)$, valid for all $\|V\|_\infty \le V_{\max}$, and from the above choice of ξ_R, ξ_E, ξ_Q, and η_{VI}.*

Theorem 3 (PAC-MDP Bound)**.** *With the above thresholds, QR-MAX is PAC-MDP. With probability at least $1-\delta$, the number of interaction steps in which its greedy policy is not ϵ-optimal is bounded by the standard optimistic-model escape bound $O((HU/p)\log(U/\delta))$, where $H = \lceil (1-\gamma)^{-1}\ln(2R_{\max}/[\epsilon(1-\gamma)])\rceil$, $p = \epsilon(1-\gamma)/(2R_{\max})$, and $U = |S||A|m_E + |S||Q|m_Q$. Equivalently, the leading unknown-component dependence is factorized rather than product-based.*

Proof Sketch. Condition on the event of Lemma 1. If the optimistic greedy policy is not ϵ-optimal, Lemma 2 implies that, by the standard R-MAX escape-event argument [Brafman and Tennenholtz, 2003; Strehl *et al.*, 2009], it must reach an unknown component within horizon H with probability at least p. Each escape is charged to the first unknown component reached. Since environment components can be charged at most $|S||A|m_E$ times and automaton components at most $|S||Q|m_Q$ times, the standard PAC-MDP argument yields the stated bound. Value iteration affects computation time but does not consume environment interactions.

Product-Space R-MAX Comparison. Applied directly to the product MDP over $S \times Q$, R-MAX learns unknown components indexed by (s, q, a), giving a product-space count of order $|S||Q||A|m$. In contrast, QR-MAX learns the environment component once for each (s, a) and the automaton component once for each (q, s'), giving the factorized count $U = |S||A|m_E + |S||Q|m_Q$. Thus the expensive environment-transition estimation is not repeated for every automaton state. In deterministic reward machines, where $m_Q = 1$, the dominant learned component is typically the shared environment model, which gives the main sample-efficiency advantage over product-space R-MAX.

A complete proof is provided in the extended version.[1]

5 Bucket-QR-Max: Continuous State Spaces

QR-MAX assumes that the environment state component $s \in S$ is *discrete*. In many real-world problems the agent instead observes a continuous vector $s^{\text{cont}} \in \mathbb{R}^d$, while actions remain finite and discrete. A common way to handle continuous states is to discretise them into a finite set of "buckets" and treat each bucket as a discrete state. We follow this idea and integrate a hash-based discretisation into QR-MAX, yielding BUCKET-QR-MAX, a model-based extension that retains the structural advantages of QR-MAX (separate counts for Markovian dynamics and non-Markovian reward, optimistic value iteration) while handling continuous observations *without neural approximation*. Under a bounded-discretisation assumption on the induced buckets (formalised as Assumption A.1 in the extended version), we derive a conditional PAC-MDP bound for BUCKET-QR-MAX in the underlying continuous MDP (Theorem A.1).

Locality-Sensitive Hashing Discretisation

Let $h : \mathbb{R}^d \to \mathcal{B}$ be a family of L independent SimHash functions [Charikar, 2002], $h(\cdot) = (h_1(\cdot), \ldots, h_L(\cdot))$, where each h_ℓ projects s^{cont} onto $\{0, \ldots, 2^{d_h} - 1\}$. Given s_t^{cont} we obtain a *bucket* $b_t = h(s_t^{\text{cont}}) \in \mathcal{B} \subset \mathbb{N}^L$. The mapping is *data–dependent*: new buckets are added on-the-fly and assigned the next integer index $\iota : \mathcal{B} \to \{0, \ldots, B_t\}$. A joint observation is therefore represented as $(b, q) \in \mathcal{B} \times Q$. The automaton component remains exact and finite, while the continuous part is covered by a dynamically growing *countable* partition.

[1]Extended version: https://arxiv.org/abs/2512.14617.

Bucket-QR-Max Counters and Reward Sums

It keeps $n_{E_T}(b, a)$ for environment transitions, $n_{E_R}(b, a)$ and $R_E(b, a)$ for environment rewards, $n_{Q_T}(q, b')$ for reward-machine transitions, and $n_{Q_R}(q, b')$ and $R_Q(q, b')$ for non-Markovian rewards. The corresponding knownness conditions are $(n_{E_T}(b, a), n_{E_R}(b, a)) \geq (t_{E_T}, t_{E_R})$, $(n_{Q_T}(q, b'), n_{Q_R}(q, b')) \geq (t_{Q_T}, t_{Q_R})$. Only when *all four* conditions are met is the triple (b, a, q) marked *known* and Value Iteration triggered. The thresholds $t_{E_T}, t_{E_R}, t_{Q_T}, t_{Q_R}$ are chosen by the same PAC-MDP concentration argument as in Sec. 4.2, applied to the induced bucket MDP. The agent maintains an *optimistic* Q-table $Q[b, q, a] = R_{\max}/(1 - \gamma)$ for every discovered bucket b and runs synchronous Value Iteration over the finite abstraction on $\iota(\mathcal{B}) \times Q$ until convergence up to ε_{VI}. The tabular solver therefore keeps QR-MAX's factorised counts and an optimistic value function over $\mathcal{O}(|\mathcal{B}_t||Q||A|)$ entries.

Exploration and Optimism

Because the hash granularity is fixed *a priori*, each bucket b plays the role of a discrete state s in QR-MAX: the same factorised counters and model estimates (P_E, P_Q, R_E, R_Q) are reused, and the Q-table remains optimistic at $R_{\max}/(1 - \gamma)$ until (b, a, q) becomes known. Thus BUCKET-QR-MAX inherits exactly the same optimistic-exploration mechanism as the discrete algorithm.

Design Intuition and Conditional PAC Guarantee

For any fixed finite partition $\mathcal{B}$ of the state space, BUCKET-QR-MAX reduces to running QR-MAX on the abstract MDP with states $(b, q) \in \mathcal{B} \times Q$. In the extended version, we formalise a bounded-discretisation assumption (small bucket diameter and Lipschitz-continuous dynamics and rewards, see Assumption A.1) and show that, under this assumption, the induced bucket MDP is a close approximation of the underlying continuous MDP. The resulting conditional PAC-MDP bound (Theorem A.1) mirrors the discrete bound in Theorem 3, with $|S|$ replaced by $|\mathcal{B}|$. Intuitively, when nearby continuous states tend to fall into the same bucket and the environment is sufficiently smooth, an ε-optimal policy for the bucket MDP is also near-optimal in the original continuous MDP. In practice, SimHash does not deterministically bound bucket diameters; our PAC guarantee is conditional on the induced partition satisfying the bounded-diameter assumption, and hash collisions can be absorbed into an additional failure probability δ_{LSH} (overall success at least $1 - (\delta + \delta_{\text{LSH}})$), see extended version for details.

Relation to Bucket-R-Max and Implementation.

The *Covering-RMAX* algorithm replaces exact states in R-MAX with a balls-in-metric-space partition [Bernstein and Shimkin, 2008]. BUCKET-QR-MAX differs in two essential ways:

1. **Hash vs. metric balls.** LSH only probabilistically groups nearby states: by increasing the number of hash functions and projections, the probability that distant states collide in the same bucket can be made small. SimHash needs *no* explicit metric and updates in $O(L)$ time, whereas searching the nearest-centre set grows with

the cover size. In high-dimensional spaces, Covering-RMAX can suffer from the curse of dimensionality, as the number of metric balls required grows with the dimension, whereas for a fixed hash family SimHash maintains a fixed per-update computational cost.

2. **Decoupled NM-reward.** We preserve the QR-MAX factorization, yielding a sample complexity in $|\mathcal{B}|\,|A| + |\mathcal{B}|\,|Q|$ instead of the cubic $|\mathcal{B}|\,|Q|\,|A|$ inherited by a naive continuous R-MAX reduction.

BUCKET-QR-MAX differs from discrete QR-MAX in exactly three lines: (i) *LSH discretisation* $b \leftarrow h(s^{\text{cont}})$; (ii) four separate counters $n_{E_T}, n_{E_R}, n_{Q_T}, n_{Q_R}$, each with its own threshold; only when all four exceed their thresholds is (b, a, q) marked known and VI invoked; (iii) all counters, model estimates, and Q-values are indexed by buckets b instead of discrete states s. Full pseudocode is reported in the extended version.

6 Experiments

In this section, we compare QR-MAX and QR-MAXRM against several state-of-the-art algorithms, on different environments of increasing difficulty.[2] Our goal is to show that the factorization introduced in QR-MAX enhances sample efficiency and enables optimal solution of complex tasks, confirming the theoretical results obtained in Sec. 4.

The compared methods include: 1) model-based methods for MDPs over the joint space $S \times Q$: UCBVI-sB (simplified Bernstein bonus), UCBVI-H (Hoeffding bonus), and UCBVI-B (Bernstein bonus) from [Azar *et al.*, 2017]; PSRL [Osband *et al.*, 2013]; OPSRL [Tiapkin *et al.*, 2022]; R-MAX [Brafman and Tennenholtz, 2003]; 2) the model-free method QRM [Icarte *et al.*, 2022] for NMRDP. We do not compare directly with [Gaon and Brafman, 2020], which focuses on learning the RM, since when this is available, the approach corresponds to R-MAX, which we evaluate. Also, as already mentioned, [Shao and Kwiatkowska, 2023] is indirectly compared, corresponding, on the problems considered, to QRM. For clarity, QR-MAX uses only the observed automaton-state signal (q) and does not assume access to η, whereas QR-MAXRM assumes the RM is given (including η) and uses it for counterfactual updates across automaton states.

Unless stated otherwise, we use the authors' recommended hyper-parameters. For finite-horizon baselines that require a horizon (UCBVI, PSRL/OPSRL), we set H to the VI-optimal path length plus a small safety margin (extended version); this uses oracle knowledge to choose a favorable H for these baselines, hence the comparison is conservative w.r.t. QR-MAX. The benchmark we use for experimental comparison takes inspiration from, and extends, the Office gridworld environment [Icarte *et al.*, 2022] (full details in the extended version), used to evaluate NMRDP solutions. In all the experiments, during training, we use stochastic actions, all having a nominal outcome, i.e., one with highest probability wrt all the others.

For each experimental configuration, we compute a VI baseline by running value iteration on the full MDP model (obtained from the environment). Periodically we compare the current learned policy (greedy w.r.t. the Q- table) to the VI policy by running 10,000 evaluation episodes per policy in the stochastic environment and collecting discounted episode returns. We apply a two-sided Welch t-test to the two return samples. With early stopping enabled, in stochastic environments training terminates when the Welch t-test fails to detect a significant difference at a 10% significance level (p-value ≥ 0.1), treating the two policies as statistically indistinguishable. In deterministic settings the t-test is skipped due to zero variance. We apply this rule uniformly; additional extended version curves (e.g., fixed-budget success rates) confirm the same ordering. For visualization purposes, we run the learned policies on the corresponding deterministic testing environments, where actions always produce their nominal outcome. This simplifies comparing the learned policies with VI solutions, in terms of required timesteps to reach the goal. We report mean $\pm$ std over 10 seeds on three configurations.

Configuration 1 — `map0_exp0` (10×10). We use the simple grid world from prior work [Tiapkin *et al.*, 2022], using $H = 50$ and eight Thompson samples, following the original papers. The agent's non-Markovian task consists in starting from the top-left cell, collecting a letter in the bottom-right cell, returning to the start position to pick up a coffee, and finally delivering both items to the office in the bottom-left cell. The issued reward is 1 on task completion, 0 otherwise. Figure 1 shows the success rate: QR-MAXRM and QR-MAX achieve significantly better performance in roughly one order of magnitude fewer training steps than UCBVI, PSRL, and OPSRL, confirming that the factorised dynamics improves sample efficiency.

Configuration 2 — `map1_exp5` (Office World, 12×9). This map contains walls and ornamental objects that impose a penalty of -100 upon contact. For UCBVI and its variants we set $H = 150$. As illustrated in Figure 2, as task complexity grows, QR-MAX and QR-MAXRM maintain a gap of nearly two orders of magnitude over UCBVI-H and UCBVI-B, and about one order over QRM (which has access to the RM), R-MAX, and UCBVI-sB. PSRL and OPSRL exceed the predefined memory/CPU budget on this instance (see extended version for the resource limits).

Configuration 3 — `map4_exp6` (15×15). The third configuration is a maze-like space featuring two coffee machines: a *good* one (reward 1000) and a *regular* one (reward 1). The agent receives a reward only after picking up a coffee and delivering it to the office. Figure 3 shows that QR-MAX and QR-MAXRM converge to the optimal policy, bringing the better coffee in 57 steps, whereas QRM, UCBVI and their variants remain trapped for many training steps in the local optimum associated with the nearer coffee machine.

Evaluation of BUCKET-QR-MAX. We evaluate BUCKET-QR-MAX on a continuous-state variant of Frozen Lake and Office World (5 subtasks), using SimHash-based discretizations and the same factorized counting scheme as in the discrete case (see extended version for details). As baselines, we train a DQN on the same inputs (s^{cont}, q) and sparse non-Markovian reward, with a standard two-layer MLP and replay buffer, and we also include a controlled R-MAX variant that uses the same SimHash buckets but learns a monolithic model

[2] Code: https://github.com/Alee08/qrmax.

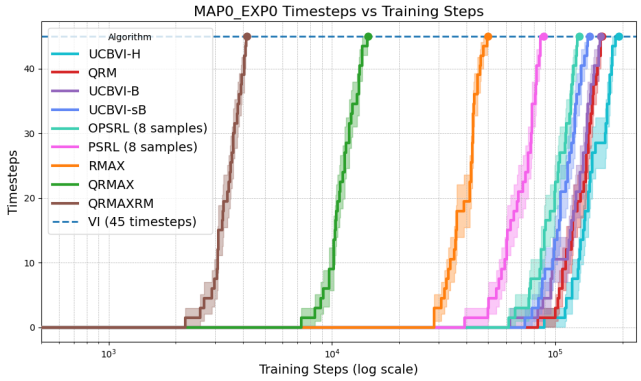

Figure 1: Configuration 1 (map0 exp0).

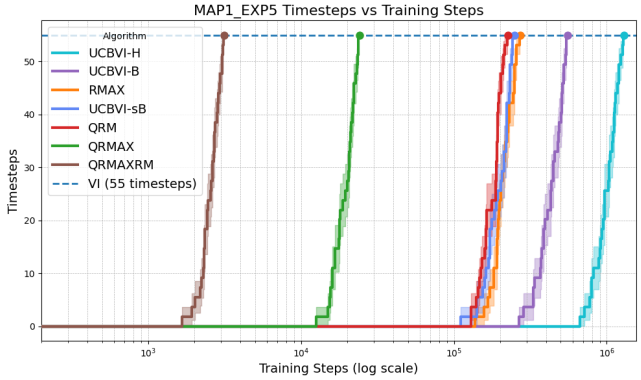

Figure 2: Configuration 2 (map1 exp5).

over joint states (b, q) (see extended version). In this setting BUCKET-QR-MAX reaches near-optimal performance in 50 ± 10 episodes, over an order of magnitude faster than DQN (which requires > 500 episodes) and with lower variance, indicating that combining SimHash discretizations with separate environment and automaton counters yields fast, stable learning in continuous state spaces without manual gridding or neural function approximation.

Summary of the Results. The experimental analysis demonstrates that, by exploiting model-based RL and the proposed factorization, QR-MAX, QR-MAXRM, and BUCKET-QR-MAX consistently achieve better sample efficiency than both model-free baselines and model-based counterparts without factorization on all the scenarios we consider. Wall-clock times remained comparable to UCBVI within the same resource limits (see extended version). We also observe different impacts of hyper-parameter tuning on solution quality. The main hyper-parameters of QR-MAX and variants are the thresholds t_E and t_Q, which control the number of samples needed to reliably estimate transitions and rewards. We set $t_E = 39$ empirically and $t_Q = 1$ because the considered RMs are deterministic. In practice, t_E is relatively easy to set because performance is weakly sensitive beyond the threshold, and training steps scale roughly linearly with t_E. In contrast, the hyper-parameters for the other methods are significantly more difficult to tune and yield higher result sensitivity. For instance, model-free RL algorithms require setting

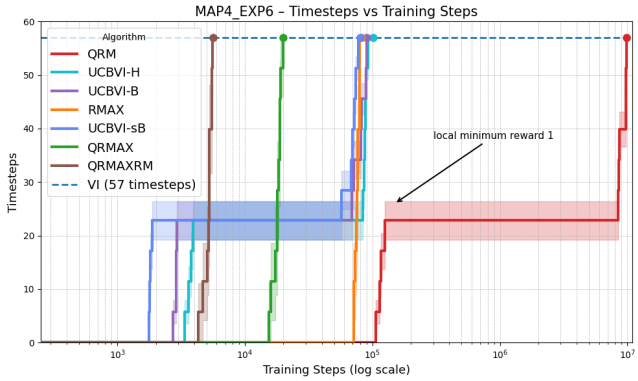

Figure 3: Configuration 3 (map4 exp6).

Map	Exp	QR-MAX	QR-MAXRM	R-MAX	QRM	UCBVI-sB	UCBVI-B	UCBVI-H	PSRL/OPSRL†
MAP0	EXP0	**14 483**	**4 150**	49 759	162 040	142 015	159 092	191 983	88 437 / 128 188
MAP1	EXP5	**24 222**	**3 125**	272 080	225 140	250 800	555 977	1 320 070	—
MAP4	EXP6	**20 150**	**5 630**	82 000	9 910 340	80 160	90 030	101 120	—

†Eight posterior samples per episode, as in prior work [Tiapkin *et al.*, 2022].

Table 1: Steps to a VI-optimal policy (mean over 10 seeds).

the learning rate, exploration factors, and initial Q-table values, resulting in high impact on the solutions; moreover, such parameters cannot easily be related to the specific problem. The other model-based RL algorithms considered in this paper include parameters such as the horizon H, whose setup requires both domain and task knowledge, and the Thompson-sampling count, which has a similar meaning of t_E and t_Q, but a higher impact especially on the algorithm's memory requirements.

7 Conclusions

In this work, we introduced QR-MAX, the first model-based RL algorithm for discrete-action NMRDPs that exploits the decoupling between Markovian environment dynamics and non-Markovian rewards. We prove that QR-MAX is PAC-MDP, i.e., all but a polynomial number of interaction steps are ε-optimal with high probability, and show on an extended Office grid-world benchmark that it substantially reduces the required interactions compared to both model-free and standard model-based approaches. We also described BUCKET-QR-MAX, which leverages SimHash to handle continuous-state NMRDPs without manual gridding or neural approximators; under mild smoothness and discretization assumptions, it admits a conditional PAC-MDP guarantee (see extended version). Across our benchmarks, QR-MAX and its variants exhibit higher sample efficiency and scalability, reduced computational cost, and easier hyper-parameter tuning than the evaluated baselines. Future directions include extending the factorization framework to deep RL techniques, handling stochastic RMs [Corazza *et al.*, 2022], learning reward models automatically, and extending the approach to multi-agent scenarios, where reward machines have already been used for cooperative and concurrent multi-agent RL [Neary *et al.*, 2021; Trapasso and Jonsson, 2025].

Acknowledgments

This research is supported by the PNRR MUR project PE0000013- FAIR.

References

[Azar *et al.*, 2017] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 263–272. PMLR, 06–11 Aug 2017.

[Bacchus *et al.*, 1996] Fahiem Bacchus, Craig Boutilier, and Adam Grove. Rewarding behaviors. In *Proc. AAAI*, pages 1160–1167, 1996.

[Bernstein and Shimkin, 2008] Andrey Bernstein and Nahum Shimkin. Adaptive aggregation for reinforcement learning with efficient exploration: Deterministic domains. In *COLT*, pages 323–334. Citeseer, 2008.

[Bourel *et al.*, 2023] Hippolyte Bourel, Anders Jonsson, Odalric-Ambrym Maillard, and Mohammad Sadegh Talebi. Exploration in Reward Machines with Low Regret. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, page 4114–4146, 2023.

[Bourel *et al.*, 2024] Hippolyte Bourel, Anders Jonsson, Odalric-Ambrym Maillard, Chenxiao Ma, and Mohammad Sadegh Talebi. Provably efficient exploration in reward machines with low regret. *CoRR*, abs/2412.19194, 2024.

[Brafman and Tennenholtz, 2003] Ronen Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3:213–231, 2003.

[Brafman *et al.*, 2018] Ronen Brafman, Giuseppe De Giacomo, and Fabio Patrizi. LTLf/LDLf non-Markovian rewards. In *Proc. AAAI*, pages 1771–1778, 2018.

[Charikar, 2002] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 380–388, New York, NY, USA, 2002. Association for Computing Machinery.

[Cipollone *et al.*, 2025] Roberto Cipollone, Marco Favorito, Flavio Maiorana, Giuseppe De Giacomo, Luca Iocchi, and Fabio Patrizi. Exploiting robot abstractions in episodic RL via reward shaping and heuristics. *Robotics and Autonomous Systems*, 193:105116, November 2025.

[Corazza *et al.*, 2022] Jan Corazza, Ivan Gavran, and Daniel Neider. Reinforcement learning with stochastic reward machines. In *Proc. AAAI*, pages 6429–6436, 2022.

[De Giacomo *et al.*, 2020] Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, and Fabio Patrizi. Restraining bolts for reinforcement learning agents. In *Proc, ICAPS*, volume 34, pages 13659–13662, 2020.

[Gaon and Brafman, 2020] Maor Gaon and Ronen Brafman. Reinforcement learning with non-Markovian rewards. In *Proc. AAAI*, pages 3980–3987, 2020.

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

[Gupta *et al.*, 2021] Gaurav Gupta, Chenzhong Yin, Jyotirmoy Deshmukh, and Paul Bogdan. Non-Markovian reinforcement learning using fractional dynamics. In *Proc. CDC*, pages 1542–1547, 2021.

[Icarte *et al.*, 2022] Rodrigo Toro Icarte, Toryn Klassen, Richard Anthony Valenzano, and Sheila McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *J. Artif. Intell. Res.*, 73:173–208, 2022.

[Kearns and Koller, 1999] Michael Kearns and Daphne Koller. Efficient reinforcement learning in factored mdps. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, page 740–747, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[Lin and Zhang, 2025] Xiaofeng Lin and Xuezhou Zhang. Efficient reinforcement learning in probabilistic reward machines. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(18):18728–18736, Apr. 2025.

[Neary *et al.*, 2021] Cyrus Neary, Zhe Xu, Bo Wu, and Ufuk Topcu. Reward machines for cooperative multi-agent reinforcement learning. In *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3–7, 2021*, pages 934–942. ACM, 2021.

[Osband and Roy, 2014] Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored mdps, 2014.

[Osband *et al.*, 2013] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling, 2013.

[Shao and Kwiatkowska, 2023] Daqian Shao and Marta Kwiatkowska. Sample efficient model-free reinforcement learning from LTL specifications with optimality guarantees. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 4180–4189. ijcai.org, 2023.

[Strehl *et al.*, 2009] Alexander Strehl, Lihong Li, and Michael Littman. Reinforcement learning in finite mdps: PAC analysis. *J. Mach. Learn. Res.*, 10:2413–2444, 2009.

[Strehl, 2007] Alexander Strehl. Model-based reinforcement learning in factored-state mdps. In *Proceedings of the 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 103–110. IEEE, 2007.

[Strens, 2000] Malcolm Strens. A bayesian framework for reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 943–950, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[Sutton and Barto, 2018] Richard Sutton and Andrew Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Thiébaux *et al.*, 2006] Sylvie Thiébaux, Charles Gretton, John Slaney, David Price, and Froduald Kabanza. Decision-theoretic planning with non-markovian rewards. *J. Artif. Intell. Res.*, 25:17–74, 2006.

[Tiapkin *et al.*, 2022] Daniil Tiapkin, Denis Belomestny, Daniele Calandriello, Eric Moulines, Remi Munos, Alexey Naumov, Mark Rowland, Michal Valko, and Pierre Menard. Optimistic posterior sampling for reinforcement learning with few samples and tight guarantees. In Alice Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[Trapasso and Jonsson, 2025] Alessandro Trapasso and Anders Jonsson. Concurrent multiagent reinforcement learning with reward machines. In *ECAI 2025 - 28th European Conference on Artificial Intelligence*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pages 3735–3742. IOS Press, 2025.

[Watkins and Dayan, 1992] Christopher Watkins and Peter Dayan. Q-learning. *Mach. Learn.*, 8(3):279–292, 1992.