Qualitative Analysis of ω-Regular Objectives on Robust MDPs

Ali Asadi, Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Mehrdad Karrabi, Ali Shafiee

Institute of Science and Technology Austria (ISTA)
{ali.asadi, krishnendu.chatterjee, ehsan.goharshady, mehrdad.karrabi, ali.shafiee}@ist.ac.at

Abstract

Robust Markov Decision Processes (RMDPs) generalize classical MDPs that consider uncertainties in transition probabilities by defining a set of possible transition functions. An objective is a set of runs (or infinite trajectories) of the RMDP, and the value for an objective is the maximal probability that the agent can guarantee against the adversarial environment. We consider (a) reachability objectives, where given a target set of states, the goal is to eventually arrive at one of them; and (b) parity objectives, which are a canonical representation for ω-regular objectives. The qualitative analysis problem asks whether the objective can be ensured with probability 1.

In this work, we study the qualitative problem for reachability and parity objectives on RMDPs without making any assumption over the structures of the RMDPs, e.g., unichain or aperiodic. Our contributions are twofold. We first present efficient algorithms with oracle access to uncertainty sets that solve qualitative problems of reachability and parity objectives. We then report experimental results demonstrating the effectiveness of our oracle-based approach on classical RMDP examples from the literature scaling up to thousands of states.

Code — https://doi.org/10.5281/zenodo.17571135
Extended version — https://arxiv.org/abs/2505.04539

1 Introduction

Robust Markov Decision Processes. Markov Decision Processes (MDPs) are the main framework for reasoning, decision making and planning under uncertainty (Puterman 1994). Solving an MDP focuses on finding a policy for the agent such that a specific trajectory-related (temporal) objective is satisfied with high probability, or the expectation of a reward function is maximized. However, the stochasticity in MDPs is usually estimated by sampling trajectories from the real-world model, resulting in that the exact transition probabilities are not known (Walley 1996; Wang and Zou 2021, 2022). Classical MDP analysis algorithms do not consider such uncertainties. To this end, Robust MDPs (RMDPs) are introduced, by considering a restricted set of transition functions and assuming that the original transition function is included (Nilim and Ghaoui 2003; Iyengar 2005). Then the main goal of RMDP analysis is to find an agent policy that maximizes the worst-case probability of a temporal property satisfaction or the worst-case expected reward with respect to all possible choices of transition function from the uncertainty set. This way, a level of *robustness* is guaranteed for the proposed policy.

Quantitative vs Logical Objectives. Most of the previous works on RMDPs, consider discounted sum or long-run average maximization objectives (Chatterjee et al. 2024; Ho, Petrik, and Wiesemann 2021; Meggendorfer, Weininger, and Wienhöft 2025; Wang et al. 2023; Nilim and Ghaoui 2003; Wang, Ho, and Petrik 2023). These objectives are referred to as *quantitative* objectives since they provide performance-related guarantees on the model. However, such objectives do not consider the logical correctness of the policies which is crucial in safety-critical applications. For example, in autonomous driving, although it is important to minimize the amount of energy consumed by the vehicle, it is more important to avoid collisions (a logical safety property). For such correctness properties, the goal is to obtain policies that ensure that the property is satisfied with high probability. Logical frameworks such as linear temporal logic (LTL), or the more general class of ω-regular objectives are used to model these properties. A canonical way of indicating ω-regular properties is through parity objectives which is a sound framework for expressing reachability, safety, and progress conditions on finite-state models (Clarke et al. 2018).

Quantitative vs Qualitative Analysis. Quantitative analysis ensures that the probability of satisfying an objective is above a given threshold $\lambda < 1$, whereas qualitative analysis focuses on ensuring the desired objective is satisfied with probability 1. The qualitative analysis is of great importance as in several applications it is required that the correct behavior happens almost-surely. For example, in the analysis of randomized embedded schedulers, an important problem is whether every thread progresses with probability 1 (Baruah et al. 1992; Chatterjee, Kößler, and Schmid 2013). On the other hand, even in applications where it might be sufficient that the correct behavior happens with probability at least $\lambda < 1$, choosing an appropriate threshold λ can be challenging, due to modeling simplifications and imprecisions, e.g., in the analysis of randomized distributed algorithms it is common to ensure correctness with probability 1 (Pogosyants, Segala, and Lynch 2000). Besides its importance in practical

applications, almost-sure convergence, like convergence in expectation, is a fundamental concept in probability theory, providing strong probabilistic guarantees (Durrett 2019).

Contributions. The above motivates the study of RMDPs with parity objectives and their qualitative analysis, which is the focus of this work. Along with parity we also consider the important special case of reachability objectives. Reachability refers to reaching a set of target states, which is a basic and fundamental objective. To this end, our contributions can be summarized as follows:

1. We present algorithms based on specific oracles on the uncertainty set (see Section 3) for qualitative analysis of reachability and parity objectives. In the extended version (Asadi et al. 2025), we show how these oracles are computed efficiently for well-known uncertainty set classes.
2. We implemented a prototype of our algorithms and conducted experiments showing their applicability and efficiency on benchmarks motivated by the literature. We compare the performance of our reachability algorithm with the state-of-the-art long-run average approach of (Chatterjee et al. 2024) (see Section 4).

Our approach is inspired by algorithms from the stochastic games literature (Chatterjee and Henzinger 2012), where both players have finite action space. However, extending the methods from stochastic games to RMDPs where the environment's action space is infinite is a challenge that we overcome in our work.

Technical Novelty. All the technical proofs can be found in the extended version (Asadi et al. 2025). We discuss the technical novelties of our work as follows:

1. In contrast to previous works that consider special classes of RMDPs, e.g. unichain and aperiodic (Wang et al. 2023), ours does not assume any structural property on the model.
2. Even in RMDPs with special classes of uncertainty, e.g. polytopic, L_1, or L_∞, the reduction of (Chatterjee et al. 2024) to stochastic games leads to an exponential blow-up of the state space. This results in an exponential-time algorithm for reachability and parity objectives. In contrast, our approach runs in polynomial time for reachability and parity objectives with constantly many priorities, and quasi-polynomial for the general case of parity objectives.

Related Works on Robust MDPs. Robust MDPs have been studied extensively in recent years. However, most of the literature on RMDPs considers discounted sum objectives (Ho, Petrik, and Wiesemann 2021; Wang et al. 2023; Ho, Petrik, and Wiesemann 2022; Behzadian, Petrik, and Ho 2021) where the goal is to find a policy that maximizes the worst-case expected sum of discounted rewards seen during a run of the RMDP. Such objectives gain their motivation from classic reinforcement learning schemes (Sutton and Barto 2018). More recently, the community is considering long-run average (a.k.a. mean-payoff) objectives (Chatterjee et al. 2024; Wang et al. 2024; Grand-Clément and Petrik 2023). It is noteworthy that given an RMDP, the qualitative analysis of reachability objectives can be reduced to finding the long-run average value of an RMDP of the same size. Several works have considered specific classes of RMDPs, e.g. interval MDPs (Chatterjee, Sen, and Henzinger 2008; Tewari and Bartlett 2007), RMDPs with convex uncertainties (Grand-Clément, Petrik, and Vieille 2023; Puggelli et al. 2013), Markov Chains with uncertainties (Sen, Viswanathan, and Agha 2006) and parametric MDPs (Winkler et al. 2019). See (Suilen et al. 2025) for a comprehensive survey.

Related Works on ω-regular and Qualitative Analysis. ω-regular and other logical objectives have been studied extensively in the context of reinforcement learning and planning (Hahn et al. 2020; Hasanbeig et al. 2019; Jothimurugan et al. 2021; Svoboda, Bansal, and Chatterjee 2024; Hau et al. 2023). For parity objectives, Zielonka's algorithm (Zielonka 1998) finds the winning set of a non-stochastic two-player parity game. Later on (Parys 2019) modified Zielonka's approach so that it runs in quasi-polynomial runtime. In (Chatterjee, Jurdzinski, and Henzinger 2003) authors show a reduction from stochastic parity games to non-stochastic parity games. (Chatterjee and Henzinger 2006) proposes a randomized sub-exponential algorithm for solving stochastic parity games. (Chatterjee and Henzinger 2012) surveys the analysis of ω-regular objectives over two-player stochastic games. Finally, (Wolff, Topcu, and Murray 2012) considers ω-regular properties on RMDPs with constant-support uncertainty and approximates the probability using value-iteration.

2 Model Definition and Preliminaries

For the rest of this paper, we denote the set of all probability distributions defined over the finite set S by $\Delta(S)$. Moreover, given a distribution P over S, we denote the support of P by $\text{supp}(P)$, the probability of sampling $B \subseteq S$ from P by $P[B]$, and if $P[B] = 1$, the restriction of P to B by $P_{|B}$. Finally, we denote the set of natural numbers as $\mathbb{N}$, Real numbers by $\mathbb{R}$ and the power set of a set S as 2^S.

MDPs. A Markov Decision Process (Puterman 1994) is defined as $M = (\mathcal{S}, \mathcal{A}, \delta)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, and $\delta \colon \mathcal{S} \times \mathcal{A} \rightharpoonup \Delta(\mathcal{S})$ is a (partial) stochastic transition relation. We denote by $\mathcal{A}(s)$ the *available* actions at state s.

The semantics of M are defined with respect to a policy $\sigma \colon (\mathcal{S} \times \mathcal{A})^* \times \mathcal{S} \to \Delta(\mathcal{A})$. Given a policy σ for M, the MDP starts at state s_0 and in the i-th turn, based on the *history* $h_i = s_0, a_0, \dots s_{i-1}, a_{i-1}$ and the current state s_i, it samples an action a_i from $\sigma(h_i, s_i)$ (over available actions $\mathcal{A}(s_i)$) and evolves to state s_{i+1} with probability $\delta(s_i, a_i)[s_{i+1}]$.

Robust MDPs. Robust Markov Decision Processes (RMDPs) (Nilim and Ghaoui 2003) extend MDPs by considering uncertainty in the transition relation. Formally, an RMDP $\mathcal{M}$ is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P})$ where $\mathcal{S}$ and $\mathcal{A}$ are the same as in the definition of MDPs, and $\mathcal{P} \colon \mathcal{S} \times \mathcal{A} \rightharpoonup 2^{\Delta(\mathcal{S})}$ is a (partial) *uncertainty set*. An entry $\mathcal{P}(s, a)$ of the uncertainty set can be characterized by constraints over real-valued variables $x_1, \dots, x_{|\mathcal{S}|}$ representing entries of distributions over $\mathcal{S}$. We assume that $\mathcal{P}(s, a)$ is compact and computable for every s, a. Later on, in section 3, we assume a certain kind of oracle is provided for $\mathcal{P}$.

The semantics of RMDPs are defined with respect to two policies: (i) an agent policy $\sigma \colon (\mathcal{S} \times \mathcal{A})^* \times \mathcal{S} \to \Delta(\mathcal{A})$, and

(ii) an adversarial policy $\rho\colon (\mathcal{S}\times\mathcal{A})^*\times\mathcal{S}\times\mathcal{A}\to\Delta(\mathcal{S})$ for the environment, where for each $h\in(\mathcal{S}\times\mathcal{A})^*$, $s\in\mathcal{S}$ and $a\in\mathcal{A}(s)$, it must hold that $\rho(h,s,a)\in\mathcal{P}(s,a)$. Given σ and ρ, the RMDP starts at s_0 and in the i-th step, given the history $h_i=s_0,a_0,\ldots,s_{i-1},a_{i-1}$ and the current state s_i, the agent samples an action a_i from $\sigma(h_i,s_i)$, then the RMDP evolves to state s_{i+1} with probability $\rho(h,s_i,a_i)[s_{i+1}]$. Intuitively, at each turn, the environment observes the action chosen by the agent and chooses a transition distribution from the uncertainty set for the RMDP to evolve accordingly. The resulting sequence $\pi=s_0,a_0,s_1,a_1,\ldots$ is called a *run* of $\mathcal{M}$. We denote the set of all runs of $\mathcal{M}$ by $\Pi_{\mathcal{M}}$.

We denote by $\mathbb{P}^{\sigma,\rho}_{\mathcal{M}}(s_0)$ the probability measure produced by the policies σ,ρ over the runs of $\mathcal{M}$ that start at s_0. This probability space is generated using cylinder sets, the details are standard and we refer the interested reader to (Baier and Katoen 2008) for more detailed discussions.

Operations over RMDPs. Given a set of states $B\subseteq\mathcal{S}$, the RMDP $\mathcal{M}\setminus B$ is defined as a tuple $(\mathcal{S}\setminus B,\mathcal{A}',\mathcal{P}')$, where (i) for every $s\in\mathcal{S}\setminus B$, available actions $\mathcal{A}'(s)$ only contains those actions from $\mathcal{A}(s)$ that cannot cause B being reached in one step, i.e., $\mathcal{A}'(s)=\{a\in\mathcal{A}(s)\mid\forall\delta\in\mathcal{P}(s,a)\colon\delta[B]=0\}$, and (ii) $\mathcal{P}'$ is the same as $\mathcal{P}$ but restricted to actions in $\mathcal{A}'$ and projected on $\mathcal{S}\setminus B$. Furthermore, the RMDP $\mathcal{M}_{|B}$ is defined as a tuple $(B,\mathcal{A}',\mathcal{P}')$, where (i) for every $s\in B$, available actions $\mathcal{A}'(s)$ only contains those actions that can reach B in one step with probability 1, i.e., $\mathcal{A}'(s)=\{a\in\mathcal{A}(s)\mid\exists\delta\in\mathcal{P}(s,a)\colon\delta[B]=1\}$, and (ii) $\mathcal{P}'(s,a)=\{\delta_{|B}\mid\delta\in\mathcal{P}(s,a)\wedge\delta[B]=1\}$.

Policy Types. An agent policy σ is called *memoryless* (or *stationary*) if it does not depend on the history, i.e. if $\sigma(h_1,s)=\sigma(h_2,s)$ for every $h_1,h_2\in(\mathcal{S}\times\mathcal{A})^*$. Moreover, σ is *pure* if it is deterministic, i.e. for each history $h\in(\mathcal{S}\times\mathcal{A})^*$ and each $s\in\mathcal{S}$, it holds that $|\sup(\sigma(h,s))|=1$. Definitions of memoryless and deterministic environment policies are analogous.

Remark. The RMDPs considered in this work are referred to as (s,a)-*rectangular* RMDPs since the environment sees both the current state and the action chosen by the agent before choosing the transition distribution.

Objectives and Values. Given an RMDP $\mathcal{M}$, an objective $\mathcal{O}$ is a set of runs of $\mathcal{M}$. For the rest of this paper, we consider two types of objectives:

1. (Reachability) Given a set $T\subseteq\mathcal{S}$, the reachability objective $Reach(T)$ is defined as the set of all runs that eventually reach a state in T:
$$Reach(T)=\{s_0,a_0,s_1,a_1,\cdots\in\Pi_{\mathcal{M}}|\exists i:s_i\in T\}$$
2. (Parity) Let $c\colon\mathcal{S}\to\{0,1,\ldots,d\}$ be a function assigning priorities to states of $\mathcal{M}$. For every run $\pi\in\Pi_{\mathcal{M}}$, let $\inf(\pi)$ be the set of states visited infinitely often in π. Then a run π is included in $Parity(c)$, if the maximum priority visited infinitely often in π is *even*:
$$Parity(c)=\{\pi\in\Pi_{\mathcal{M}}|\max\{c(\inf(\pi))\}\text{ is even}\}.$$

Note that the environment's version of the parity objective can be defined similarly, by replacing *even* with *odd*.

The value of an agent policy σ with respect to an objective $\mathcal{O}$, given an initial state s_0 is defined as
$$Val^{s_0,\sigma}_{\mathcal{M}}(\mathcal{O})=\inf_{\rho}\mathbb{P}^{\sigma,\rho}_{\mathcal{M}}(s_0)[\mathcal{O}]$$
The agent's goal is to find a policy that maximizes the value with respect to a specific objective $\mathcal{O}$. To this end, the value of the RMDP $\mathcal{M}$ given an initial state s_0 is defined as follows:
$$Val^{s_0}_{\mathcal{M}}(\mathcal{O})=\sup_{\sigma}Val^{s_0,\sigma}_{\mathcal{M}}(\mathcal{O})=\sup_{\sigma}\inf_{\rho}\mathbb{P}^{\sigma,\rho}_{\mathcal{M}}(s_0)[\mathcal{O}]$$

Problem Statement. We develop algorithms for the following two problems, given an RMDP $\mathcal{M}=(\mathcal{S},\mathcal{A},\mathcal{P})$, an initial state $s_0\in\mathcal{S}$ and a reachability or parity objective $\mathcal{O}$:

1. (Limit-Sure Analysis) Decide whether $Val^{s_0}_{\mathcal{M}}(\mathcal{O})=1$.
2. (Almost-Sure Analysis) Decide whether there exists an agent policy σ that guarantees $Val^{s_0,\sigma}_{\mathcal{M}}(\mathcal{O})=1$.

A positive answer to the almost-sure analysis problem implies a positive answer to the limit-sure analysis problem, however, the reverse does not hold in general. We will show that for RMDPs with parity objectives, the answers to the limit-sure and almost-sure analysis coincide.

3 Algorithm

In this section, we establish our algorithms for proving almost-sure reachability and parity objectives given an RMDP $\mathcal{M}$. We first introduce two oracle functions $force^{\mathcal{M}}_{\mathscr{A}}$ and $force^{\mathcal{M}}_{\mathscr{E}}$, both of type $\mathcal{S}\times 2^{\mathcal{S}}\to\{true,false\}$, which we assume are provided together with the uncertainty set.

Oracles

Given an RMDP $\mathcal{M}=(\mathcal{S},\mathcal{A},\mathcal{P})$ we assume that two Boolean functions (called *oracles*) $force^{\mathcal{M}}_{\mathscr{A}},force^{\mathcal{M}}_{\mathscr{E}}\colon\mathcal{S}\times 2^{\mathcal{S}}\to\{true,false\}$ are given as part of the model description. Informally speaking, $force^{\mathcal{M}}_{\mathscr{A}}(s,B)$ for $s\in\mathcal{S}$ and $B\subseteq\mathcal{S}$ indicates whether when the RMDP is at state s, the agent can force $\mathcal{M}$ to reach B with a non-zero probability in the next step, i.e., whether she can choose an action $a\in\mathcal{A}(s)$ such that the set B would be reached in the next step with a non-zero probability, no matter what transition function the environment chooses from $\mathcal{P}(s,a)$.

$$force^{\mathcal{M}}_{\mathscr{A}}(s,B)=\begin{cases}true & \exists a\in\mathcal{A}.\forall\delta\in\mathcal{P}(s,a).\delta[B]>0\\ false & otherwise\end{cases}$$

Similarly, $force^{\mathcal{M}}_{\mathscr{E}}(s,B)$ indicates whether whenever the RMDP is at state s, despite the action chosen by the agent, the environment can choose a transition function so that B is reached with a non-zero probability.

$$force^{\mathcal{M}}_{\mathscr{E}}(s,B)=\begin{cases}true & \forall a\in\mathcal{A}.\exists\delta\in\mathcal{P}(s,a).\delta[B]>0\\ false & otherwise\end{cases}$$

Example. Consider the RMDP in Figure 1 where action b can only be applied to s_1, s_2, s_4, and s_5 without uncertainty, i.e., the RMDP stays in the same state when action b is taken with probability 1. On the other hand, we have the following uncertainties for action a over s_1,s_2 and s_3:

$$\mathcal{P}(s_i,a)=\{\mathcal{P}\in\Delta(\mathcal{S})\mid p_i+r_i=1\wedge\|(p_i,r_i)-(\frac{1}{2},\frac{1}{2})\|_2\leq 0.2\}$$

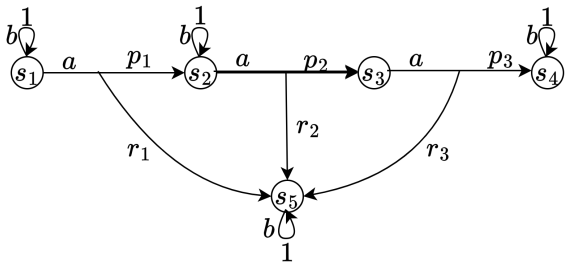

Figure 1: Running example for our algorithms.

where p_i is the probability of transitioning to s_{i+1} and r_i is the probability of reaching s_5. Then, $force^{\mathcal{M}}_{\mathscr{A}}(s_1, \{s_1\})$, $force^{\mathcal{M}}_{\mathscr{A}}(s_2, \{s_5\})$ and $force^{\mathcal{M}}_{\mathscr{E}}(s_3, \{s_4\})$ are all $true$, while $force^{\mathcal{M}}_{\mathscr{E}}(s_1, \{s_5\})$ and $force^{\mathcal{M}}_{\mathscr{E}}(s_2, \{s_3\})$ are $false$.

Practicality of Oracles. Efficient (PTIME computable) oracles can be designed for real-world uncertainty set classes such as $L_1, L_2, \ldots, L_\infty$, and linearly defined (polytopic) uncertainty sets. In the full version (Asadi et al. 2025), we show efficient oracles for each of these classes of uncertainty sets. For general uncertainty sets, the oracles can be computed by satisfiability checks.

In what follows, we assume such oracles are given as part of the model presentation for $\mathcal{M}$ and its subRMDPs with respect to the operations described in Section 2. We provide algorithms for proving almost-sure reachability and parity specifications given access to such oracles and will then analyze them by the number of oracle calls that they make.

Positive Attractors. By utilizing the oracle $force^{\mathcal{M}}_{\mathscr{A}}$, we define the *Positive Attractor* procedure $PAttr_{\mathscr{A}}$ for the agent as in Algorithm 1. The key idea is that T_i contains all the states where the agent can reach T from them in less than or equal to i steps, with a positive probability. The analogous positive attractor $PAttr_{\mathscr{E}}$ for the environment is defined by replacing $force^{\mathcal{M}}_{\mathscr{A}}$ with $force^{\mathcal{M}}_{\mathscr{E}}$ in Algorithm 1 (See full version (Asadi et al. 2025)). Lemma 1 characterizes the properties of $PAttr_{\mathscr{A}}(\mathcal{M}, T)$ and $PAttr_{\mathscr{E}}(\mathcal{M}, T)$:

Lemma 1. *Given an RMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ with oracles $force^{\mathcal{M}}_{\mathscr{A}}$ and $force^{\mathcal{M}}_{\mathscr{E}}$ and a target set $T \subseteq \mathcal{S}$, the following assertions holds:*

1. *(Correctness of $PAttr_{\mathscr{A}}$) $s \in PAttr_{\mathscr{A}}(\mathcal{M}, T)$ iff there exists a pure memoryless agent policy σ such that for all environment policies ρ we have $\mathbb{P}^{\sigma,\rho}_{\mathcal{M}}(s)[Reach(T)] > 0$;*
2. *(Correctness of $PAttr_{\mathscr{E}}$) $s \in PAttr_{\mathscr{E}}(\mathcal{M}, T)$ iff there exists a pure memoryless environment policy ρ such that for all agent policies σ, we have $\mathbb{P}^{\sigma,\rho}_{\mathcal{M}}(s)[Reach(T)] > 0$;*
3. *(Complexity) $PAttr_{\mathscr{A}}(\mathcal{M}, T)$ and $PAttr_{\mathscr{E}}(\mathcal{M}, T)$ always terminate with $O(|\mathcal{S}|^2)$ oracle calls.*

In other words, $PAttr_{\mathscr{A}}(\mathcal{M}, T)$ contains all states s of $\mathcal{M}$ where the agent has a policy to enforce reaching T with a non-zero probability starting from s, despite the policy chosen by the environment. The set $PAttr_{\mathscr{E}}(\mathcal{M}, T)$ has the same meaning but for the environment.

Algorithm 1: $PAttr_{\mathscr{A}}(\mathcal{M}, T)$ Positive Attractor for the agent

Input: RMDP $\mathcal{M}$ and Target set T.
Initialize $T_0 = T$, $i = 0$.
repeat
  $i = i + 1$
  $T_i = T_{i-1} \cup \{s \in \mathcal{S} | force^{\mathcal{M}}_{\mathscr{A}}(s, T_{i-1})\}$
until $T_i = T_{i-1}$
return T_i

Reachability

We introduce our algorithm for solving the almost-sure reachability problem in RMDPs outlined in Algorithm 2. Complete proofs are provided in the the full version (Asadi et al. 2025).

Given an RMDP $\mathcal{M}$ with state-set $\mathcal{S}$, and a target set $T \subseteq \mathcal{S}$, the algorithm proceeds in steps, removing several states in each step. Each step first computes the set $B = \mathcal{S} \setminus PAttr_{\mathscr{A}}(\mathcal{M}, T)$. These are states of the RMDP where the environment can employ a policy to never reach T from them. If $B = \varnothing$, then no matter what policy the environment chooses, every state has a non-zero probability of reaching the target. Lemma 2 shows that in such cases, the target T can be reached almost-surely from every state of the RMDP.

Lemma 2. *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ be an RMDP and $T \subseteq \mathcal{S}$. If $m > 0$ and a pure memoryless policy σ for the agent exists, such that for every state $s \in \mathcal{S}$, we have $Val^{s,\sigma}_{\mathcal{M}}(Reach(T)) \geq m$, then for every state $s \in \mathcal{S}$, we have $Val^{s,\sigma}_{\mathcal{M}}(Reach(T)) = 1$.*

In case $B \neq \varnothing$, the algorithm computes $Z = PAttr^{\mathcal{M}}_{\mathscr{E}}(B)$, which are the states where the environment can employ a policy to reach B with non-zero probability, hence reaching T with probability less than 1. Note that by removing Z from $\mathcal{M}$, the algorithm also updates $\mathcal{A}(s)$ for all $s \in \mathcal{S}$ to only contain those actions that cannot cause Z being reached in one step. The algorithm removes Z from $\mathcal{M}$ and carries on to the next loop iteration. Theorem 3 establishes the correctness and oracle complexity of the algorithm and shows that almost-sure and limit-sure coincide for reachability objectives.

Theorem 3. *Given an RMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ and a target set $T \subseteq \mathcal{S}$, the following assertions hold:*

1. *(Correctness) for all states $s \in \mathcal{S}$, we have $Val^{s}_{\mathcal{M}}(Reach(T)) = 1$, if and only if $s \in ASReach(\mathcal{M}, T)$;*
2. *(Complexity) $ASReach(\mathcal{M}, T)$ always terminates with $O(|\mathcal{S}|^3)$ oracle calls;*
3. *(Almost-Sure vs Limit-Sure) there exists a pure memoryless policy σ^* such that for all states $s \in \mathcal{S}$, we have $Val^{s}_{\mathcal{M}}(Reach(T)) = 1$, if and only if $Val^{s,\sigma^*}_{\mathcal{M}}(Reach(T)) = 1$.*

Example. Consider the RMDP in Figure 1 with the uncertainty set as in the previous example. Suppose $\{s_5\}$ is the target set. In its first iteration, the algorithm computes $B = \mathcal{S} \setminus PAttr_{\mathscr{A}}(\{s_5\}) = \{s_4\}$. This is because by taking a from s_1, s_2 and s_3, there is a non-zero probability of reaching s_5 in the next step. Next, the algorithm computes $Z = PAttr_{\mathscr{E}}(\mathcal{M}, B) = \{s_3, s_4\}$. The reason is that the only

Algorithm 2: $ASReach(\mathcal{M}, T)$ Almost-Sure Reachability

Input: RMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$, Target set T.
repeat
  $B = \mathcal{S} \setminus PAttr_{\mathscr{A}}(\mathcal{M}, T)$ {cannot reach T}
  $Z = PAttr_{\mathscr{E}}(\mathcal{M}, B)$ {cannot *almost-surely reach* T}
  $\mathcal{M} = \mathcal{M} \setminus Z$
until $B = \varnothing$
return $\mathcal{S}$

available action at s_3 is a which has a non-zero probability of reaching s_4. The algorithm then removes Z from $\mathcal{M}$ and proceeds to the next loop iteration, where only s_2 is removed. In the third loop iteration s_1 will be removed. The algorithm will then terminate by returning $\{s_5\}$.

This example shows the existence of RMDPs where the loop in Algorithm 2 is taken more than once. By extending the chain of s_i states, one can construct an RMDP, for which Algorithm 2 takes $O(|\mathcal{S}|)$ iterations before terminating.

Given that for L_p uncertainty sets, the oracles can be computed in linear time (see the full version (Asadi et al. 2025)), the time complexity of $ASReach$ in such cases is as follows:

Corollary 4. *Given an RMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ where $\mathcal{P}$ is an L_p uncertainty set, and a reachability target T, algorithm $ASReach(\mathcal{M}, T)$ runs in time $O(|\mathcal{S}|^4)$.*

Parity

In this section, we introduce our algorithm to solve almost-sure parity for the agent, inspired by the algorithm of (Zielonka 1998). The outline of our approach is formally shown in Algorithm 3. In addition, we present an analogous algorithm to solve almost-sure parity for the environment, shown in the full version (Asadi et al. 2025).

Given an RMDP $\mathcal{M}$ with state-set $\mathcal{S}$, a priority function c, and maximum priority d, the algorithm proceeds in steps, removing several states in each step. Without loss of generality, we assume that d is even (if d is odd, we increase it by 1). Each step first computes the set $B = \mathcal{S} \setminus PAttr_{\mathscr{A}}(\mathcal{M}, \mathcal{S}_d)$. These are states of the RMDP in which the environment can employ a policy to never reach the set of maximum priority states $\mathcal{S}_d$. It then constructs a new RMDP $\mathcal{M}'$ by restricting the state-space to B. Note that by restricting the state-space to B in $\mathcal{M}$, the environment only chooses transitions $\delta \in \mathcal{P}(s, a)$ if $\delta[B] = 1$. Next, it recursively computes the set of states $W_{\mathscr{E}}$ in $\mathcal{M}'$, where the environment can employ a policy to guarantee almost-surely that the maximum priority visited infinitely often is odd. Finally, it removes the states in $PAttr_{\mathscr{E}}(\mathcal{M}, W_{\mathscr{E}})$ from $\mathcal{M}$ and carries on to the next loop iteration. The intuition behind the algorithm is as follows. Every run starting from the state $s \in PAttr_{\mathscr{A}}(\mathcal{M}, \mathcal{S}_d)$ either stays in $PAttr_{\mathscr{A}}(\mathcal{M}, \mathcal{S}_d)$ forever or eventually reaches B. If it always stays in $PAttr_{\mathscr{A}}(\mathcal{M}, \mathcal{S}_d)$, then the agent has a policy of reaching $\mathcal{S}_d$ infinitely often, and consequently, almost surely satisfies the parity objective. Otherwise, it eventually reaches B. The environment does not have any incentive to visit $PAttr_{\mathscr{A}}(\mathcal{M}, \mathcal{S}_d)$ again and the agent cannot enforce it. Therefore, the run always stays in B. If $W_{\mathscr{E}} = \varnothing$, then no matter what policy the environment chooses, the agent has

Algorithm 3: $ASParity_{\mathscr{A}}(\mathcal{M}, c, d)$ agent almost-sure parity

Input: RMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$, priority function c, maximum priority d
If $\mathcal{S} = \varnothing$ **then return** $\varnothing$ **end if**
repeat
  $\mathcal{S}_d = \{s \in \mathcal{S} | c(s) = d\}$ {maximum priority states}
  $B = \mathcal{S} \setminus PAttr_{\mathscr{A}}(\mathcal{M}, \mathcal{S}_d)$ {cannot reach priority-d}
  $\mathcal{M}' = \mathcal{M}_{|B}$
  $W_{\mathscr{E}} = ASParity_{\mathscr{E}}(\mathcal{M}', c, d-1)$ {for environment}
  $G = PAttr_{\mathscr{E}}(\mathcal{M}, W_{\mathscr{E}})$ {does not satisfy parity almost-surely for agent}
  $\mathcal{M} = \mathcal{M} \setminus G$
until $W_{\mathscr{E}} \equiv \varnothing$
return $\mathcal{S}$

a policy which guarantees the parity with a non-zero probability. Lemma 5 shows that in such cases, for all states, the agent has a policy with value 1.

Lemma 5. *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ and c be an RMDP and a priority function, respectively. If $m > 0$ and a pure memoryless policy for the agent σ exists, such that for every state $s \in \mathcal{S}$, we have $Val^{s,\sigma}_{\mathcal{M}}(Parity(c)) \geq m$, then for every state $s \in \mathcal{S}$, we have $Val^{s,\sigma}_{\mathcal{M}}(Parity(c)) = 1$.*

If $W_{\mathscr{E}} \neq \varnothing$, then the environment employs a policy for the states in $PAttr_{\mathscr{E}}(\mathcal{M}, W_{\mathscr{E}})$ so that the parity objective is satisfied with probability less than 1. Removing the set $PAttr_{\mathscr{E}}(\mathcal{M}, W_{\mathscr{E}})$ from $\mathcal{M}$ results in a new RMDP with a smaller state space. The algorithm continues until $W_{\mathscr{E}} = \varnothing$. The following theorem establishes the correctness and oracle complexity of the algorithm and shows that almost-sure and limit-sure coincide for reachability objectives.

Theorem 6. *Given an RMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ and a priority function $c\colon \mathcal{S} \to \{0, \ldots, d\}$, the following assertions hold:*

1. *(Correctness) for all states $s \in \mathcal{S}$, we have $Val^{s}_{\mathcal{M}}(Parity(c)) = 1$, if and only if $s \in ASParity_{\mathscr{A}}(\mathcal{M}, c, d)$;*
2. *(Complexity) $ASParity_{\mathscr{A}}(\mathcal{M}, c, d)$ always terminates with $O\left(|\mathcal{S}|^{d+2}\right)$ oracle calls;*
3. *(Almost-Sure vs Limit-Sure) there exists a pure memoryless policy σ^* such that for all states $s \in \mathcal{S}$, we have $Val^{s}_{\mathcal{M}}(Parity(T)) = 1$, if and only if $Val^{s,\sigma^*}_{\mathcal{M}}(Parity(T)) = 1$.*

Remark 7. *In case the maximum priority is constant, the algorithm $ASParity_{\mathscr{A}}$ runs in polynomial time. Many objectives such as liveness, reach-avoid, safety, progress, etc. can be modeled as parity objectives with constant maximum priority. However, the upper bound on oracle calls is exponential with respect to the maximum priority in general. In the extended version (Asadi et al. 2025), we present a similar algorithm $EffASParity_{\mathscr{A}}$ with a quasi-polynomial upper bound over oracle calls.*

Corollary 8. *Given an RMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ and a priority function $c\colon \mathcal{S} \to \{0, \ldots, d\}$, the following hold:*

1. *(Correctness) for all states $s \in \mathcal{S}$, we have $Val^{s}_{\mathcal{M}}(Parity(c)) = 1$, if and only if $s \in EffASParity_{\mathscr{A}}(\mathcal{M}, c, d, |\mathcal{S}|, |\mathcal{S}|)$;*

2. *(Complexity) EffASParity$_{\mathscr{A}}(\mathcal{M}, c, d, |\mathcal{S}|, |\mathcal{S}|)$ always terminates with $|\mathcal{S}|^{O(\log_2 d)}$ oracle calls.*

Example. Consider the RMDP in Figure 1 with the uncertainty set as in the previous examples. The priority function c is defined as $c(s) = \begin{cases} 1 & s \in \{s_2, s_4\} \\ 2 & otherwise \end{cases}$.

First, the procedure $ASParity_{\mathscr{A}}(\mathcal{M}, c, 2)$ is called. In its first iteration, the algorithm computes $B = \mathcal{S} \setminus PAttr_{\mathscr{A}}(\mathcal{M}, \{s_1, s_3, s_5\}) = \{s_4\}$. Next, the algorithm constructs a new RMDP $\mathcal{M}' = \mathcal{M}_{|\{s_4\}}$. The algorithm then calls $ASParity_{\mathscr{E}}(\mathcal{M}', c, 1)$ which outputs $\{s_4\}$ since there are no states with even priority in this sub-RMDP. The algorithm computes $PAttr_{\mathscr{E}}(\mathcal{M}, \{s_4\}) = \{s_3, s_4\}$. This is because by taking a from s_3, there is a non-zero probability of reaching s_4. It then removes s_3 and s_4 from $\mathcal{M}$ and proceeds to the next loop iteration, where only s_2 is removed. The algorithm will then terminate by returning $\{s_1, s_5\}$.

Given the linear time complexity of our oracle computations for L_p uncertainty sets, the following complexity bounds are implied by Theorem 6 and Theorem 8:

Corollary 9. *Given an RMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ where $\mathcal{P}$ is an L_p uncertainty set, and a priority function $c\colon \mathcal{S} \to \{0, \ldots, d\}$, the following statements hold:*
1. *$ASParity(\mathcal{M}, c, d)$ runs in time $O(|\mathcal{S}|^{d+3})$.*
2. *$EffASParity(\mathcal{M}, c, d, |\mathcal{S}|, |\mathcal{S}|)$ runs in time $O(|\mathcal{S}|^{\log_2 d})$.*

4 Experimental Evaluation

We implemented a prototype of our algorithms $ASReach$, $ASParity$ and $EffASParity$ to perform experiments and observe their practical performance. We compare the performance of our almost-sure reachability algorithm $ASReach$ against the state-of-the-art policy iteration method (RPPI) on polytopic uncertainty sets (Chatterjee et al. 2024).

The purpose of this section is threefold, (i) to demonstrate the applicability and efficiency of our algorithm for almost-sure reachability ($ASReach$) on well-motivated benchmarks (ii) to show runtime improvement of $ASReach$ on polytopic uncertainty sets (L_1 and L_∞) in comparison to the baseline (RPPI), and (iii) to demonstrate and compare the performance of $ASParity$ and $EffASParity$ algorithms.

Baselines. We compare the runtime of our algorithm $ASReach$ against the *robust polytopic policy iteration* (RPPI) algorithm proposed in (Chatterjee et al. 2024). Given a polytopic RMDP, RPPI reduces the problem of computing long-run average value to that of a two-player stochastic game and uses policy iteration to find its long-run average value.

In order to compare $ASReach$ to this baseline, we slightly change our benchmarks by adding a self-loop to the target states. We then set a reward of 0 for all transitions except for the newly added self-loops which are assigned a reward of 1. With this change, the target set is almost-surely reachable from an initial state s if and only if the long-run average value of s (computed by RPPI) is equal to 1.

Implementation Details. We implement our algorithms $ASReach$, $ASParity$, and $EffASParity$ to compute the set of states where the value of the RMDP starting at those states is equal to 1 with respect to the reachability and parity objectives explained next. For computing oracles, we implemented algorithms provided in the full version (Asadi et al. 2025). We implemented the algorithms in Python 3.11 and ran the experiments on a machine with an Apple M1 chip with macOS 14.3, and 8GB of RAM with a 60-second timeout.

Benchmarks. We present two sets of benchmarks: Frozen Lake benchmarks with reachability and parity objectives and QComp benchmarks with reachability objectives. For robustness, we consider uncertainty sets with respect to L_p norms between probability distributions for $p = 1, 2$ and ∞. The details are as follows:

- **Frozen Lake.** This benchmark modifies the Frozen Lake environment in OpenAI Gym (Towers et al. 2024). Consider an $n \times n$ grid with an agent in the top left corner of it. The agent can choose to move right, left, up, or down but some cells in the grid are holes where the agent cannot enter them. Moreover, the grid is slippery, meaning that if the agent chooses to move in one direction, it may move in a perpendicular direction to the chosen action. We make this model robust by introducing uncertainty to the transitions. For the uncertainty, each state s of the model is equipped with a non-negative real number $R(s)$ such that $0 \leq R(s) \leq R_{max}$ for some upper bound R_{max}. the adversarial environment can change the probabilities such that the L_p distance of the modified and the original probability distributions is less than or equal to $R(s)$ when the RMDP is in state s. As a reachability objective, the goal of the agent is to reach the bottom right cell of the grid. We consider frozen lake models with $n = 10k$ for $1 \leq k \leq 8$ and choose the holes and the uncertainty radii $R(s)$ randomly with three different upper bounds ($R_{max} \in \{0.5, 1, 1.5\}$). Moreover, we used three different random seeds (for choosing the wholes and the uncertainty radii), leaving us with 24 instances for each configuration of p and R_{max}.
  For an ω-regular objective, suppose that there are two types of resources, one provided in the leftmost column and the other on the rightmost column of the grid. The agent's goal is to alternate between these two sets of resources forever. Similar to reachability, we chose the holes and the uncertainty radii $R(s)$ randomly with three different upper bounds. In this case, we consider models with $n = 10k$ for $1 \leq k \leq 5$ (a total of 15 instances for each configuration of p and R_{max}). Smaller models were chosen for the parity objectives due to the fact that our parity algorithms have super-polynomial complexity while the reachability algorithm runs in polynomial time.
- **QComp.** This benchmark set is inspired by the MDPs provided in the QComp repository (Hartmanns et al. 2019). The original benchmark set contains huge MDPs that require massive resources for analysis. Therefore, we only consider those MDPs that are equipped with reachability objectives and have less than or equal to 10^4 states. This leaves us with 24 instances. We introduce robustness to these benchmarks by allowing an adversarial environment to modify the probability distributions such that the L_p distance between the modified and the original probability

Benchmark	Uncertainty	*ASReach*						*RPPI*					
		$R_{max} = 0.5$		$R_{max} = 1$		$R_{max} = 1.5$		$R_{max} = 0.5$		$R_{max} = 1$		$R_{max} = 1.5$	
		count	avg. time	count	avg. time	count	avg. time	count	avg. time	count	avg. time	count	avg. time
Frozen Lake	L_1	24	10.1	24	10.8	24	10.0	4	1.0	4	5.5	1	6.2
	L_2	24	10.3	24	0.7	24	0.3	-	-	-	-	-	-
	L_∞	24	10.6	21	6.83	24	0.7	4	8.2	3	0.4	6	2.4
QComp	L_1	24	6.8	23	5.1	24	6.4	9	7.0	10	7.8	10	7.8
	L_2	23	5.4	24	7.4	24	6.7	-	-	-	-	-	-
	L_∞	23	5.4	24	6.9	24	7.0	10	6.9	10	6.8	10	6.8
Total		142	8.1	140	6.3	144	5.2	27	6.3	27	6.3	27	6.2
Solved By Both		27	0.1	27	0.1	27	0.1	27	6.3	27	6.3	27	6.2

Table 1: Runtime comparison of reachability algorithms on Frozen Lake and QComp benchmarks. The times are in seconds.

Benchmark	Uncertainty	*ASParity*						*EffASParity*					
		$R_{max} = 0.5$		$R_{max} = 1$		$R_{max} = 1.5$		$R_{max} = 0.5$		$R_{max} = 1$		$R_{max} = 1.5$	
		count	avg. time	count	avg. time	count	avg. time	count	avg. time	count	avg. time	count	avg. time
Frozen Lake	L_1	15	7.3	15	7.4	15	8.0	12	7.8	12	8.0	12	8.7
	L_2	15	8.8	15	4.8	15	0.7	12	9.0	15	4.6	15	0.6
	L_∞	15	7.6	15	10.9	15	5.6	12	8.4	15	11.6	15	6.5
Total		45	7.9	45	7.7	45	4.8	36	8.4	42	8.1	42	5.0

Table 2: Runtime comparison of parity algorithms on Frozen Lake benchmarks. The average times are in seconds.

distributions is less or equal to a global value R.

Tables Description. The results for reachability and parity algorithms are summarized in Table 1 and Table 2, respectively. The parameter p corresponds to the L_p norms used to define the uncertainty sets in the problem, with $p = 1, 2,$ or ∞, representing different geometric shapes of uncertainty sets. The parameter R_{max} denotes the maximum radius of the uncertainty sets, with $R_{max} = 0.5, 1,$ or 1.5. For each combination of p and R_{max}, the following metrics are reported: (a) count: the number of instances solved within the time limit; and (b) avg. time: the average time (in seconds) required to solve instances that were successfully solved within the time limit. Higher count values and lower avg. time values indicate better performance.

Results for Reachability. Table 1 summarizes our experiments comparing *ASReach* and the baseline approach. The results demonstrate that *ASReach* outperforms the baseline by solving 426 instances, compared to only 81 solved by the baseline. This highlights the effectiveness of our algorithm for solving the almost-sure reachability problem, outperforming the RPPI method. Moreover, while the RPPI approach is restricted to polytopic RMDPs, Table 1 shows that our method can be efficiently applied to RMDPs with L_2 uncertainty sets. An advantage of our method is that it is oracle-based, hence we expect it to solve RMDPs with higher degree L_p-based uncertainties as well. Lastly, note that on the benchmarks that were solved by both methods, the average runtime of our approach is about 0.1 seconds while the baseline takes more than 6 seconds. This shows that while the baseline only solves the simplest benchmarks, our method is faster even on these simple instances by an order of magnitude.

Results for Parity. Table 2 summarizes our experimental results comparing our two algorithms, *ASParity* and *EffASParity* for parity objectives over the frozen lake benchmarks. These results show that although the worst-case complexity of *ASParity* is worse than *EffASParity*, but in practice, *ASParity* is faster and can solve 135 instances in the designated timeout, where *EffASParity* can solve 120. Moreover, Figure 2 shows the number of solved instances over time for each algorithm. The fact that *ASParity*'s curve is higher and more to the left, shows that *ASParity* performs better on our benchmarks.

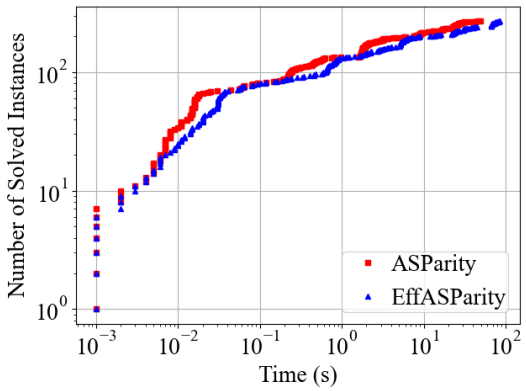

Figure 2: Runtime comparison of our parity algorithms on the Frozen Lake Model.

5 Conclusion and Future Work

In this work, we presented algorithms for the qualitative analysis of RMDPs with reachability and parity objectives. There are several interesting directions for future work. Whether the algorithmic upper bound for parity objectives can be further improved to polynomial time is a major open problem, even for turn-based games. While the present work establishes the theoretical foundations, another interesting direction is exploring efficient heuristics for parity algorithms for further scalability and practical applications.

Ethical Statement

This paper presents a primarily theoretical study focused on algorithmic foundations. As such, this work does not raise any ethical concerns.

Acknowledgements

This work was supported by ERC CoG 863818 (ForM-SMArt) and Austrian Science Fund (FWF) 10.55776/COE12. We also thank Hossein Zakerinia for his helpful feedback.

References

Asadi, A.; Chatterjee, K.; Goharshady, E. K.; Karrabi, M.; and Shafiee, A. 2025. Qualitative Analysis of ω-Regular Objectives on Robust MDPs. arXiv:2505.04539.

Baier, C.; and Katoen, J. 2008. *Principles of model checking*. MIT Press.

Baruah, S.; Koren, G.; Mao, D.; Mishra, B.; Raghunathan, A.; Rosier, L.; Shasha, D.; and Wang, F. 1992. On the competitiveness of on-line real-time task scheduling. *Real-Time Systems*, 4: 125–144.

Behzadian, B.; Petrik, M.; and Ho, C. P. 2021. Fast Algorithms for $L_-\infty$-constrained S-rectangular Robust MDPs. In *NeurIPS*, 25982–25992.

Chatterjee, K.; Goharshady, E. K.; Karrabi, M.; Novotný, P.; and Zikelic, D. 2024. Solving Long-run Average Reward Robust MDPs via Stochastic Games. In *IJCAI*, 6707–6715. ijcai.org.

Chatterjee, K.; and Henzinger, T. A. 2006. Strategy Improvement and Randomized Subexponential Algorithms for Stochastic Parity Games. In *STACS*, volume 3884 of *Lecture Notes in Computer Science*, 512–523. Springer.

Chatterjee, K.; and Henzinger, T. A. 2012. A survey of stochastic ω-regular games. *J. Comput. Syst. Sci.*, 78(2): 394–413.

Chatterjee, K.; Jurdzinski, M.; and Henzinger, T. A. 2003. Simple Stochastic Parity Games. In *CSL*, volume 2803 of *Lecture Notes in Computer Science*, 100–113. Springer.

Chatterjee, K.; Kößler, A.; and Schmid, U. 2013. Automated analysis of real-time scheduling using graph games. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, 163–172.

Chatterjee, K.; Sen, K.; and Henzinger, T. A. 2008. Model-Checking omega-Regular Properties of Interval Markov Chains. In *FoSSaCS*, volume 4962 of *Lecture Notes in Computer Science*, 302–317. Springer.

Clarke, E. M.; Henzinger, T. A.; Veith, H.; and Bloem, R., eds. 2018. *Handbook of Model Checking*. Springer.

Durrett, R. 2019. *Probability: theory and examples*, volume 49. Cambridge university press.

Grand-Clément, J.; and Petrik, M. 2023. Reducing Blackwell and Average Optimality to Discounted MDPs via the Blackwell Discount Factor. In *NeurIPS*.

Grand-Clément, J.; Petrik, M.; and Vieille, N. 2023. Beyond discounted returns: Robust Markov decision processes with average and Blackwell optimality. *CoRR*, abs/2312.03618.

Hahn, E. M.; Perez, M.; Schewe, S.; Somenzi, F.; Trivedi, A.; and Wojtczak, D. 2020. Model-Free Reinforcement Learning for Stochastic Parity Games. In *CONCUR*, volume 171 of *LIPIcs*, 21:1–21:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Hartmanns, A.; Klauck, M.; Parker, D.; Quatmann, T.; and Ruijters, E. 2019. The Quantitative Verification Benchmark Set. In *TACAS (1)*, volume 11427 of *Lecture Notes in Computer Science*, 344–350. Springer.

Hasanbeig, M.; Kantaros, Y.; Abate, A.; Kroening, D.; Pappas, G. J.; and Lee, I. 2019. Reinforcement Learning for Temporal Logic Control Synthesis with Probabilistic Satisfaction Guarantees. In *CDC*, 5338–5343. IEEE.

Hau, J. L.; Delage, E.; Ghavamzadeh, M.; and Petrik, M. 2023. On Dynamic Programming Decompositions of Static Risk Measures in Markov Decision Processes. In *NeurIPS*.

Ho, C. P.; Petrik, M.; and Wiesemann, W. 2021. Partial Policy Iteration for L1-Robust Markov Decision Processes. *J. Mach. Learn. Res.*, 22: 275:1–275:46.

Ho, C. P.; Petrik, M.; and Wiesemann, W. 2022. Robust ϕ-Divergence MDPs. In *NeurIPS*.

Iyengar, G. N. 2005. Robust Dynamic Programming. *Math. Oper. Res.*, 30(2): 257–280.

Jothimurugan, K.; Bansal, S.; Bastani, O.; and Alur, R. 2021. Compositional Reinforcement Learning from Logical Specifications. In *NeurIPS*, 10026–10039.

Meggendorfer, T.; Weininger, M.; and Wienhöft, P. 2025. Solving Robust Markov Decision Processes: Generic, Reliable, Efficient. In *AAAI*.

Nilim, A.; and Ghaoui, L. E. 2003. Robustness in Markov Decision Problems with Uncertain Transition Matrices. In *NIPS*, 839–846. MIT Press.

Parys, P. 2019. Parity Games: Zielonka's Algorithm in Quasi-Polynomial Time. In *MFCS*, volume 138 of *LIPIcs*, 10:1–10:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Pogosyants, A.; Segala, R.; and Lynch, N. 2000. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. *Distributed Computing*, 13: 155–186.

Puggelli, A.; Li, W.; Sangiovanni-Vincentelli, A. L.; and Seshia, S. A. 2013. Polynomial-Time Verification of PCTL Properties of MDPs with Convex Uncertainties. In *CAV*, volume 8044 of *Lecture Notes in Computer Science*, 527–542. Springer.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley.

Sen, K.; Viswanathan, M.; and Agha, G. 2006. Model-Checking Markov Chains in the Presence of Uncertainties. In *TACAS*, volume 3920 of *Lecture Notes in Computer Science*, 394–410. Springer.

Suilen, M.; Badings, T.; Bovy, E. M.; Parker, D.; and Jansen, N. 2025. *Robust Markov Decision Processes: A Place Where AI and Formal Methods Meet*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. A Bradford Book.

Svoboda, J.; Bansal, S.; and Chatterjee, K. 2024. Reinforcement Learning from Reachability Specifications: PAC Guarantees with Expected Conditional Distance. In *ICML*. OpenReview.net.

Tewari, A.; and Bartlett, P. L. 2007. Bounded Parameter Markov Decision Processes with Average Reward Criterion. In *COLT*, volume 4539 of *Lecture Notes in Computer Science*, 263–277. Springer.

Towers, M.; Kwiatkowski, A.; Terry, J. K.; Balis, J. U.; Cola, G. D.; Deleu, T.; Goulão, M.; Kallinteris, A.; Krimmel, M.; KG, A.; Perez-Vicente, R.; Pierré, A.; Schulhoff, S.; Tai, J. J.; Tan, H.; and Younis, O. G. 2024. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *CoRR*, abs/2407.17032.

Walley, P. 1996. Measures of Uncertainty in Expert Systems. *Artif. Intell.*, 83(1): 1–58.

Wang, Q.; Ho, C. P.; and Petrik, M. 2023. Policy Gradient in Robust MDPs with Global Convergence Guarantee. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, 35763–35797. PMLR.

Wang, Y.; Velasquez, A.; Atia, G. K.; Prater-Bennette, A.; and Zou, S. 2023. Robust Average-Reward Markov Decision Processes. In *AAAI*, 15215–15223. AAAI Press.

Wang, Y.; Velasquez, A.; Atia, G. K.; Prater-Bennette, A.; and Zou, S. 2024. Robust Average-Reward Reinforcement Learning. *J. Artif. Intell. Res.*, 80: 719–803.

Wang, Y.; and Zou, S. 2021. Online Robust Reinforcement Learning with Model Uncertainty. In *NeurIPS*, 7193–7206.

Wang, Y.; and Zou, S. 2022. Policy Gradient Method For Robust Reinforcement Learning. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, 23484–23526. PMLR.

Winkler, T.; Junges, S.; Pérez, G. A.; and Katoen, J. 2019. On the Complexity of Reachability in Parametric Markov Decision Processes. In *CONCUR*, volume 140 of *LIPIcs*, 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Wolff, E. M.; Topcu, U.; and Murray, R. M. 2012. Robust control of uncertain Markov Decision Processes with temporal logic specifications. In *CDC*, 3372–3379. IEEE.

Zielonka, W. 1998. Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees. *Theor. Comput. Sci.*, 200(1-2): 135–183.