Test Where Decisions Matter: Importance-driven Testing for Deep Reinforcement Learning

Stefan Pranger,[1] **Hana Chockler,**[2] **Martin Tappler,**[3] **Bettina Könighofer**[1]
[1]Institute of Information Security,
Graz University of Technology
[2]Kings College London
[3]Institute of Computer Engineering,
TU Wien
{stefan.pranger,bettina.koenighofer}@tugraz.at
hana.chockler@kcl.ac.uk
martin.tappler@tuwien.ac.at

Abstract

In many Deep Reinforcement Learning (RL) problems, decisions in a trained policy vary in significance for the expected safety and performance of the policy. Since RL policies are very complex, testing efforts should concentrate on states in which the agent's decisions have the highest impact on the expected outcome. In this paper, we propose a novel model-based method to rigorously compute a ranking of state importance across the entire state space. We then focus our testing efforts on the highest-ranked states. In this paper, we focus on testing for safety. However, the proposed methods can be easily adapted to test for performance. In each iteration, our testing framework computes optimistic and pessimistic safety estimates. These estimates provide lower and upper bounds on the expected outcomes of the policy execution across all modeled states in the state space. Our approach divides the state space into safe and unsafe regions upon convergence, providing clear insights into the policy's weaknesses. Two important properties characterize our approach. (1) Optimal Test-Case Selection: At any time in the testing process, our approach evaluates the policy in the states that are most critical for safety. (2) Guaranteed Safety: Our approach can provide formal verification guarantees over the entire state space by sampling only a fraction of the policy. Any safety properties assured by the pessimistic estimate are formally proven to hold for the policy. We provide a detailed evaluation of our framework on several examples, showing that our method discovers unsafe policy behavior with low testing effort.

1 Introduction

Deep reinforcement learning (RL) [1] is a powerful method for training policies that complete tasks in complex environments. Due to the high potential of RL in safety-critical domains, such as autonomous driving [2], ensuring the reliability of its safety-critical properties is becoming increasingly vital. Formal verification provides provable correctness guarantees [3]. However, the most significant challenge in the formal verification of RL policies is scalability, which limits its current applicability [4]. As for conventional software, a complete safety evaluation by exhaustively testing a policy's decisions is infeasible. Hence, it is necessary to establish as much confidence as possible in a policy with a limited testing budget.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

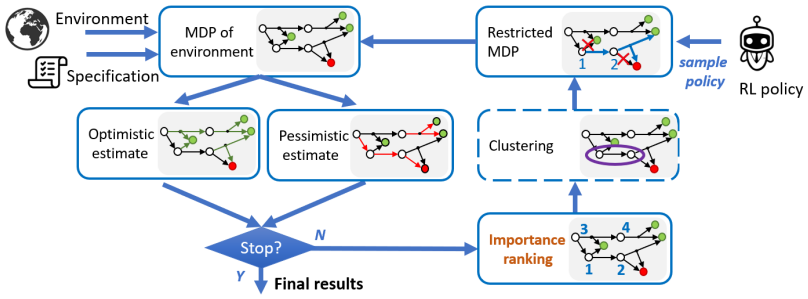


Figure 1: High-level view of the algorithm for importance-driven testing for RL.

We propose a novel model-based testing framework for RL policies, which tests policies in the states where *their decisions matter most*. We follow the insights from Chockler et al. [5] that not all decisions hold equal significance on the expected safety and performance of a policy. Decisions in certain states may have a significant impact on the overall expected outcome of the policy, while in other states, the impact may not be as severe or critical. The core of our algorithm is a *ranking of the importance of states of the environment*. This ranking is based on the difference that the decision in a particular state makes on the expected overall performance (e.g., accumulated reward) or safety of the policy. For lack of space, we focus on safety from here on. The proposed method can easily be adapted to evaluate the agent's performance, which we discuss in Appendix D.

Figure 1 outlines our algorithm. The inputs to our algorithm are a model of the environment in the form of a Markov decision process [6], a formal safety specification φ, and an RL agent in the form of a deterministic policy. In each iteration, our algorithm computes *optimistic and pessimistic estimates*, which provide lower and upper bounds for the expected probability of satisfying the safety specification over all possible policies. The algorithm terminates if the maximal difference between the estimates gets below some threshold or a maximal number of executed test cases is reached. As long as the stopping criterion is not met, the algorithm computes an *importance ranking* of the states. The higher the rank of a state, the more influence the decision in that state has for satisfying or violating the safety specification. Next, the most important decisions of the policy are sampled and used to fix the decisions, thus *restricting* the MDP. The algorithm continues with the restricted MDP to iteratively refine the estimates. Our testing framework can be modified through an optional step by *clustering the highly ranked states*. A fraction of test cases is then uniformly selected from the individual clusters. The intuition behind clustering is that the agent is likely to behave similarly in comparable situations. Following this intuition, we mark all states in a cluster as safe if all tested states of this cluster are verified to be safe. Otherwise, the entire cluster is marked as unsafe. This increases the scalability of our testing approach since only a fraction of each cluster needs to be tested for deriving test verdicts for all states in the cluster. However, since not all decisions in a cluster are sampled, unsafe policy behavior can be missed.

Our algorithm provides the following *benefits*:

- Optimal Test-Case Selection: At any time in the testing process, our approach evaluates the policy in the states that are most critical for safety.
- Guaranteed Safety: A pessimistic estimate provides a formal verification guarantee: under the given model it is guaranteed that if the pessimistic estimate for a given state satisfies the testing criteria, then the agent's policy is formally verified from that state.
- Highlighting the most important decisions is a central technique in explainable AI [7, 8]. We provide a rigorous method to compute an importance ranking. Simpler policies that only use the top-ranked decisions can help understand the policy's decision-making [5].
- The iterative nature of our approach can be used in a debugging process to construct a safety frontier: if a sampled decision in a certain state is evaluated to be unsafe, the next ranking iteration assigns higher importance to the predecessor states to be tested next. Thus, the unsafe region around a safety hazard grows until it reveals all states where the policy behaves unsafely.
- Upon convergence, our approach partitions the state space into safe and unsafe regions. The identified unsafe regions offer interpretable insights into the policy's current weaknesses.

1.1 Related Work

Evaluation of RL policies. Off-policy evaluation (OPE) [9, 10, 11] aims to estimate the expected performance of a newly trained policy by using executions of a previously trained policy. In contrast, our approach estimates the performance of the policy under test by computing the best-possible (optimistic) estimate and the worst-possible (pessimistic) estimate in the current MDP. In contrast to OPE, our framework provides formal verification guarantees over the entire state space: any safety property assured by the pessimistic estimate is formally proven to hold. Several recent works proposed evaluation strategies to analyze RL policies [12], by adapting software testing techniques to RL. Various approaches apply fuzz or search-based testing as a basis to find safety-critical situations in the black-box environment [13, 14, 15, 16], in which to test the policy. Most efforts of the testing community focused on selecting test cases that falsify safety with high probability [15, 17]. These methods effectively reveal unsafe behavior, but they do not provide safety assurance from non-failing tests, as they lack proper notions of coverage. In contrast, our testing approach is model-based. Model-based testing of probabilistic systems was proposed in [18]. To the best of our knowledge, there is no model-based testing approach for RL policies with formalized criteria of completeness. That is, we are the first to propose safety estimates with formal interpretations which form the basis of our test-case generation.

Model-based formal methods for model-free RL. Several recent works have proposed approaches for developing RL controllers by combining model-based formal methods and model-free RL. The appeal of this combination lies in the strengths of each approach: model-based methods offer formal safety and correctness guarantees, while model-free RL demonstrates superior scalability and yields high-performance controllers by learning from the full-order system dynamics [19, 20]. Most of the existing work in this area addresses the problem of safe exploration in RL [21, 22]. To the best of our knowledge, our work is the first to employ similar techniques for analyzing a trained policy.

Importance ranking. Ranking policy decisions has been proposed for explaining and simplifying RL policies. In [5], the ranking is based on statistical fault localization computed on a set of executions of the original policy and its small perturbations. A continuation of this work [23] uses an average treatment effect to rank policy decisions. In contrast, we provide a rigorous method to compute the importance ranking. Our estimates consider any possible behavior of the policy over the entire state space. Thus, the estimates provide strong verification guarantees. Similarly to ranking policy decisions, [24] rank the importance of individual neurons in a network to assess coverage of a given test set.

2 Background

Markov Decision Process. A *Markov decision process* (MDP) [25] $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu)$ is a tuple with a finite state set $\mathcal{S}$, a finite set $\mathcal{A} = \{a_1 \dots, a_n\}$ of actions, a probabilistic transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, and a probability distribution of initial states $\mu : \mathcal{S} \to [0, 1]$. An *execution* (or path) is a finite or infinite sequence $\rho = s_0, a_0, s_1, a_1 \dots$ with $\mathcal{P}(s_i, a_i, s_{i+1}) > 0$ and $\mu(s_0) > 0$. A (memoryless deterministic) *policy* $\pi : \mathcal{S} \to \mathcal{A}$ is a function mapping states to actions. Π denotes the set of all memoryless deterministic policies. Applying π to an MDP $\mathcal{M}$ induces a Markov chain (MC) $\mathcal{M}^\pi$. An execution in $\mathcal{M}^\pi$ is a sequence $\rho = s_0, s_1, s_2, \dots$ with $\mathcal{P}(s_i, \pi(s_i), s_{i+1}) > 0$. $\mathbb{P}^\pi_s$ denotes the unique probability measure of $\mathcal{M}^\pi$ over infinite executions starting in s.

Probabilistic Model Checking. Probabilistic model checking [3] computes the probabilities of satisfying a temporal-logic formula φ over a finite or infinite horizon. We define the properties below with a bound n. For the unbounded horizon, $n = \infty$. For a given MDP $\mathcal{M}$, a policy π, and a property φ in Computation Tree Logic (CTL) [3], model checking computes the following probabilities:

- $\mathbb{P}_{\mathcal{M}^\pi,\varphi} \colon \mathcal{S} \times \mathbb{N} \to [0, 1]$ is the expected probability to satisfy φ state $s \in \mathcal{S}$ within n steps in $\mathcal{M}^\pi$.
- $\mathbb{P}^{\mathsf{max}}_{\mathcal{M},\varphi}(s, n) = \max_{\pi\in\Pi} \mathbb{P}_{\mathcal{M}^\pi,\varphi}(s, n)$ is the *maximal* expected probability *over all policies in* Π from a state s within n steps.
- $\mathbb{P}^{\mathsf{min}}_{\mathcal{M},\varphi}(s, n) = \min_{\pi\in\Pi} \mathbb{P}_{\mathcal{M}^\pi,\varphi}(s, n)$ is the *minimal* expected probability *over all policies in* Π from a state s within n steps.

For the remainder of this paper, let φ be a formula in the safety fragment of CTL. Using φ and a user-defined safety threshold δ_φ, we define safety objectives as follows:

Algorithm 1 Importance–driven model-based testing (IMT)

Input: MDP $\mathcal{M}$, policy π, safety objective $\langle \varphi, \delta_\varphi \rangle$
Parameters: # samples m, safety threshold δ_φ, minimal difference ε_φ
Output: failure states $\mathcal{S}_f \subseteq \mathcal{S}$, safe states $\mathcal{S}_s \subseteq \mathcal{S}$, estimates $e_{opt} : \mathcal{S} \to \mathbb{R}$ and $e_{pes} : \mathcal{S} \to \mathbb{R}$

1: $\mathcal{M}^{(0)} \leftarrow \mathcal{M}$; $\mathcal{S}_u \leftarrow \mathcal{S}$; $\mathcal{S}_f \leftarrow \emptyset$; $\mathcal{S}_s \leftarrow \emptyset$; $i \leftarrow 0$
2: **loop**
3: $e_{opt}, e_{pes} \leftarrow$ computeEstimates($\mathcal{M}^{(i)}$)
4: $\mathcal{S}_s \leftarrow \mathcal{S}_s \cup \{s \in \mathcal{S}_u \mid e_{pes}(s) \geq \delta_\varphi\}$
5: $\mathcal{S}_f \leftarrow \mathcal{S}_f \cup \{s \in \mathcal{S}_u \mid e_{opt}(s) < \delta_\varphi\}$
6: $\mathcal{S}_u \leftarrow \mathcal{S}_u \setminus \{s \in \mathcal{S}_u \mid e_{opt}(s) < \delta_\varphi \vee e_{pes}(s) \geq \delta_\varphi\}$
7: **if** $[\max_s(e_{opt}(s) - e_{pes}(s)) < \varepsilon_\varphi]$ **then**
8: **stop**
9: **end if**
10: $\mathcal{S}_{rank} \leftarrow$[computeRanking($\mathcal{M}^{(i)}, m$)]
11: $\{(s_1, a_1) \dots (s_m, a_m)\} \leftarrow$ samplePolicy($\pi, \mathcal{S}_{rank}$)
12: $\mathcal{M}^{(i+1)} \leftarrow$ restrictMDP($\mathcal{M}^{(i)}, \{(s_1, a_1) \dots (s_m, a_m)\}$)
13: $i \leftarrow i + 1$
14: **end loop**
15: **return** $\mathcal{S}_f, \mathcal{S}_s, e_{opt}, e_{pes}$

Definition 2.1 (Safety objective)**.** Given an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu)$, a safety property φ, and a threshold $\delta_\varphi \in [0, 1]$. A *safety objective* is a tuple $\langle \varphi, \delta_\varphi \rangle$. A policy π satisfies $\langle \varphi, \delta_\varphi \rangle$ from a given state $s \in \mathcal{S}$ within n steps if $\mathbb{P}_{\mathcal{M}^\pi, \varphi}(s, n) \geq \delta_\varphi$.

Reinforcement Learning. An RL [1] agent learns a task via interactions with an unknown environment modeled by an MDP $\mathcal{M}$ with an associated reward function $\mathcal{R} : \mathcal{S} \to \mathbb{R}$. In each state $s \in \mathcal{S}$, the agent chooses an action $a \in \mathcal{A}$, the environment then moves to a state s' with probability $\mathcal{P}(s, a, s')$. The return $\texttt{ret}_\rho$ of an execution ρ is the discounted cumulative reward defined by $\texttt{ret}_\rho = \Sigma_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t)$, using the discount factor $\gamma \in [0, 1]$. The objective of the agent is to learn a deterministic memoryless *optimal policy* $\pi^\star$ that maximizes the expectation of the return.

3 Importance-driven Testing for RL

In this section, we will describe our framework for importance-driven model-based testing, which we abbreviate with IMT. An overview of our algorithm is depicted in Fig. 1. Its central elements are the computation of the estimates and the importance ranking that guides the selection of the test cases. In Sec. 3.1 we discuss IMT in detail, and in Sec.3.2 we discuss its extension with clustering.

3.1 Importance-driven Model-Based Testing

Alg. 1 gives the pseudo-code of our approach for importance-driven safety testing. Our algorithm evaluates a policy π with respect to a safety objective $\langle \varphi, \delta_\varphi \rangle$ over a horizon of n steps (for the unbounded horizon, $n = \infty$). The algorithm takes as input an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu)$, a policy under test $\pi : \mathcal{S} \to \mathcal{A}$, and a safety objective $\langle \varphi, \delta_\varphi \rangle$. It returns as result a classification of states into safe and failure states ($\mathcal{S}_s$ and $\mathcal{S}_f$, respectively), and the optimistic and pessimistic estimates for all states in the state space ($e_{opt} : \mathcal{S} \to [0, 1]$ and $e_{pes} : \mathcal{S} \to [0, 1]$, respectively), which are derived as the expected maximal and minimal probability of satisfying the safety objective $\langle \varphi, \delta_\varphi \rangle$.

Safety estimates. In Line 3, IMT computes the safety estimates for the current (restricted) MDP $\mathcal{M}^{(i)}$. The optimistic estimate $e_{opt}(s, n)$ is the maximal expected probability of satisfying φ for an execution in $\mathcal{M}^{(i)}$ from a given state s within a n steps quantified over all policies. Similarly, the pessimistic estimate $e_{pes}(s, n)$ is the minimal expected probability of satisfying φ.This yields the following definition:

Definition 3.1 (Safety estimates)**.** For a given MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu)$, a given safety property φ, and a given number of n steps, the *optimistic* and *pessimistic safety estimate* $e_{opt}, e_{pes} : \mathcal{S} \times \mathbb{N} \to [0, 1]$

are defined as follows:

$$\forall s \in \mathcal{S}\colon e_{opt}(s,n) = \mathbb{P}^{\mathsf{max}}_{\mathcal{M},\varphi}(s,n), \text{ and } \quad \forall s \in \mathcal{S}\colon e_{pes}(s,n) = \mathbb{P}^{\mathsf{min}}_{\mathcal{M},\varphi}(s,n).$$

For a state action pair (s,a) and a bound n, the maximal expected probability of satisfying φ from a state s after executing a is

$$\forall s \in \mathcal{S}, \forall a \in \mathcal{A} : e_{opt}(s,a,n) = \sum_{s' \in \mathcal{S}} (\mathcal{P}(s,a,s') \cdot e_{opt}(s', n-1)).$$

Based on the estimates, the algorithm classifies undetermined states from $\mathcal{S}_u$ as verified safe and adds them to $\mathcal{S}_s$ or classifies them as unsafe and adds to the set of failure states $\mathcal{S}_f$ (Lines 4 and 5). A state $s \in \mathcal{S}$ satisfies the safety objective $\langle \varphi, \delta_\varphi \rangle$ if $e_{pes}(s,n) \geq \delta_\varphi$. Note that the pessimistic safety estimate is achieved in an execution that chooses the most unsafe actions in each non-restricted state. Thus, if for a given state $e_{pes}(s,n) \geq \delta_\varphi$, then $\mathbb{P}_{\mathcal{M}^\pi,\varphi}(s,n) \geq \delta_\varphi$ holds. *This highlights the strength of our algorithm: by assuming the worst policy behavior in unrestricted states, we provide verification results without sampling the policy in every state.* A state $s \in \mathcal{S}$ is unsafe if $e_{opt}(s,n) \leq \delta_\varphi$. The optimistic safety probability is achieved in an execution that chooses the safest action in each non-restricted state. Thus, if $e_{opt}(s,n) \leq \delta_\varphi$, the policy π cannot pick actions that would yield higher probabilities of satisfying φ from s.

Stopping criteria. In Line 7, the stopping criterion is defined via a user-defined threshold ε_φ for the minimal difference between the estimates. IMT stops if the difference between the optimistic and the pessimistic safety estimate is below the threshold ε_φ for all states, i.e., $\max_s [e_{opt}(s,n) - e_{pes}(s,n)] < \varepsilon_\varphi$. For small values of e_{pes}, further restricting the MDP would only marginally change the testing results. Otherwise, IMT continues with sampling the policy and restricting the MDP, as the optimistic and pessimistic estimates are sufficiently different. As an alternative stopping criterion, a user could also define a total testing budget.

Importance ranking. In each iteration, IMT computes an importance ranking over all states in the current MDP $\mathcal{M}^{(i)}$ (Line 10). In the following steps, the m most important decisions of the policy are sampled and used to restrict $\mathcal{M}^{(i)}$, which results in refined estimates. The rank of a state s reflects the *maximal difference that a decision can have* on satisfying the safety objective.

Definition 3.2. (Importance ranking for safety.) Given an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu)$, a safety property φ, and a bound n, the importance ranking $rank\colon \mathcal{S} \times \mathbb{N} \to \mathbb{R}$ is given as the *maximal difference between the optimistic estimates* with respect to the available actions:

$$\forall s \in \mathcal{S}\colon rank(s,n) = \max_{a,a' \in \mathcal{A}} (e_{opt}(s,a,n) - e_{opt}(s,a',n)).$$

For the importance ranking, we consider the impact of decisions on the *optimistic estimates*. That is, a state s is important if, for some actions a and a', it holds that the expected maximal safety probability that can still be achieved after executing a from s is considerably larger than the probability that can be obtained after executing a' from s. The importance ranking returns the set of states $\mathcal{S}_{rank}$ of the m highest ranked states of $\mathcal{M}^{(i)}$.

Sampling the policy. In Line 11, IMT samples the decisions of the policy in the highest ranked states $s \in \mathcal{S}_{rank}$ of $\mathcal{M}^{(i)}$. This results in the set $\Gamma = \{(s_1, a_1), \ldots, (s_m, a_m)\}$ with $a_i = \pi(s_i)$.

Restricting the MDP. In Line 12, our algorithm restricts $\mathcal{M}^{(i)}$ according to the sampled policy's decisions, i.e., actions not chosen by π in the sampled states are removed from $\mathcal{M}^{(i)}$. Given the current MDP $\mathcal{M}^{(i)} = (\mathcal{S}, \mathcal{A}, \mathcal{P}^{(i)}, \mu)$ and the sampled state-action pairs Γ, the restricted MDP $\mathcal{M}^{(i+1)} = (\mathcal{S}, \mathcal{A}, \mathcal{P}^{(i+1)}, \mu)$ has the following probabilistic transition function:

$$\forall s, s' \in \mathcal{S}\, \forall a \in \mathcal{A} : \mathcal{P}^{(i+1)}(s,a,s') = \begin{cases} \mathcal{P}^{(i)}(s,a,s') & s \notin \mathcal{S}_{rank} \text{ or } (s,a) \in \Gamma \\ 0 & \text{else.} \end{cases}$$

In every iteration of the algorithm, more actions in the MDP model become fixed to the actions chosen by π, leading to more accurate safety estimates for π, i.e., $e_{pes}(s,n)$ monotonically increases and $e_{opt}(s,n)$ monotonically decreases, for all $s \in \mathcal{S}$.

Theorem 1. The algorithm IMT as described in Alg. 1 terminates.

Proof Sketch. For a fully restricted MDP, for any $s \in \mathcal{S}$, for any n, it holds that $e_{opt}(s,n) = e_{pes}(s,n)$. This holds because a fully restricted MDP is a Markov chain that describes the policy completely. Hence the estimates are the same.

Algorithm 2 IMT with Clustering

Input: $\mathcal{M}$, π, $\langle \varphi, \delta_\varphi \rangle$
Parameters: importance threshold δ_i, testing fraction κ, testing horizon n, δ_φ, ε_φ
Output: $\mathcal{S}_f \subseteq \mathcal{S}$, $\mathcal{S}_s \subseteq \mathcal{S}$, $e_{opt} : \mathcal{S} \to \mathbb{R}$, $e_{pes} : \mathcal{S} \to \mathbb{R}$

1: // Line 1 — 8 are as in Alg. 1
9: $\mathcal{S}_{rank} \leftarrow$[computeRanking($\mathcal{M}^{(i)}$)]
10: $\mathcal{C} \leftarrow$[clusterStates($\mathcal{S}_{rank}, \delta_i$)]
11: $\{(c_1, v_1) \dots (c_{|\mathcal{C}|}, v_{|\mathcal{C}|})\} \leftarrow$ executeTests($\pi, \mathcal{C}, \kappa$)
12: $\mathcal{M}^{(i+1)} \leftarrow$ restrictMDP($\mathcal{M}^{(i)}, \{(c_1, v_1) \dots (c_{|\mathcal{C}|}, v_{|\mathcal{C}|})\}$)
13: // Line 12 — 14 are as in Alg. 1

3.2 Importance-driven Model-Based Testing with Clustering

In this section, we extend IMT by introducing clustering in Alg. 1. Fig. 1 shows the high-level view of IMT including clustering. For problems with very large state spaces, sampling the agents in all highly-ranked states becomes too expensive. To tackle this scalability issue, we propose to cluster similar states and test only a fixed fraction of the states in each cluster. By doing so, we balance the trade-off between accuracy and scalability: The fewer states from a cluster are tested, the higher the scalability of our testing approach. However, the likelihood that some unsafe behavior of the agent remains undetected increases. Clustering offers the additional advantage that similar states are grouped. Under the assumption that the agent implements a policy that selects the same action in similar situations, IMT most likely detects unsafe behavior by sampling a large enough fraction of each cluster. Alg. 2 states the changes in the pseudo-code for IMT with clustering.

Clustering. In Line 10, after computing the importance ranking, IMT performs clustering on all states with an importance ranking value greater than some bound $\delta_i = rank(s.n)$. States are clustered according to their state information and their importance value, i.e., we compute a clustering assignment $cl : \mathcal{S} \times [\delta_i, 1] \to \mathbb{N}$. This gives a partitioning of $\mathcal{S}_{rank}$ into sets of states sharing the same cluster label. Note that any off-the-shelf clustering algorithm can be used to compute the clusters of states. [1]

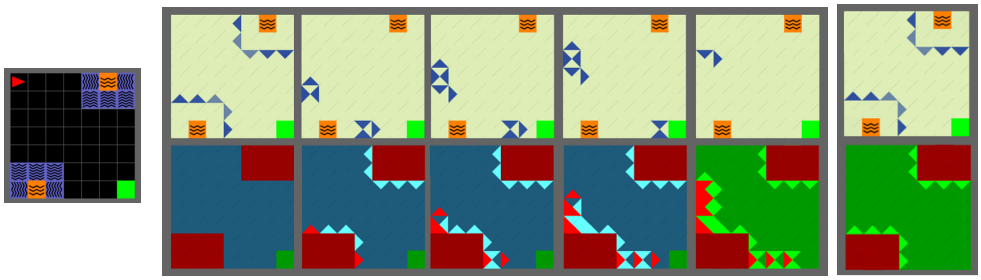
Executing tests. In Line 11, the behavior of the policy in the clustered states is evaluated. From each cluster, a fixed percentage κ of states is randomly selected to be tested. To test a state s, the agent is executed from s for a certain number of steps. If the safety objective is violated during the execution, s is marked as unsafe and added to $\mathcal{S}_f$. Based on the testing results of the individual states we assign verdicts v_j to the clusters proposing a conservative evaluation of safety. Each cluster c_j with a tested state $s \in \mathcal{S}_f$ is assigned a failing verdict $v_j = \texttt{FAIL}$. Consequently, all states $s \in c_j$ are marked unsafe and added to $\mathcal{S}_f$. Conversely, if a cluster c_j does not contain a single tested state from $\mathcal{S}_f$, it is assigned a safe verdict $v_j = \texttt{SAFE}$, and its states are added to $\mathcal{S}_s$.

Restricting the MDP. In Line 12, IMT restricts $\mathcal{M}^{(i)}$ in all states that belong to a cluster c_j by turning the states into sink states. Additionally, if a cluster c_j has the verdict $v_j = \texttt{FAIL}$, all states are considered a safety violation.

The effects of clustering. Since IMT with clustering only tests a fraction κ of each individual cluster, the size and quality of the computed clusters affect the testing process. Clusters that are too large can lead to unnecessary testing efforts, as safe behavior might be deemed unsafe due to conservative evaluation. Additionally, if a cluster contains states that are not sufficiently similar, IMT with clustering may fail to detect unsafe behavior in the policy.

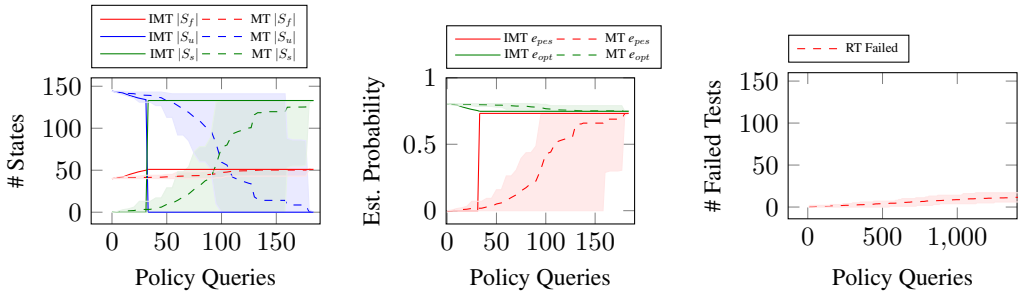
Complexity Analysis. We discuss the computational complexity of a single iteration of IMT. The safety estimates are computed via value iteration in $\mathcal{O}(poly(size(\mathcal{M})) \cdot n)$, with n being the bound for the objective [3]. The computation of the ranking only requires sorting of the computed estimates and thus requires $\mathcal{O}(|\mathcal{S}| \log |\mathcal{S}|)$ time. The subsequent restriction of $\mathcal{M}$ is linear in the number of actions present in $\mathcal{M}$, i.e. $\mathcal{O}(|\mathcal{S}| \cdot |\mathcal{A}|)$. Lastly, the complexity of sampling the policy is dependent on the network architecture and the costs for clustering the state space depend on the chosen algorithm.

[1] For vision-based state spaces, deep clustering approaches could be used [26].



(a) GridWorld. (b) Results for π_1. (c) Results for π_2.

Figure 2: Slippery Gridworld example: setting (left), visualization of evaluating π_1 (middle), and π_2 (right).



(a) Verified States. (b) Average safety estimates. (c) Results for random testing.

Figure 3: Slippery Gridworld example: Evaluation results of π_1.

4 Experimental Evaluation

All details of the experimental setup can be found in Appendix A. We provide the implementation and tested policies as supplementary material. We compare IMT with our model-based approach *without* importance ranking (MT) and model-free random testing (RT) as a baseline. Thus, in MT, our algorithm restricts the MDP by the sampled agent's decisions and computes e_{opt} and e_{pes} to provide evaluation results on the entire state space but *samples the policy randomly*. For RT, the policy is executed from random states for a certain number of steps. Any violation of the evaluation objective is reported. We report the runtimes averaged over 10 runs for each experiment in seconds, unless indicated otherwise, as *total time($\pm$ STDev) / total time for computing the estimates($\pm$ STDev) / total time for querying the policy ($\pm$ STDev).*

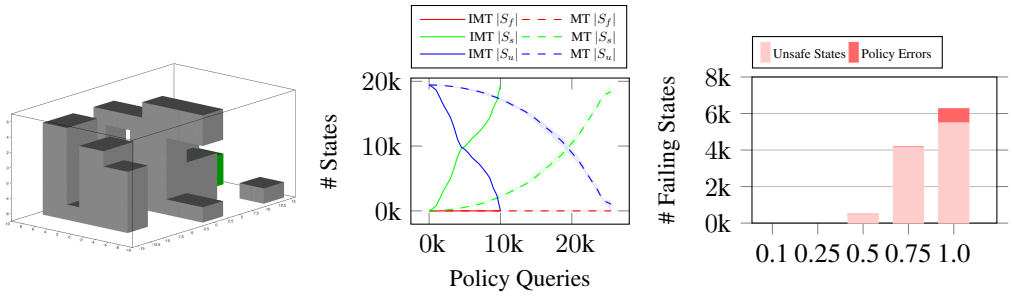
4.1 Slippery Gridworld

We performed our first experiment in the Farama Minigrid environment [27]. A description of the environment and the RL training parameters are given in Appendix B. The Gridworld is depicted in Fig. 2a. The agent has to reach the green goal without touching the lava. The lava is surrounded by slippery tiles, stepping on which carries a predefined probability of slipping into lava. The size of the state space is $|\mathcal{S}| = 7 \times 7 \times 4 = 196$, with 49 cells multiplied by 4 for the different orientations of the agent. The safety objective φ requires the agent to not enter the lava with a probability $\delta_\varphi \geq 1.0$.

RL training parameters. We trained policies π_1 and π_2 by utilizing a DQN. We used a sparse reward function with a reward of 1 for reaching the green goal and -1 for falling into the lava. We trained π_1 using a fixed initial state and π_2 using initial states uniformly sampled from $\mathcal{S}$.

IMT/MT parameters. We used a horizon of $n = \infty$, a minimal difference of $\varepsilon_\varphi = 0.05$, a number of samples per iteration of $m = 10$. For IMT, no states with a ranking value close to 0 were sampled.

Visualizing IMT. Fig. 2b visualizes the iterations of our IMT algorithm when evaluating π_1. Per iteration, the picture on the top visualizes the highest-ranked states, with the intensity of the color capturing the ranking. Note that a state represents the (x, y)-coordinates of the grid and the orientation of the agent and is thus visualized as a triangle. Per iteration, IMT samples π_1 in the highest-ranked



(a) UAV Reach-Avoid setting. (b) Results for $noise = 0.1$ (c) Number of safety violations.

Figure 4: UAV Task: setting (4a), verified states (4b), and number of identified safety violations (4c).

states (blue triangles) and computes the estimates. The pictures on the bottom show the updated sets of verified states after computing the estimates: $\mathcal{S}_s$ is visualized in green, $\mathcal{S}_f$ in red, and $\mathcal{S}_u$ in blue. Brighter colors represent states in which the decisions of π_1 were sampled. IMT terminates after 5 iterations when evaluating π_1. Note that the evaluation iteratively reveals the area in which π_1 violates safety. Fig. 2c visualizes the evaluation of the policy π_2. IMT terminates after a single iteration and positively verifies π_2 in all states in which the safety objective can be fulfilled.

Evaluation results. Fig. 3a plots the number of verified states when evaluating π_1. Solid lines represent IMT results, dashed lines represent MT, where green lines represent $|S_s|$, red lines represent $|S_f|$, and blue lines $|S_u|$. We repeated the analysis via MT 10 times: the shaded area represents the minimal and maximal values, and the dashed lines the average number of states. After sampling π_1 only 33 times, IMT terminates with $|\mathcal{S}_s| = 145$, $|\mathcal{S}_f| = 51$, and $|\mathcal{S}_u| = 0$. Thus, IMT provides complete verification results of π_1 over the entire state space with only 33 policy samples. In contrast, on average, MT verifies the entire state space after sampling the agent's decisions almost on the entire state space. Fig. 3b plots the values for the optimistic (green) and pessimistic (red) safety estimates for IMT (solid lines) and MT (dashed lines), averaged over all states, which show that the averaged estimates of IMT tighten faster than for MT. Finally, we report the findings of RT in Fig. 3c when executing a test case for 10 steps. The results show the clear advantage of exploiting our testing approach. By utilizing testing with model checking, we obtain verification results on the entire state space in contrast to RT which is only able to report a small number of states from which φ is violated.

Runtimes. The costs for computing the estimates per iteration are in the range of milliseconds. The total runtime to verify π_1 was $12.29(\pm 0.7)$ / $1.11(\pm 0.10)$ / $0.11(\pm 0.01)$, with IMT and $25.62(\pm 1.8)$ / $3.21(\pm 0.23)$ / $0.41(\pm 0.02)$ with MT.

4.2 UAV Reach-Avoid Task

For the second set of experiments, we test policies computed for drone navigation by Badings et al. [28]. We refer to this work for details regarding the policy and environment, which is illustrated in. Fig. 4a. The task of the drone is to navigate to the goal location (green box). The safety objective φ states that the drone must not collide with a building (grey boxes) and must stay within the boundaries with a probability $\delta_\varphi \geq 0.95$. The state space $|\mathcal{S}|$ comprises 25.517 states. The wind in the simulation affects the drone, which is modeled stochastically and controlled through the parameter η.

IMT parameters. We used $m = 500$ samples per iteration, n, and ε_φ, as above.

Evaluation results. Our testing approach IMT/MT was able to verify control policies computed under five difference noise settings of $\eta \in \{0.1, 0.25, 0.5, 0.75, 1.0\}$ over the entire state space. Fig. 4c gives the number of verified unsafe states per policy. All policies with $\eta < 0.75$ are verified safe in all states from which it is possible to behave safely (light red bars indicate states from which safety violations cannot be avoided). Even though the policies have been specially designed to be safe, IMT was able to find safety violations for policies with $\eta \geq 0.75$. The policies showed unsafe behavior in 15 or 775 additional states, respectively, for which safe behavior would have been possible (dark bars). Fig. 4b shows the verification results for IMT and MT for $\eta = 0.1$. The test results for the policies with $\eta \geq 0.25$ can be found in Appendix C. As before, adding importance-ranking for sampling the policy decreases the number of required samples to verify the policy. We performed RT

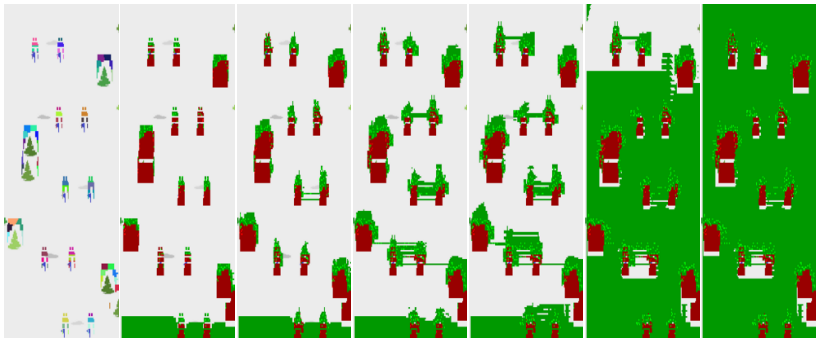


Figure 5: The initial clustering and iterations of the algorithm for an average cluster size of 25.

with a budget of 50.000 queries and a maximum number of 3 time steps per test case. Averaged over 10 runs, RT found 1120 ($\pm$76.29) failed test cases for the policy with $\eta = 1.0$ and 613 ($\pm$53) failed test cases for $\eta = 0.75$.

Runtimes. The runtimes for evaluating the policy are 269.8($\pm$5.5)/ 73.74($\pm$1.1)/ 0.02($\pm$0.02) for IMT, and 1793($\pm$21.1) / 185.28($\pm$3.2) / 0.02($\pm$0.02) for MT for a noise level of $\eta = 0.2515$ This shows that for larger examples, adding importance-based sampling significantly reduces the time needed to verify the policy.

4.3 Atari Skiing

We have evaluated IMT with clustering, IMTc for short, by testing a learned policy for Atari Skiing [29]. In Skiing, the player controls the tilt of the skies to reach the goal as fast as possible. The safety objective φ is to avoid collisions with trees and poles with a probability of $\delta_\varphi \geq 1.0$. A state describes the (x, y) position, the $tilt \in [1..8]$ of the ski, and velocity $v \in [0..5]$ of the skier. The state space $\mathcal{S}$ comprises roughly $2.2 * 10^6$ states.

(a) Verified States. (b) Number of safe and failed tests. (c) Number of states in the final clusters.

Figure 6: Atari Skiing Example: Evaluation results for the tested policy.

IMT parameters. We used a time horizon of $n = 200$, a minimal difference of $\epsilon_\varphi = 0.05$, and a fraction $\kappa = 0.2$ of tested states per cluster. The clusters have been computed using k-means for states with $\delta_i > 0.8$ with a k to create average cluster sizes of $\zeta \in \{25, 50, 100, 150\}$.

Visualizing IMT. Fig. 5 visualizes the initial clustering of the highest-ranked states and the iterations of IMTc with an average cluster size of $\zeta = 25$. We show the results for states in which $tilt = 4$, i.e. the skier is *aligned with the slope*, and $v = 4$. The visualization for different values of $tilt$ and v can be found in Appendix E. We depict states from which the policy has been tested with lighter colors. Darker colors depict implied results.The results show that the agent robustly learned to avoid collisions (it avoids any collision as long as it is not placed too close to an obstacle).

Evaluation results. We evaluated IMTc using different values for ζ and compared it with IMT, i.e. $\zeta = 1$, and RT. Fig. 6a plots the total number of failure states $\mathcal{S}_f$ and safe states $\mathcal{S}_s$ for the whole state space over the number of executed test cases for different values of ζ. The green curves (left to right) plot the results for $\mathcal{S}_s$ using the cluster sizes $\{25, 50, 100, 150, 1\}$, the red curves for $\mathcal{S}_f$ accordingly. For comparison, we executed RT 10 times and plot the average number of failing (orange dashed) and safe test cases (teal dashed), where the shaded areas show the minimal and

maximal values. The results show that IMTc terminates faster with smaller cluster sizes. Larger cluster sizes overapproximate unsafe regions more heavily. Thus, more testing effort is needed around the unsafe regions in the subsequent iterations. However, all instances of IMTc reduce the testing budget required compared to IMT, which was to be expected since only 20% of the states of each cluster were tested. These facts are also underlined by Figures 6b and 6c, which show the number of safe and failed tests, and the implied verdicts for cluster states in the final iteration, respectively. Fig. 6b shows that clustering heavily increases the scalability of our approach since it lowers the needed testing budget by up to a factor of 5 for $\zeta = 25$.

Runtimes. The runtimes for evaluating the policy, excluding the time needed to render the testing results, for $\zeta \in \{25, 50, 100, 150\}$ are 86 minutes (± 8.3) / 40 (± 4.5) / 8.5 (± 2.3). Evaluating the policy for $\zeta = 1$ took 127 minutes / 59 (± 7.3) / 35.9 (± 4.4).

5 Conclusion & Future Work

We presented importance-driven testing for RL agents. The process iteratively (1) ranks the states based on the influence of the agent's decisions on the expected overall safety, (2) samples the DRL policy under test from the ranking, and (3) restricts the model of the environment. By utilizing probabilistic model checking, our algorithm provides upper and lower bounds on the expected outcomes of the policy execution across all modeled states in the state space. These estimates provide formal guarantees about the violation or compliance of the policy to formal properties. We presented an extension of the basic algorithm by introducing clustering to increase scalability. In future work, we will adapt IMT to allow the testing of stochastic policies by adapting the restriction of the MDP and the verification procedure. We will Furthermore, we will introduce several abstraction techniques to further increase the scalability of our approach. Finally, we will use recently proposed approaches to both learn discrete models of domains that are continuous in both their state and action spaces to increase the applicability of IMT and learn the MDP online during the training phase of the policy.

Bettina Könighofer and Stefan Pranger were supported by the State Government of Styria, Austria - Department Zukunftsfonds Steiermark, Martin Tappler was supported by the WWTF project ICT22-023, and Hana Chockler was supported in part by the UKRI Trustworthy Autonomous Systems Hub (EP/V00784X/1), the UKRI Strategic Priorities Fund to the UKRI Research Node on Trustworthy Autonomous Systems Governance and Regulation (EP/V026607/1), and CHAI - EPSRC Hub for Causality in Healthcare AI with Real Data (EP/Y028856/1). We thank both Antonia Hafner and Martin Plank for their proof-of-concept implementations of the experimental evaluation.

References

[1] H.-n. Wang, N. Liu, Y.-y. Zhang, D.-w. Feng, F. Huang, D.-s. Li, and Y.-m. Zhang, "Deep reinforcement learning: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 21, no. 12, pp. 1726–1744, 2020.

[2] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. K. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, 2022.

[3] C. Baier and J. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[4] M. Landers and A. Doryab, "Deep reinforcement learning verification: A survey," *ACM Comput. Surv.*, vol. 55, jul 2023.

[5] H. Pouget, H. Chockler, Y. Sun, and D. Kroening, "Ranking policy decisions," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pp. 8702–8713, 2021.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning - an introduction*. Adaptive computation and machine learning, MIT Press, 1998.

[7] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," *Knowledge-Based Systems*, vol. 214, p. 106685, 2021.

[8] S. Milani, N. Topin, M. Veloso, and F. Fang, "Explainable reinforcement learning: A survey and comparative review," *ACM Comput. Surv.*, 2024.

[9] M. Uehara, C. Shi, and N. Kallus, "A review of off-policy evaluation in reinforcement learning," *arXiv preprint arXiv:2212.06355*, 2022.

[10] Y. Chandak, S. Niekum, B. da Silva, E. Learned-Miller, E. Brunskill, and P. S. Thomas, "Universal off-policy evaluation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27475–27490, 2021.

[11] N. Jiang and L. Li, "Doubly robust off-policy value evaluation for reinforcement learning," in *International conference on machine learning*, pp. 652–661, PMLR, 2016.

[12] J. Uesato, A. Kumar, C. Szepesvári, T. Erez, A. Ruderman, K. Anderson, K. D. Dvijotham, N. Heess, and P. Kohli, "Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures," in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[13] M. Biagiola and P. Tonella, "Testing of deep reinforcement learning agents with surrogate models," *CoRR*, vol. abs/2305.12751, 2023.

[14] Q. Pang, Y. Yuan, and S. Wang, "Mdpfuzz: testing models solving markov decision processes," in *ISSTA '22: 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, ACM, 2022.

[15] A. Zolfagharian, M. Abdellatif, L. C. Briand, M. Bagherzadeh, and R. S, "A search-based testing approach for deep reinforcement learning agents," *IEEE Trans. Software Eng.*, vol. 49, no. 7, pp. 3715–3735, 2023.

[16] M. Tappler, F. C. Córdoba, B. K. Aichernig, and B. Könighofer, "Search-based testing of reinforcement learning," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022*, 2022.

[17] Z. Li, X. Wu, D. Zhu, M. Cheng, S. Chen, F. Zhang, X. Xie, L. Ma, and J. Zhao, "Generative model-based testing on decision-making policies," in *38th IEEE/ACM International Conference on Automated Software Engineering*, 2023.

[18] M. Gerhold and M. Stoelinga, "Model-based testing of probabilistic systems," in *Fundamental Approaches to Software Engineering - 19th International Conference, FASE 2016*, 2016.

[19] F. Den Hengst, V. François-Lavet, M. Hoogendoorn, and F. van Harmelen, "Planning for potential: efficient safe reinforcement learning," *Machine Learning*, vol. 111, no. 6, pp. 2255–2274, 2022.

[20] J. Song, X. Xie, and L. Ma, "SIEGE: A semantics-guided safety enhancement framework for ai-enabled cyber-physical systems," *IEEE Transactions on Software Engineering*, vol. 49, no. 8, pp. 4058–4080, 2023.

[21] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proceedings of the 32th AAAI conference on artificial intelligence*, 2018.

[22] W.-C. Yang, G. Marra, G. Rens, and L. De Raedt, "Safe reinforcement learning via probabilistic logic shields," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23* (E. Elkind, ed.), pp. 5739–5749, International Joint Conferences on Artificial Intelligence Organization, 8 2023. Main Track.

[23] D. C. McNamee and H. Chockler, "Causal policy ranking," *CoRR*, vol. abs/2111.08415, 2021.

[24] S. Gerasimou, H. F. Eniser, A. Sen, and A. Cakan, "Importance-driven deep learning system testing," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pp. 702–713, 2020.

[25] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[26] Y. Ren, J. Pu, Z. Yang, J. Xu, G. Li, X. Pu, P. S. Yu, and L. He, "Deep clustering: A comprehensive survey," *CoRR*, vol. abs/2210.04142, 2022.

[27] M. Chevalier-Boisvert, L. Willems, and S. Pal, "Minimalistic gridworld environment for gymnasium," 2018. https://github.com/Farama-Foundation/Minigrid.

[28] T. S. Badings, L. Romao, A. Abate, D. Parker, H. A. Poonawala, M. Stoelinga, and N. Jansen, "Robust control for dynamical systems with non-gaussian noise via formal abstractions," *J. Artif. Intell. Res.*, vol. 76, pp. 341–391, 2023.

[29] E. Beeching, "Trained policy for atari skiing." https://huggingface.co/edbeeching/atari_2B_atari_skiing_1111 This work is licensed under the Apache 2 license.

[30] S. Pranger, B. Könighofer, L. Posch, and R. Bloem, "TEMPEST - synthesis tool for reactive systems and shields in probabilistic environments," in *Automated Technology for Verification and Analysis, ATVA 2021*, 2021.

[31] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

A General Experimental Setup

All experiments have been executed on a desktop computer with a 8×3.9GHz Intel i5-8265U CPU and 16GB of RAM using a *single worker*, i.e. we did not use any form of multithreading.

We implemented our importance-driven testing framework in `Python` and used the probabilistic model checking tool `Tempest` [30] to compute the estimates and the importance ranking.

B Details for the Gridworlds Experiments.

In this section, we give the details of the experiments in Section 4.1 and Appendix D.2.

Environmental Details. The agent behaves in the style of an omnidirectional robot. It is able to perform seven actions: Moving forward, turning left, turning right, picking up objects, dropping the object being carried, interacting with doors or other objects, and idlying. The slippery tiles in 2a introduce stochastic behaviour. If the agent tries to move forward on a slippery tile, it only manages to move to its intended tile in front of it with a probability of $\frac{3}{9}$. Otherwise, it slips

- with a probability of $\frac{1}{9}$ to either the adjacent tile to its left or its right, respectively, or
- with a probability of $\frac{2}{9}$ to either the tile to the left or to the right of the tile in front of the agent, respectively.

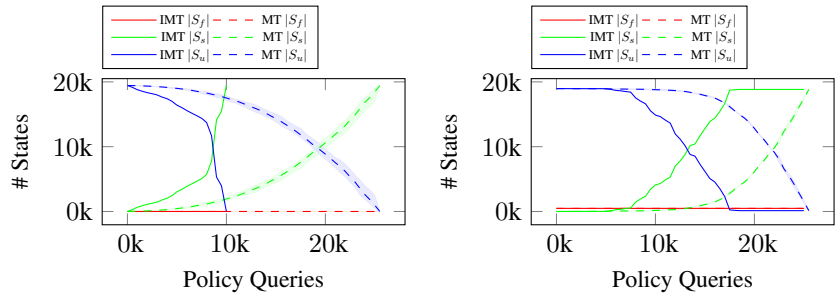
The tiles belonging to one-way streets in 8a, depicted by a blue arrow, do not allow the agent to move against the direction of the one-way. This especially means that an agent is not allowed to enter a one-way from the wrong side.

RL Training Details. We used a standard implementation of DQN from *stable-baselines3* [31] with a CnnPolicy. The network to classify and train the agent follows a standard approach taken from [32]: It features 3 convolutional layers and a linear activation layer. The learning parameters have been slightly altered with the following modifications:

- *discount factor* γ: 0.95
- *exploration scheme:* A linear decay from 0.7 to 0.01 over the first 90% of the learning duration.

The agents for policies π_1, π_2, π_3, and π_4 have been trained with a total number of 500000 steps. An episode lasted a maximum number of 100 timesteps or ended prematurely if the agent caused a safety violation or if it reached the goal.

C Additional Results for UAV Reach-Avoid Experiment



(a) Results for $noise = 0.25$

(b) Results for $noise = 0.5$

(c) Results for $noise = 0.75$

(d) Results for $noise = 1.0$

Figure 7: Evaluation results for the UAV Reach-Avoid task under noise levels $0.25, 0.5, 0.75$ and 1.0.

η	0.1	0.25	0.5
IMT	269sec. ($\pm$5.5)	320sec. ($\pm$16.1)	1264sec. ($\pm$3.8)
MT	1793sec. ($\pm$21.7)	2227sec. ($\pm$0.2)	2570sec. ($\pm$125.4)

η	0.75	1.0
IMT	1383sec. ($\pm$4.4)	2422sec. ($\pm$3.87)
MT	2844sec. ($\pm$34.3)	4016sec. ($\pm$6.23)

Table 1: Average synthesis times for the different policies.

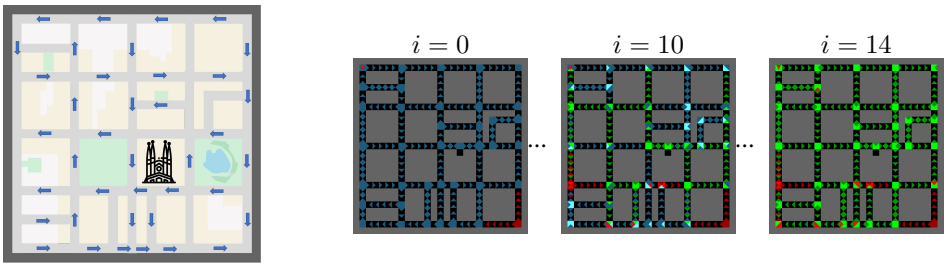
D Testing for Performance

In this section we discuss the necessary background and definitions needed to adapt IMT for testing for performance.

Model checking of performance objectives. Model checking can be used to compute the expected accumulated reward for all states and actions in $\mathcal{M}$. In particular, for a given MDP $\mathcal{M}$, a policy π, and a reward function $\mathcal{R} : \mathcal{S} \to \mathbb{R}$, it computes the following values:

- $\mathbb{E}_{\mathcal{M}^\pi,\mathcal{R}} : \mathcal{S} \times \mathbb{N} \to \mathbb{R}$ gives the expected accumulated reward in $\mathcal{M}^\pi$ from a state s within n steps.
- $\mathbb{E}^{\max}_{\mathcal{M},\mathcal{R}}(s,n) = \max_{\pi\in\Pi} \mathbb{E}_{\mathcal{M}^\pi,\mathcal{R}}(s,n)$ gives the *maximal* expected accumulated reward *over all policies in* Π from a state s within n steps.
- $\mathbb{E}^{\min}_{\mathcal{M},\mathcal{R}}(s,n) = \min_{\pi\in\Pi} \mathbb{E}_{\mathcal{M}^\pi,\mathcal{R}}(s,n)$ gives the *minimal* expected accumulated reward *over all policies in* Π from a state s within n steps.

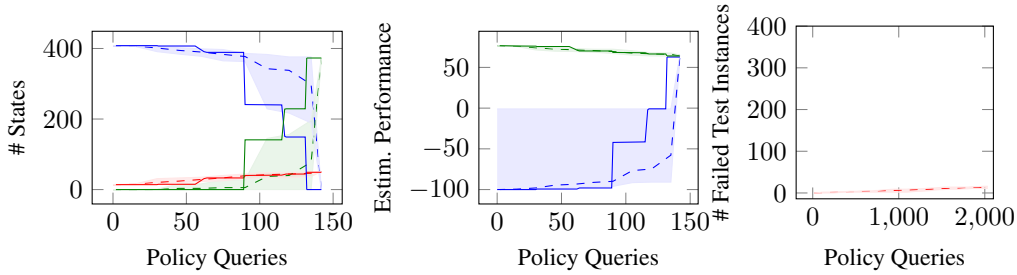
A *performance objective* $\langle \mathcal{R}, \delta_{\mathcal{R}} \rangle$ is defined over the reward function $\mathcal{R}$ and a threshold $\delta_{\mathcal{R}} \in \mathbb{R}$ that defines the lowest-acceptable expected accumulated reward over n steps.



(a) Map of an area in Barcelona.

(b) Selected results for iterations $i = 0$, $i = 10$, $i = 14$ of testing π_3.

Figure 8: Urban navigation example: setting (left) and visualization of evaluating π_3 (right).



(a) Verified States.

(b) Average safety estimates.

(c) Results for random testing.

Figure 9: Urban navigation example: Evaluation results of π_3.

Definition D.1 (Performance objective)**.** Given an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu)$, a reward function $\mathcal{R} : \mathcal{S} \to \mathbb{R}$, and a threshold $\delta_{\mathcal{R}} \in \mathbb{R}$. A policy π satisfies the performance objective $\langle \mathcal{R}, \delta_{\mathcal{R}} \rangle$ from a given state $s \in \mathcal{S}$ within a given number of steps n if

$$\mathbb{E}_{\mathcal{M}^{\pi},\mathcal{R}}(s, n) \geq \delta_{\mathcal{R}}.$$

D.1 Importance-driven Performance Testing

IMT can be easily adapted to evaluate a policy π for performance objectives. To tailor Alg. 1 for performance testing, we provide as inputs a performance objective $\langle \mathcal{R}, \delta_{\mathcal{R}} \rangle$, and a minimal difference in performance $\varepsilon_{\mathcal{R}}$ between optimistic and pessimistic estimates. These inputs replace the corresponding safety-related parameters φ, δ_{φ}, and ε_{φ}. The performance estimates (Line 3) are defined by:

Definition D.2 (Performance estimates)**.** For a given MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu)$, a reward function $\mathcal{R} : \mathcal{S} \to \mathbb{R}$, and a given number of n steps, the *optimistic* and *pessimistic performance estimate* $e_{opt,\mathcal{R}}, e_{pes,\mathcal{R}} : \mathcal{S} \times \mathbb{N} \to \mathbb{R}$ are defined as follows:

$$\forall s \in \mathcal{S} : e_{opt,\mathcal{R}}(s, n) = \mathbb{E}^{\max}_{\mathcal{M},\mathcal{R}}(s, n), \text{ and } \forall s \in \mathcal{S} : e_{pes,\mathcal{R}}(s, n) = \mathbb{E}^{\min}_{\mathcal{M},\mathcal{R}}(s, n).$$

For the optimistic performance estimate, the computation assumes that the policy π selects the action that maximizes the expected reward in each unrestricted state. Conversely, for the pessimistic estimate, the assumption is that the least optimal actions concerning the reward are chosen. A state $s \in \mathcal{S}$ satisfies the performance objective $\langle \mathcal{R}, \delta_{\mathcal{R}} \rangle$ if $e_{pes,\mathcal{R}}(s, n) \geq \delta_{\mathcal{R}}$ (Line 4). A state $s \in \mathcal{S}$ violates the performance objective if $e_{opt,\mathcal{R}}(s, n) \leq \delta_{\mathcal{R}}$ (Line 5).

For the stopping criterion, the difference between performance estimates is compared to $\varepsilon_{\mathcal{R}}$ (Line 7).

D.2 Urban Navigation Task

We modeled a part of Barcelona in a Minigrid environment as illustrated in Fig. 8a. The size of the state space is $|\mathcal{S}| = 426$. The agent's task is to navigate to Sagrada Família (within 100 steps), while respecting the traffic rules, i.e., not driving against the one-ways. We trained and evaluated two policies π_3 and π_4.

RL training parameters. π_3 and π_4 were trained utilizing a DQN. The agent is given the reward of 1 for reaching the goal and additionally -0.01 per step. Both policies have been trained using initial states uniformly sampled from $|\mathcal{S}|$.

IMT/MT parameters. We used the same parameters as for the experiment in Section 4.1, but used $m = 15$.

Visualizing IMT. Fig. 8b visualizes the sets of verified states for π_3 in selected iterations of Alg. 1. We visualize $\mathcal{S}_s$, $\mathcal{S}_f$, and $\mathcal{S}_u$ in the same way as in our first experiment. Brighter colors again represent states where π_3 was sampled. IMT iteratively samples the agent's decisions at crossings ranked on the difference the decisions of the individual roads have on the total length of the path to Sagrada Família. Figures 10 and 11 visualize $\mathcal{S}_s$, $\mathcal{S}_f$, and $\mathcal{S}_u$ for π_4. IMT needs 9 iterations to fully verify that π_4 behaves optimally in any of the modelled states.

Evaluation results. Fig. 9a plots the number of verified states and Fig. 9b plots the estimates when evaluating π_3. As above, we compare IMT and the average results for MT over 10 runs. Even though π_3 performed well on large parts of the state space, IMT and MT identified wrong decisions at several crossings that do not allow the agent to reach the goal in time. For RT we executed π_3 with a budget of 2000 policy queries and a maximum number of 100 time steps. Fig. 9c plots the average number of identified violations. While IMT and MT only have to sample the agent's decisions at the crossings to verify the entire state space, RT on average only found 13.25 (± 2.75) states from which a test case failed. Fig. 12a plots the number of verified states and 12b plots the estimates when evaluating π_4.

Runtimes. The runtime to verify π_3 was 36.79 sec (± 2.0) with MT and 24.70 sec (± 2.2) with IMT. Both approaches MT and IMT needed a similar total runtime of about 9.84sec. (± 0.2) to verify π_4.

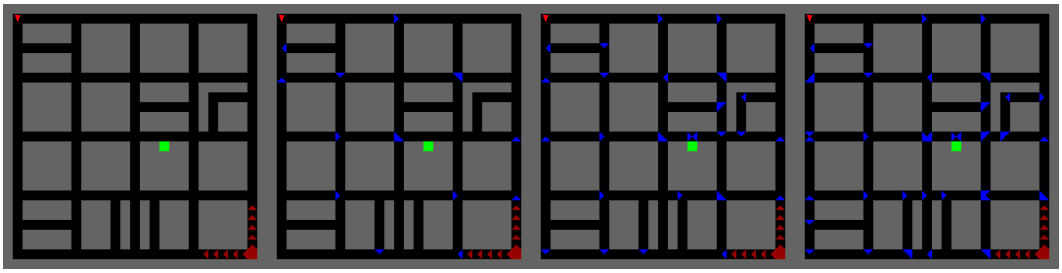


Figure 10: The intermediate results for iteration $0 - 4$ for the verification of π_4 using IMT.

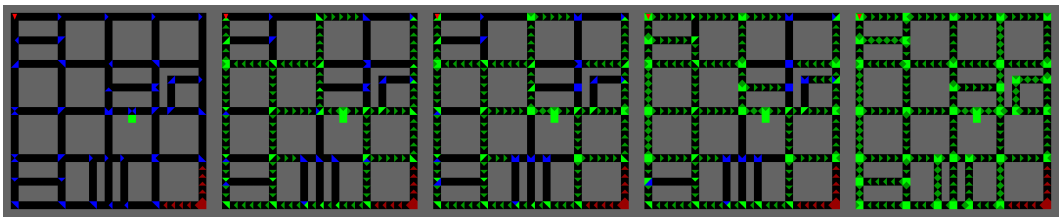
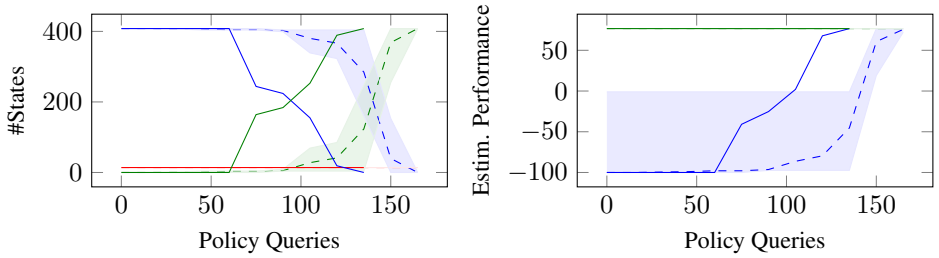


Figure 11: The intermediate results for iteration $5 - 9$ for the verification of π_4 using IMT.



(a) Verified States.

(b) Average safety estimates.

Figure 12: Urban navigation example: Evaluation results of π_4.

E Additional Results for Atari Skiing Experiment

E.1 Results for $\zeta = 25$

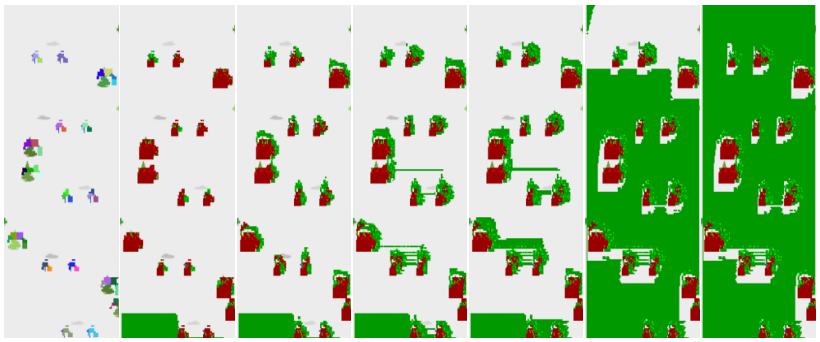


Figure 13: The initial clustering and iterations of IMT for an average cluster size of 25, $tilt = 2$, and $v = 2$.

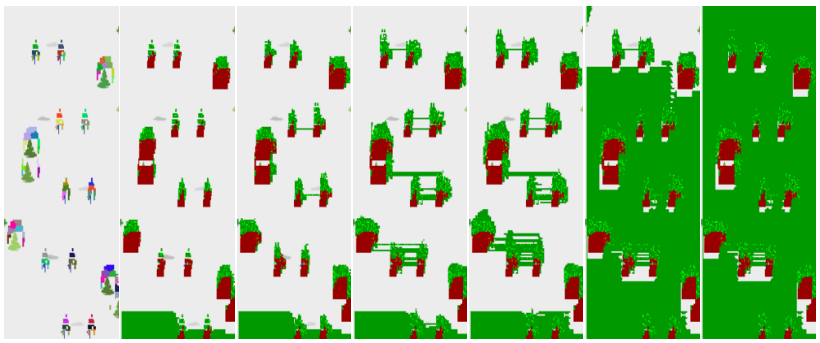


Figure 14: The initial clustering and iterations of IMT for an average cluster size of 25, $tilt = 4$, and $v = 4$.

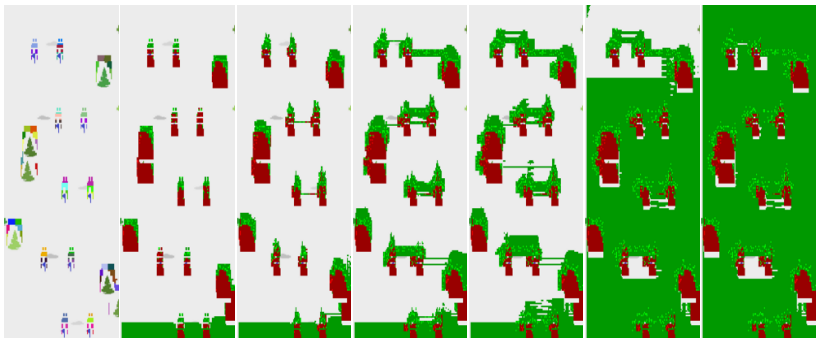


Figure 15: The initial clustering and iterations of IMT for an average cluster size of 25, $tilt = 5$, and $v = 4$.

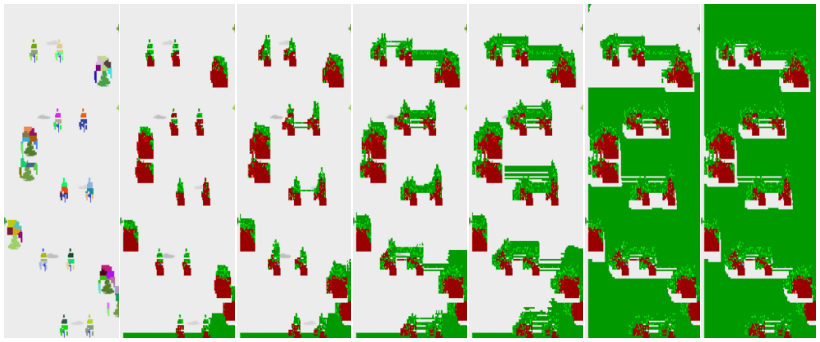


Figure 16: The initial clustering and iterations of IMT for an average cluster size of 25, $tilt = 6$, and $v = 3$.

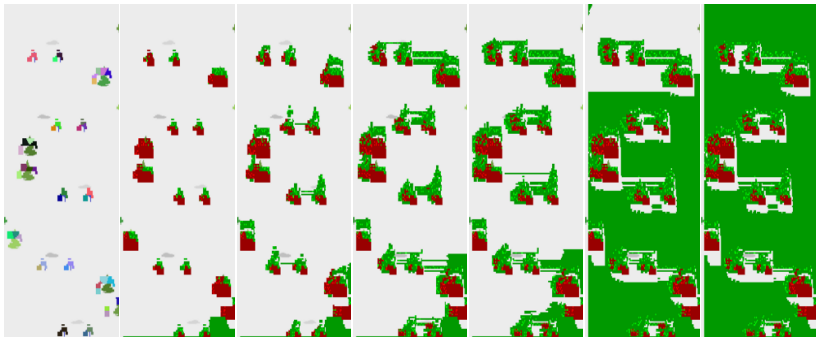


Figure 17: The initial clustering and iterations of IMT for an average cluster size of 25, $tilt = 7$, and $v = 2$.

E.2 Results for $\zeta = 100$

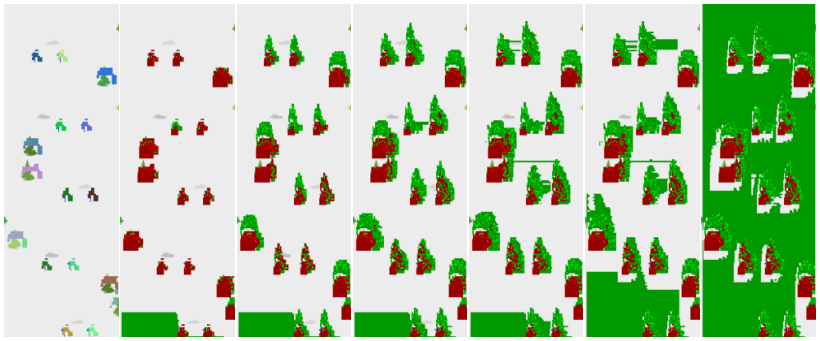


Figure 18: The initial clustering and iterations of IMT for an average cluster size of 100, $tilt = 2$, and $v = 2$.

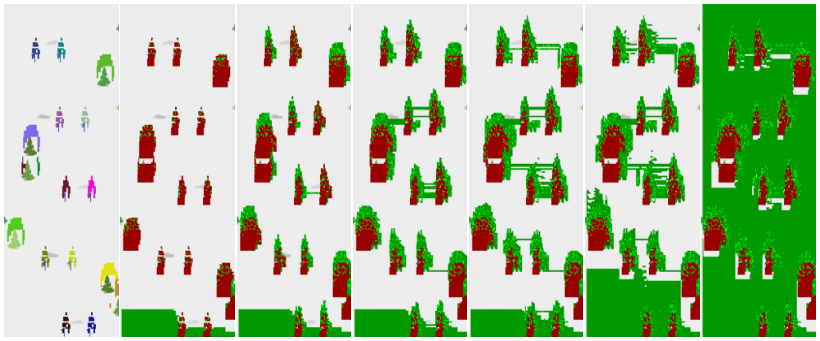


Figure 19: The initial clustering and iterations of IMT for an average cluster size of 100, $tilt = 4$, and $v = 4$.

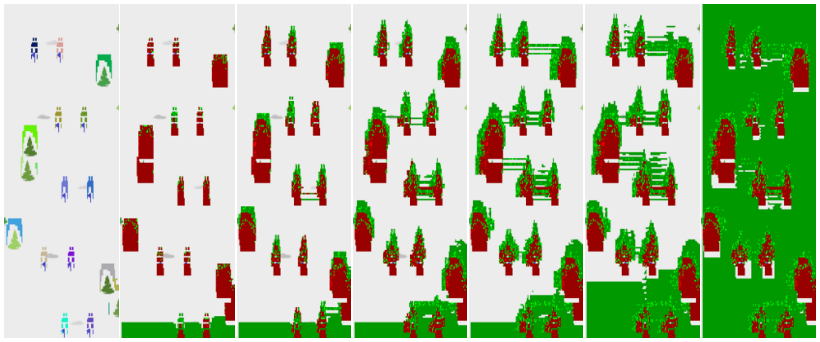


Figure 20: The initial clustering and iterations of IMT for an average cluster size of 100, $tilt = 5$, and $v = 4$.

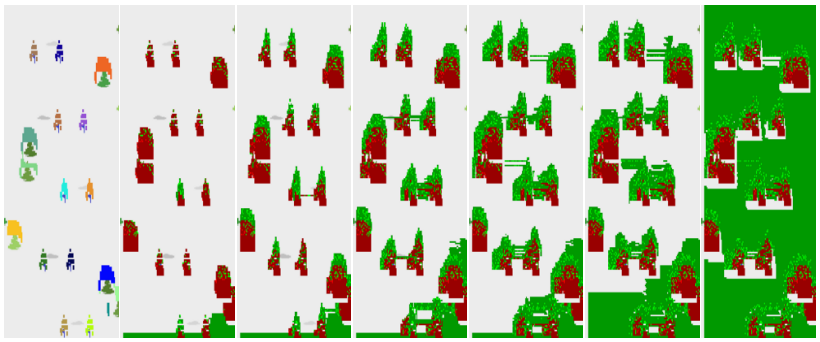


Figure 21: The initial clustering and iterations of IMT for an average cluster size of 100, $tilt = 6$, and $v = 3$.

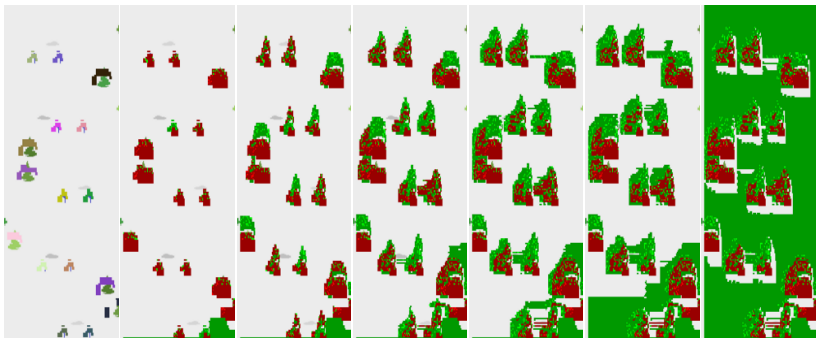


Figure 22: The initial clustering and iterations of IMT for an average cluster size of 100, $tilt = 7$, and $v = 2$.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We present the first model-based testing approach that gives formal verification guarantees. We discuss the method in detail and provide a detailed evaluation in the form of several experiments.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The method currently assumes that the policies under test are determinisitic, as stated in the introduction. We will extend IMT to test stochastic policies in future work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: -

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide a docker image containing all the necessary data and code to reproduce the experiments. The README containing instructions on how to reproduce the results is available via https://figshare.com/s/b8dfb68930da2593749c and the docker image can be downloaded from: https://figshare.com/s/011d813f0b4ad260db5e.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: As stated above, we provide a docker image as artifact that contains all data and code to reproduce the experiments.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so No is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Each experiment is accompanied by a paragraph stating the parameters chosen for our method IMT.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our method is designed to verify a deterministic policy against a safety objective with a fixed threshold. The verification process is therefore not subject to random sampling and we cannot report any statistical significance. For the comparison with random testing, we plot the variability over multiple runs. For evaluating the effect of clustering to increase scalability, we have observed a clear, and expected, trend in the results for different values of ζ and have therefore not statistically verified this.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Each experimental is accompanied by a paragraph stating the runtimes for each individual experiment. In A we state that each experiment has been conducted on a consumer laptop, using a single-threaded implementation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We are not in violation with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: -

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: -

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have used two different existing assets, namely a policy from huggingface, which is attributed in our references and licensed under Apache 2, and the policies for the UAV-Reach-Avoid-Task. The latter have been computed using the source code which is available on Github: https://github.com/LAVA-LAB/DynAbs.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: -

    Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: -

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: -

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.