COMPACT SHIELD REPRESENTATIONS VIA SAFETY-PRESERVING DECISION-TREE APPROXIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Shielding is a well-established technique for enforcing formal safety guarantees in reinforcement learning. However, its practical deployment remains limited in part by the lack of interpretability of synthesized shields. Although recent work has explored decision trees (DTs) as explainable representations of shields, exact DT representations can become prohibitively large. We address this limitation by introducing an algorithm that learns compact DT approximations of safety shields by explicitly trading permissiveness for representation complexity. To preserve the safety guarantees of the original shield, the approximation never classifies an action deemed unsafe by the original shield as safe. To mitigate the resulting performance degradation, we introduce an approximation cost that accounts for the agent's state visitation frequencies and action preferences, together with the risk induced by each action. A user-defined threshold controls the trade-off between fidelity to the original shield and DT complexity. We evaluate our algorithm across diverse domains and show that allowing a small amount of conservative approximation can substantially reduce DT size while preserving the safety guarantees of the original shield and causing at most small changes in agent performance.

1 INTRODUCTION

Reinforcement learning (RL) (Sutton & Barto, 1998) has demonstrated strong performance in complex sequential decision-making problems, but typically provides no formal guarantees that unsafe actions are avoided during training or deployment (Gu et al., 2024). Shielding addresses this problem by placing a formally synthesized safety mechanism between the RL agent and the environment (Alshiekh et al., 2018). Given a model of the safety-relevant dynamics and a safety specification, a shield restricts the actions available to the agent according to their risk of violating the specification.

Safety guarantees alone, however, do not make the resulting mechanism easy to inspect. A synthesized shield is typically represented as a large mapping from states to sets of permitted actions, making it difficult for developers or operators to understand when and why the shield intervenes (Pranger et al., 2021). Decision trees (DTs) provide a natural representation for such mappings because their decisions can be inspected as predicates over state variables (Dwivedi et al., 2023). Yet directly encoding a shield as a DT can result in prohibitively large DTs, substantially limiting the practical value of the representation. Rieder et al. (2025) address this problem by decomposing an exact shield representation into a hierarchy of DTs. A high-level DT partitions states by safety risk, while localized DTs specify the actions permitted within their corresponding regions. This decomposition can substantially reduce the size of individual explanations. However, the overall representation of the shield often remains complex: recovering the complete shield requires reasoning over the high-level DT together with potentially many localized DTs. We instead seek a compact representation that captures the entire shield.

A key reason why DT representations of shields can become large is the requirement of *exact fidelity*: the DT must reproduce the action set of the original shield in every state. The central idea of this paper is therefore to relax the exact fidelity requirement and explicitly exploit the trade-off between fidelity to the original shield and the structural complexity of the resulting DT while maintaining safety. Such accuracy–complexity trade-offs are well established in the literature on interpretable surrogate models, where compact DTs are used to approximate complex classifiers or control policies (Krishnan et al., 1999; Bastani et al., 2018; Kinská et al., 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

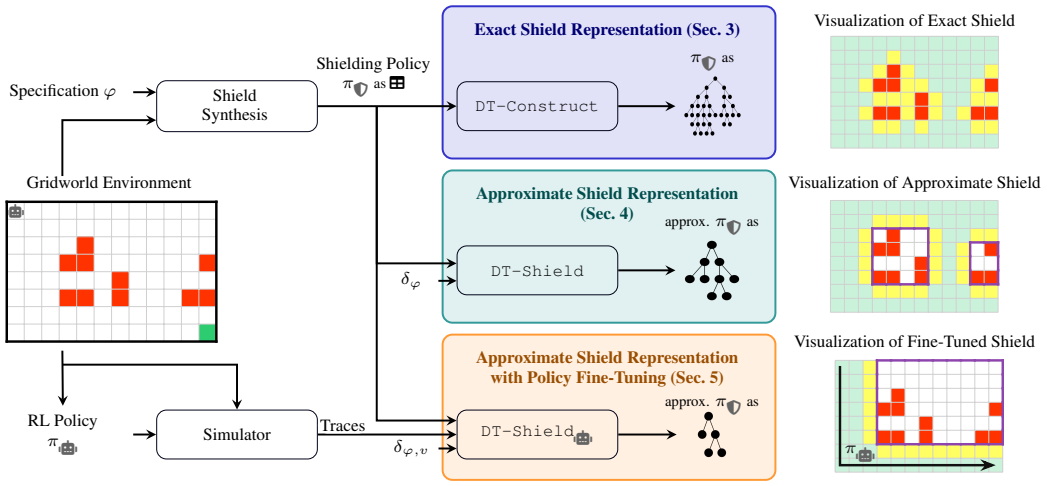

Figure 1: Overview of the construction of shield representations as decision trees. Top (blue): exact shield representation. Middle (green): approximate shield representation. Bottom (orange): approximate shield representation with policy fine-tuning.

We propose `DT-Shield`, a new algorithm for learning safety-preserving DT approximations of probabilistic safety shields. `DT-Shield` assigns each leaf only actions that are permitted by the original shield in every state represented by that leaf. Consequently, the resulting DT under-approximates the original shield: it may reject actions that the shield would permit, but it never permits an action rejected by the shield. The safety guarantees of the original shield are therefore preserved by construction.

To *balance shield permissiveness and representation size*, the user can specify a precision threshold, which determines the fraction of safe actions that may be misclassified as unsafe. However, the consequences of such misclassifications can vary substantially across actions: preventing the agent from executing some actions may lead to a significant reduction in performance, whereas excluding others may have only a limited effect. To account for these differences, we introduce two weighting schemes, depending on whether the shield is used during RL training or during deployment.

During training, actions are weighted according to their safety risk. Misclassifying an action whose induced risk is close to the maximum admissible risk is considered less costly than misclassifying an action that induces little risk. Intuitively, such an approximation may cause the agent to take a precautionary action slightly earlier than necessary, while avoiding situations in which the shield unnecessarily restricts the agent in clearly safe regions of the state space. This preserves the agent's ability to learn effective behavior while allowing for a more compact shield representation.

During deployment, the cost additionally depends on how frequently the agent visits the corresponding state and on the preference it assigns to the action. Consequently, misclassifying an action in a rarely visited state, or an action that the agent is unlikely to select, is considered less costly than misclassifying an action that the agent selects frequently. This allows the representation to focus its fidelity on those parts of the state-action space that are most relevant to the deployed policy.

Fig. 1 summarizes the computation of the different shield representations and also provides an overview of the paper's structure. In this example, an RL agent explores a slippery gridworld environment and must reach the green target cell while avoiding the red lava cells. Given the Markov decision process (MDP) $\mathcal{M}$ describing the environment and a temporal-logic safety specification characterizing unsafe behavior, a shield can be synthesized using standard shield-synthesis techniques (Alshiekh et al., 2018; Könighofer et al., 2025). The resulting shielding policy $\pi_{\shield}$ maps each MDP state to a set of permitted actions.

In Sec. 3, we describe how such a shielding policy can be represented exactly as a DT using standard construction algorithms (Rokach & Maimon, 2014). This exact representation serves as the basis for our approximate shield representations. In the gridworld visualization, the exact shield restricts the agent only where necessary to prevent transitions toward lava cells. Yellow cells indicate states

in which the shield restricts the agent's available actions, whereas in green cells all actions remain permitted. In Sec. 4, we introduce our `DT-Shield` algorithm for constructing approximate shield representations, assigning low approximation costs to misclassifying actions that are safe but whose risk lies close to the admissible safety threshold. The visualization on the right shows one possible approximate shield: it prevents the agent from entering the narrow region between the lava cells, thereby yielding a substantially smaller DT representation. White cells denote regions in which `DT-Shield` may classify originally safe actions as unsafe at comparatively low additional approximation cost. In Sec. 5, we extend this construction to compute an approximate DT that is tailored to a given RL policy $\pi_{\text{🤖}}$. In this policy-weighted variant, the approximation additionally accounts for the states and actions relevant to the policy's actual behavior. For example, if the policy primarily traverses the left and bottom regions of the gridworld, the remaining areas receive comparatively little weight. This results in a large set of white cells in regions that are rarely or never visited, where `DT-Shield` can discard safe actions at low approximation cost.

Our main contributions are as follows:

- We propose `DT-Shield`, which is, to the best of our knowledge, the first approach for computing *compact representations of shields*. In contrast to exact representations, `DT-Shield` deliberately trades a controlled amount of permissiveness for substantially reduced representation complexity.
- To guide the approximation for RL training, we introduce a weighting scheme based on the risk associated with an action. To adapt the approximation for deployment, we extend this scheme to account for state-visitation frequencies and the RL agent's action preferences.
- We evaluate `DT-Shield` on shields from several challenging application domains, including collision avoidance in the high-fidelity driving simulator CARLA (Dosovitskiy et al., 2017) and aircraft taxiing in the Boeing TaxiNet simulator (Fremont et al., 2020). We compare `DT-Shield` against state-of-the-art approaches for representing shields with DTs. Across all experiments, we find that allowing a small degree of conservative approximation can substantially reduce DT size while causing only minor degradation in the shielded RL agent's performance.

2 PRELIMINARIES

Markov Decision Processes. An *MDP* (Sutton & Barto, 1998) $\mathcal{M} = \langle \mathcal{S}, s_0, \mathcal{A}, \mathcal{P} \rangle$ consists of a finite state space $\mathcal{S}$ with initial state s_0, a finite action set $\mathcal{A}$, and a transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathit{Dist}(\mathcal{S})$. The actions enabled in $s \in \mathcal{S}$ are $\mathcal{A}(s) = \{a \in \mathcal{A} \mid \exists s', \mathcal{P}(s,a)(s') \neq 0\}$, and we require $|\mathcal{A}(s)| \geq 1$. Fixing a finite feature set $\mathcal{V} = \{x_1, \ldots, x_n\}$, every state is identified with its feature values $s = (\overline{x_1}, \ldots, \overline{x_n})$, $\overline{x_i} = s(x_i) \in \mathbb{Z}$.

Policies resolve action choices in MDPs. A memoryless *nondeterministic policy* is a relation $\pi : \mathcal{S} \to 2^{\mathcal{A}}$ and a memoryless *deterministic policy* is a function $\pi : \mathcal{S} \to \mathcal{A}$. In this paper, we use memoryless *randomized policies* which define a mapping $\pi : \mathcal{S} \to \mathit{Dist}(\mathcal{A})$. Applying a policy π to an MDP $\mathcal{M}$ induces a Markov chain $\mathcal{M}^\pi = \langle \mathcal{S}, s_0, \mathcal{P} \rangle$, where $\mathcal{P} : \mathcal{S} \to \mathit{Dist}(\mathcal{S})$ denotes the induced transition function.

A *trace* $\tau = s_0 s_1 \ldots s_n$ of $\mathcal{M}^\pi$ is a finite sequence of states such that $\mathcal{P}(s_i)(s_{i+1}) > 0$ for all $0 \leq i < n$. The probability of observing τ is given by $\mathbb{P}(\tau) = \prod_{0 \leq i < n} \mathcal{P}(s_i)(s_{i+1})$.

Probabilistic Model Checking. We express safety properties φ (e.g., that a set of unsafe states is never reached) in Computation Tree Logic (CTL) (Baier & Katoen, 2008), optionally subject to a finite horizon h. The unbounded case is obtained by setting $h = \infty$. Given an MDP $\mathcal{M}$ and a safety property φ, probabilistic model checking (Baier & Katoen, 2008) can compute the corresponding satisfaction probabilities, for example, using value iteration, policy iteration, or linear programming:

- $\mathbb{P}_{\mathcal{M}^\pi, \varphi} \colon \mathcal{S} \times \mathbb{N} \to [0,1]$, the probability that φ holds within h steps from $s \in \mathcal{S}$ under a policy π;
- $\mathbb{P}^{\min}_{\mathcal{M}, \varphi}(s,h) = \min_\pi \mathbb{P}_{\mathcal{M}^\pi, \varphi}(s,h)$ is the *minimal* expected probability in the MDP $\mathcal{M}$ *over all policies* from a state s within h steps.

Decision Trees. A *decision tree* (DT) (Breiman et al., 1984) represents a function f that maps inputs to a finite set of labels. In our setting, the input is a state and the label is a set of actions, i.e., $f \colon \mathcal{S} \to 2^{\mathcal{A}}$. Formally, a DT is a tuple $\mathcal{T} = \langle Tr, \Gamma, \rho, \Lambda, \lambda \rangle$, where Tr is a finite, rooted, ordered

Algorithm 1 `DT-Construct`

Require: Shielding policy $\pi_{\shield}$, $\mathcal{S}' \subseteq \mathcal{S}$, predicates Γ
Ensure: DT $\mathcal{T}_{\shield}$ representing $\pi_{\shield}$
1: **if** $\pi_{\shield}(s_1) = \pi_{\shield}(s_2) \forall s_1, s_2 \in \mathcal{S}'$ **then**
2: **return** leaf ℓ with label $\lambda(\ell) = \pi_{\shield}(s)$ for $s \in \mathcal{S}'$
3: **end if**
4: $p^* \leftarrow \arg\min_{p \in \Gamma} I(\pi_{\shield}, \mathcal{S}', p)$
5: ${\mathcal{S}'}^+ \leftarrow \{s \in \mathcal{S}' \mid p^*(s)\}$
6: ${\mathcal{S}'}^- \leftarrow \mathcal{S}' \setminus {\mathcal{S}'}^+$
7: Create node n with $\rho(n) = p^*$ and children
DT-CONSTRUCT$(\pi_{\shield}, {\mathcal{S}'}^+, \Gamma)$ and DT-CONSTRUCT$(\pi_{\shield}, {\mathcal{S}'}^-, \Gamma)$
8: **return** n

binary tree with inner nodes N and leaves L, Γ is a set of predicates, $\rho\colon N \to \Gamma$ assigns a predicate to each inner node, and $\lambda\colon L \to \Lambda$ assigns a label to each leaf. We use a predicate set Γ consisting of axis-aligned thresholds over the features of $\mathcal{S}$: $\Gamma = \{[x_i \sim c] \mid x_i \in \mathcal{V},\ c \in \mathbb{Z},\ \sim \in \{\leq, <, \geq, >, =\}\}$, where $[x_i \sim c]$ holds for s if $\overline{x_i} \sim c$. Given an input $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, the predicates determine a unique path from the root to a leaf in Tr: at inner node $\nu \in N$ on the path, the left child is selected if $\rho(\nu)$ holds for x, and the right child otherwise. Let $\ell_{\mathcal{T}}(x) \in L$ denote the reached leaf. The DT represents $f(x) = \lambda(\ell_{\mathcal{T}}(x))$, and we write $\mathcal{T}(x)$ for $f(x)$.

Reinforcement Learning. In reinforcement learning (RL) (Sutton & Barto, 1998), an agent interacts with an unknown MDP $\mathcal{M}$ with reward function $\mathcal{R}\colon \mathcal{S} \to \mathbb{R}$. The return of a trace $\tau = s_0 s_1 \ldots$ is $\mathtt{ret}_\tau = \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t)$, where $\gamma \in [0, 1]$ is the discount factor. The agent seeks a policy $\pi_{\robot}\colon \mathcal{S} \to \mathit{Dist}(\mathcal{A})$ maximizing the expected return.

3 Exact Shield Representation via Decision Trees

In this section, we first review the standard definition of probabilistic safety shields over MDPs (Alshiekh et al., 2018; Jansen et al., 2020; Könighofer et al., 2025) and then discuss how established DT-learning algorithms can be used to represent such shields exactly (Rokach & Maimon, 2014). Building on these foundations, Sections 4 and 5 introduce our `DT-Shield` algorithm for computing approximate shield representations.

Shields. A *shield* for an MDP $\mathcal{M}$ is a policy $\pi_{\shield}\colon \mathcal{S} \to 2^{\mathcal{A}}$ that restricts the actions available to an agent according to a safety specification φ. Following Jansen et al. (2020), the *risk* of violating φ from state $s \in \mathcal{S}$ within horizon h is the probability of a safety violation under the safest policy,

$$\mathit{risk}_{\mathcal{M},\varphi}(s, h) = \mathbb{P}^{\mathsf{min}}_{\mathcal{M},\neg\varphi}(s, h). \tag{1}$$

For $h \geq 1$, the risk of an action $a \in \mathcal{A}(s)$ fixes a as the first action and follows the safest policy thereafter,

$$\mathit{risk}_{\mathcal{M},\varphi}(s, a, h) = \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a)(s')\, \mathit{risk}_{\mathcal{M},\varphi}(s', h-1). \tag{2}$$

Given a user-defined safety threshold $\eta \in [0, 1]$, the shield $\pi_{\shield}\colon \mathcal{S} \to 2^{\mathcal{A}}$ permits, in each state $s \in \mathcal{S}$, all actions whose risk does not exceed the threshold:

$$\pi_{\shield}(s) = \{a \in \mathcal{A}(s) \mid \mathit{risk}_{\mathcal{M},\varphi}(s, a, h) < \eta\}. \tag{3}$$

Recursive Construction of $\mathcal{T}_{\shield}$. `DT-Construct` (Alg. 1) builds $\mathcal{T}_{\shield}$ recursively from $\pi_{\shield}$ and $\mathcal{S}$. Each recursive call operates on a subset of states $\mathcal{S}' \subseteq \mathcal{S}$, starting with $\mathcal{S}' = \mathcal{S}$. If the shield permits the same action set in all states of $\mathcal{S}'$, the call returns a leaf labeled with this action set (Line 2). Otherwise, Line 4 splits $\mathcal{S}'$ by the predicate that minimizes the impurity I, defined as follows. For a label $\gamma \in \Lambda$, let $q_\gamma(\pi_{\shield}, \mathcal{S}') = \frac{|\{s \in \mathcal{S}' \mid \pi_{\shield}(s) = \gamma\}|}{|\mathcal{S}'|}$ denote the fraction of states in $\mathcal{S}'$ with label γ. The label heterogeneity of $\mathcal{S}'$ is measured by its Shannon entropy $H(\pi_{\shield}, \mathcal{S}') = -\sum_{\gamma \in \Lambda} q_\gamma(\pi_{\shield}, \mathcal{S}') \log_2 q_\gamma(\pi_{\shield}, \mathcal{S}')$, omitting terms with $q_\gamma(\pi_{\shield}, \mathcal{S}') = 0$. For a predicate $p \in$

Algorithm 2 `DT-Shield`: Cost-guided construction of an approximate shield DT

Require: Shield $\pi_{\shield}$, states $\mathcal{S}$, predicates Γ, precision threshold δ_φ
Ensure: DT $\mathcal{T}_\approx$ approximating $\pi_{\shield}$
1: $C_g \leftarrow \sum_{s\in\mathcal{S}} \sum_{a\in\pi_{\shield}(s)} c(s,a)$
2: Initialize $\mathcal{T}_\approx$ with a single root leaf ℓ labeled via Eq. 4
3: Initialize a priority queue Q ordered by $C(\cdot)$, containing ℓ
4: **while** $1 - C(\mathcal{T}_\approx)/C_g < \delta_\varphi$ and $Q \neq \emptyset$ **do**
5: $\quad \ell \leftarrow \mathrm{popmax}(Q)$
6: $\quad p^* \leftarrow \arg\min_{p\in\Gamma} I(\pi_{\shield}, S_\ell, p)$
7: $\quad$ Split ℓ into ℓ^+ and ℓ^- using p^* and label them via Eq. 4
8: $\quad$ **for** $\ell' \in \{\ell^+, \ell^-\}$ **do**
9: $\quad\quad$ **if** $C(\ell') > 0$ **then**
10: $\quad\quad\quad$ Insert ℓ' into Q
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: **end while**
14: **return** $\mathcal{T}_\approx$

Γ, let ${\mathcal{S}'}_p^+ = \{s \in \mathcal{S}' \mid p(s)\}$ and ${\mathcal{S}'}_p^- = \mathcal{S}' \setminus {\mathcal{S}'}_p^+$ denote the states for which p is true and false, respectively. The impurity of the split induced by p is the weighted entropy $I(\pi_{\shield}, \mathcal{S}', p) = \frac{|{\mathcal{S}'}_p^+|}{|\mathcal{S}'|} H(\pi_{\shield}, {\mathcal{S}'}_p^+) + \frac{|{\mathcal{S}'}_p^-|}{|\mathcal{S}'|} H(\pi_{\shield}, {\mathcal{S}'}_p^-)$. `DT-Construct` selects $p^* = \arg\min_{p\in\Gamma} I(\pi_{\shield}, \mathcal{S}', p)$ and recurses in Line 7 on ${\mathcal{S}'}_{p^*}^+$ and ${\mathcal{S}'}_{p^*}^-$. Since `DT-Construct` only creates a leaf once all its states agree on the permitted action set, the resulting DT reproduces the shield in every state. We therefore call $\mathcal{T}_{\shield}$ *exact*, i.e., $\mathcal{T}_{\shield}(s) = \pi_{\shield}(s)$ for all $s \in \mathcal{S}$.

4 Approximate Shield Representation via Decision Trees

The exact representation of $\pi_{\shield}$ produced by `DT-Construct` can result in prohibitively large DTs. We therefore introduce `DT-Shield`, an algorithm for constructing compact DT approximations of shields while preserving their safety guarantees by construction. Rather than splitting until all states within a leaf agree on the same action set, `DT-Shield` stops the construction early and deliberately trades permissiveness for a smaller DT. The method relies on two key components: (i) a *labeling function* that assigns each early-stopped leaf an under-approximating action set, and (ii) an *approximation cost* that quantifies the loss of relevant permissiveness induced by this approximation. Guided by this cost, `DT-Shield` greedily refines the most costly leaves until the DT's precision reaches a user-defined precision requirement δ_φ.

`DT-Shield` (Alg. 2). In contrast to `DT-Construct`, `DT-Shield` maintains a global priority queue of leaves instead of recursion. This allows splitting leaves based on their relevance until the user-defined precision requirement is met. In Line 3, the queue is initialized with a single root leaf covering all states. The label of a leaf is defined as follows: Let $S_\ell = \{s \in \mathcal{S} \mid \ell_{\mathcal{T}}(s) = \ell\}$ denote the set of states reaching a leaf ℓ. The labeling function λ assigns to ℓ only those actions that $\pi_{\shield}$ permits in every state of S_ℓ:

$$\lambda(\ell) = \bigcap_{s\in S_\ell} \pi_{\shield}(s). \tag{4}$$

Due to the intersection, $\lambda(\ell)$ can restrict actions permitted by $\pi_{\shield}$. We define an approximation cost of action a in state s to quantify the loss of permissiveness:

$$c(s,a) = \eta - \mathit{risk}_{\mathcal{M},\varphi}(s,a,h). \tag{5}$$

This measures the distance to the safety threshold. We expect that removing an action whose risk is close to the threshold η restricts the agent less than removing an action with little or no risk, which restricts the agent in clearly safe situations. The defined cost is nonnegative since $a \in \pi_{\shield}(s)$.

The approximation cost associated with a leaf ℓ is then the sum of costs $c(s, a)$ over all actions a removed via the leaf labeling $\lambda(\ell)$, and the cost of the DT $\mathcal{T}$ is the sum of the costs of its leaves:

$$C(\ell) = \sum_{s \in S_\ell} \sum_{a \in \pi_{\text{🛡}}(s) \setminus \lambda(\ell)} c(s, a), \qquad C(\mathcal{T}) = \sum_{\ell \in L} C(\ell). \tag{6}$$

The costs $C(\ell)$ are high whenever the approximation through $\lambda(\ell)$ removes a large set of allowed actions or actions with low risk.

In each iteration, `DT-Shield` pops the leaf with the highest cost (Line 5) and splits it using the impurity-minimizing predicate of `DT-Construct` (Line 6). Line 7 labels both resulting children via Eq. 4 and inserts them into the queue if their cost is positive. Leaves with zero cost already represent the shield exactly.

To measure the current quality of the DT, we define its (weighted) precision as:

$$\text{Prec} = 1 - \frac{C(\mathcal{T})}{\sum_{s \in \mathcal{S}} \sum_{a \in \pi_{\text{🛡}}(s)} c(s, a)}. \tag{7}$$

Since $C(\mathcal{T})$ quantifies the total weighted loss of safe actions across all states, this formula quantifies the fraction of the total safe-action cost that the DT preserves. Line 4 terminates construction once $\text{Prec} \geq \delta_\varphi$ for a user-defined precision requirement δ_φ or when the queue is empty. In the latter case, all leaves are exact, and `DT-Shield` coincides with `DT-Construct`.

Safety Guarantees. The conservative leaf assignment ensures that approximation never introduces an action that is absent from the original shield.

Theorem 1 (Safety Preservation)**.** *For every $s \in \mathcal{S}$, $\mathcal{T}_{\approx}(s) \subseteq \pi_{\text{🛡}}(s)$.*

Proof. Let $\ell = \ell_{\mathcal{T}_{\approx}}(s)$ be the leaf reached by s. Since $s \in S_\ell$ and $\lambda(\ell) = \bigcap_{s' \in S_\ell} \pi_{\text{🛡}}(s')$, we have $\mathcal{T}_{\approx}(s) = \lambda(\ell) \subseteq \pi_{\text{🛡}}(s)$. □

Thus, every action represented by `DT-Shield` is also permitted by the original shield. The approximation can therefore reduce permissiveness but cannot admit an action prohibited by the shield, so the safety guarantees of the original shield are preserved.

5 Approximate Shield Representation with Policy Fine-Tuning

Alg. 2 treats all state-action pairs alike. In practice, however, a deployed policy visits only a subset of states. Furthermore, discarding an action the agent frequently selects affects its performance far more than discarding an action in a state it never visits or an action that has low probability in $\pi_{\text{🤖}}$ (see Fig. 1). We therefore fine-tune the DT construction to the behavior of a given RL agent by updating the cost function with a policy-tuned weight.

Let $\pi_{\text{🤖}}$ denote the agent's trained policy. We collect a set of traces $\mathcal{D}$ by executing $\pi_{\text{🤖}}$ under the exact shield $\pi_{\text{🛡}}$ and let $N_{\mathcal{D}}(s) = \sum_{\tau \in \mathcal{D}} |\{i \mid \tau_i = s\}|$ denote the number of visits to s. For each state-action pair (s, a), we compute a weight based on the probability of a in $\pi_{\text{🤖}}$ and the visitation count of s:

$$w_{\text{🤖}}(s, a) = \begin{cases} 0 & \text{if } a \notin \pi_{\text{🛡}}(s) \\ \varepsilon & \text{if } a \in \pi_{\text{🛡}}(s) \text{ and } N_{\mathcal{D}}(s) = 0 \\ N_{\mathcal{D}}(s) \cdot \pi_{\text{🤖}}(s)(a) & \text{otherwise,} \end{cases} \tag{8}$$

Rather than considering only the actions actually taken along $\mathcal{D}$, Eq. 8 weights every safe action according to the agent's preference. This captures how strongly the policy relies on each action: removing a highly preferred action is more likely to affect the agent's performance, whereas actions with (near-)zero probability incur (near-)zero cost and can therefore be discarded with little impact. The small constant $\varepsilon > 0$ (e.g. 10^{-6}) ensures that unvisited states are still taken into account, which is important for learning a DT with high precision. Its small value assigns these states negligible importance compared to states encountered in $\mathcal{D}$.

Weighting the approximation cost Eq. 5 by $w_{\text{🤖}}$ yields the *policy-weighted approximation cost*

$$c_{\text{🤖}}(s, a) = w_{\text{🤖}}(s, a) \cdot c(s, a). \tag{9}$$

Replacing c by $c_{🤖}$ in Eq. 6 yields the policy-weighted cost of a leaf and of a DT,

$$C_{🤖}(\ell) = \sum_{s \in S_\ell} \sum_{a \in \pi_{🛡}(s) \setminus \lambda(\ell)} c_{🤖}(s,a), \qquad C_{🤖}(\mathcal{T}) = \sum_{\ell \in L} C_{🤖}(\ell), \tag{10}$$

and, analogously to Eq. 7, the *policy-weighted precision*

$$\mathrm{Prec}_{🤖} = 1 - \frac{C_{🤖}(\mathcal{T})}{\sum_{s \in \mathcal{S}} \sum_{a \in \pi_{🛡}(s)} c_{🤖}(s,a)}. \tag{11}$$

We refer to Alg. 2 run with these quantities and a precision threshold $\delta_{\varphi,v}$ as `DT-Shield`$_{🤖}$. The resulting DT $\mathcal{T}_{🤖}$ represents $\pi_{🛡}$ precisely in frequently visited states and for the actions $\pi_{🤖}$ prefers, while approximating it coarsely elsewhere.

Safety Guarantees. `DT-Shield`$_{🤖}$ differs from `DT-Shield` only in the cost used to rank leaves in the priority queue and to evaluate the stopping criterion, and `DT-Shield` is recovered as the special case $w_{🤖}(s,a) = 1$ for all $a \in \pi_{🛡}(s)$ and $\delta_{\varphi,v} = \delta_{\varphi}$. Since the labeling function Eq. 4 is unchanged, Theorem 1 carries over directly, and $\mathcal{T}_{🤖}$ preserves the safety guarantees of $\pi_{🛡}$.

6 Evaluation

We investigate the achievable size–performance trade-off of approximate shield representations:

RQ1: To what extent can `DT-Shield` reduce DT size while preserving the performance of agents trained under the approximate shield?

RQ2: To what extent can policy-aware fine-tuning with `DT-Shield`$_{🤖}$ further reduce DT size while preserving the performance of an agent trained under the exact shield?

Benchmarks. We use six benchmarks: *Windy City* (Könighofer et al., 2025) (a drone in stochastic wind), *Museum Heist* (a thief evading a guard), *Pac-Man* (Gross et al., 2022), *Lunar Lander* (Towers et al., 2024), *TaxiNet* (Fremont et al., 2020) (autonomous Boeing taxiing), and *CARLA* (Dosovitskiy et al., 2017) (driving simulation with the goal of avoiding a pedestrian). For the first four tasks, we train RL agents. App. A describes the environments, rewards, and safety properties in more detail.

Setup. We synthesize shields using `tempespy` (Pranger & Könighofer, 2026). For each RL environment we train 5 seeds of MaskablePPO for 200,000 steps. Each agent is evaluated on 100 episodes. As the trained agents are independent units, we treat each seed's mean reward over its 100 episodes as a single sample and report the mean and the 95% confidence interval (Student's t) across the 5 seeds. In RQ2, the exact and the approximate shield are evaluated on the same agents, so we also report paired differences. All experiments ran in Docker containers on AMD EPYC 9274F CPUs, each limited to one core and 12 GB of memory.

Baselines. We compare against the exact shield DT built by `DT-Construct` $\mathcal{T}_{🛡}$ and against the sum of the sizes of all DTs by Rieder et al. (2025).

RQ1: Approximate shields. Table 1 reports results at a relatively high precision of 90% or above, allowing only a limited amount of approximation error. Despite this stringent precision requirement, `DT-Shield`$_{🤖}$ reduces DT size by 56–90% relative to the exact representation and consistently produces substantially smaller trees than the existing baseline (Rieder et al., 2025). Crucially, these reductions come at little cost to agent performance: across all benchmarks with trained agents, policies trained under the approximate shields achieve performance comparable to those trained under the exact shield. Fig. 2 examines this trade-off across a broader range of precision levels. As precision decreases to 0.6, DT size is reduced by approximately another order of magnitude, while agent performance remains largely unaffected. Only below a precision of 0.6 do we observe substantial performance degradation. These results indicate that `DT-Shield` can aggressively compress shield representations while preserving policy performance over a broad range of approximation levels. Full results are provided in App. B.

RQ2: Fine-tuning to a trained agent. Table 2 reports the performance of RL agents with the shield-fine-tuning of `DT-Shield`$_{🤖}$. Focusing the approximation on the states and actions relevant to the agent yields considerably smaller trees than in RQ1: `DT-Shield` reduces the DT size by

Table 1: RQ1: DT size of $\mathcal{T}_{◐}$, the baseline by Rieder et al. (2025), and DT-Shield's smallest approximation with at least 90% precision (Prec.), together with the size reduction (Red.) relative to $\mathcal{T}_{◐}$. Reward (mean $\pm$ 95% confidence interval) is that of an agent trained and evaluated under the exact shield $\pi_{◐}$ and under the approximation $\mathcal{T}_{\approx}$; agents of different columns are trained separately.

		#Nodes of Baselines		DT-Shield			Reward	
Environment	$\lvert\mathcal{S}\rvert$	$\mathcal{T}_{◐}$	Rieder et al. (2025)	Prec.	#Nodes	Red.	$\pi_{◐}$	$\mathcal{T}_{\approx}$
Windy City	1,000	231	281	90%	101	56%	0.60 ± 0.04	0.63 ± 0.09
Museum Heist	195,455	39,871	60,234	90%	4,025	90%	2.05 ± 0.11	2.18 ± 0.08
Lunar Lander	24,957	377	747	90%	73	81%	209.83 ± 8.15	202.48 ± 9.51
Pac-Man	8,035	1,475	1,627	90%	557	62%	40.96 ± 0.10	40.97 ± 0.04
TaxiNet	587	153	223	90%	63	59%	–	–
CARLA	4,205	65	197	93%	17	74%	–	–

Table 2: RQ2: DT size and reward for an agent trained under $\pi_{◐}$, evaluated with the smallest policy-weighted approximation DT-Shield$_{🤖}$ with at least 95% weighted precision (Prec.). Columns as in Table 1, but both rewards are of the same 5 agents.

		#Nodes of Baselines		DT-Shield$_{🤖}$			Reward	
Environment	$\lvert\mathcal{S}\rvert$	$\mathcal{T}_{◐}$	Rieder et al. (2025)	Prec.	#Nodes	Red.	$\pi_{◐}$	$\mathcal{T}_{🤖}$
Windy City	1,000	231	281	96%	44	81%	0.60 ± 0.04	0.66 ± 0.04
Museum Heist	195,455	39,871	60,234	95%	392	99%	2.05 ± 0.11	1.95 ± 0.07
Lunar Lander	24,957	377	747	97%	21	94%	209.83 ± 8.15	205.11 ± 9.38
Pac-Man	8,035	1,475	1,627	95%	445	70%	40.96 ± 0.10	40.96 ± 0.07

70–99% relative to the exact representation. The effect is most pronounced for large shields: in Museum Heist, the DT shrinks from 39,871 to 392 nodes, an order of magnitude smaller than the approximation obtained in RQ1. Despite this drastic reduction, deploying the fine-tuned shields has little impact on performance: the agents' reward decreases by at most 5%, and in Pac-Man the difference to the exact shield is indistinguishable. The full results are contained in App B.

Ablation. We ablate the cost computation of DT-Shield$_{🤖}$ on the agents of RQ2 (Fig. 14 in App. B). We compare four settings: $c_{🤖}$, c, uniform cost, and uniform cost with $w_{🤖}(s, a)$. We observe that both components contribute, and the combination yields the best results.

Limitations. While Huysmans et al. (2011) show that smaller DTs can improve interpretability, tree size is not the only factor that determines how easily a DT can be understood. Even a compact DT may remain difficult to interpret if its internal predicates are unintuitive or poorly aligned with human-understandable concepts. An important direction for future work is therefore to incorporate explainability more explicitly into the construction process, for example, by learning DTs with simpler, semantically meaningful predicates and evaluating their comprehensibility for human users. As a second limitation, although users can control the degree of approximation through the precision thresholds δ_{φ} and $\delta_{\varphi,v}$, these thresholds do not translate directly into a predictable reduction in tree size. However, if the resulting compression is insufficient, a new DT can be computed efficiently with a larger approximation budget by continuing from the previously obtained DT rather than restarting the construction from scratch.

7 Related Work

Safe reinforcement learning via shielding. Alshiekh et al. (2018) introduced shielding for RL, constructing a shield from a model of the safety-relevant dynamics that blocks actions leading to a safety violation. Jansen et al. (2020) generalized this idea to a quantitative setting, computing *probabilistic shields* that bound the probability of a safety violation. Comprehensive overviews of shielding and safe RL are given by Könighofer et al. (2025) and Gu et al. (2024). Tools such as `tempestpy` (Pranger & Könighofer, 2026) synthesize shields from temporal-logic specifications.

Decision-tree representations of controllers and shields. DTs are widely used in explainable AI because of their intuitive, human-readable structure (Dwivedi et al., 2023). Krishnan et al. (1999) introduced the idea of extracting DTs as compact surrogates for opaque models such as neural net-

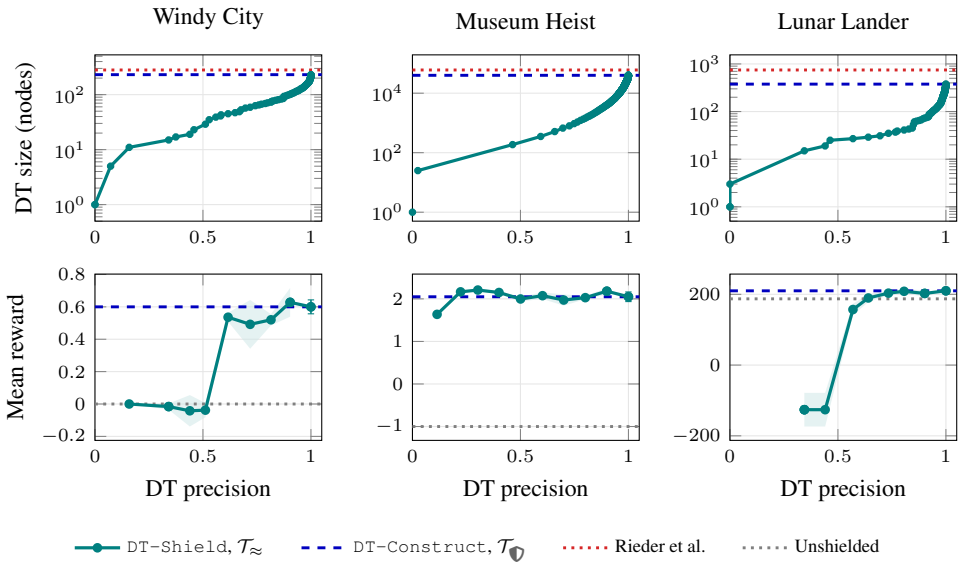

Figure 2: RQ1. Top: size of $\mathcal{T}_{\approx}$ produced by `DT-Shield` at different precisions; dashed: exact shield DT; dotted: the baseline of Rieder et al. (2025). Bottom: mean reward of agents trained and evaluated under $\mathcal{T}_{\approx}$, with the 95% confidence interval; dashed: $\mathcal{T}_{\mathbb{O}}$, dotted: unshielded agent. The size panels of all environments are in Appendix B.

works, establishing the fundamental trade-off between fidelity and structural complexity. Recently, Kinská et al. (2026) introduced `dtControl2+`ε, which learns DT controllers that trade optimality with explainability by allowing to deviate from the optimal policy within a user-defined bound ε. However, this approach targets *optimal policies* and does not account for the safety-specific structure of shields. The most directly related work is that of Rieder et al. (2025), who represent shield decisions with a *hierarchy* of DTs that require exact fidelity and do not produce a single compact representation of the shield. In contrast, `DT-Shield` relaxes exact fidelity to obtain a *single* approximate DT whose error is safety-preserving by construction. Lastly, the literature has also considered representing RL policies directly as DTs (Topin et al., 2021; Bastani et al., 2018; Das Gupta et al., 2015; Kohler et al., 2024; Demirović et al., 2022). We instead focus on the shield.

8 Conclusion

We presented `DT-Shield`, an approach for constructing compact DT representations of probabilistic safety shields by trading permissiveness for representation size. By restricting leaf labels to actions permitted by the original shield, the approximation preserves the shield's safety guarantees by construction. We further introduced weighted approximation costs, such that the cost of misclassifying an action in a given state depends both on the risk induced by that action and on its relevance to a given RL policy. Across six benchmarks, `DT-Shield` reduced DT size by 56–90% at 90% precision, while policy fine-tuning achieved reductions of 70–99% at 95% weighted precision, with mean reward remaining within 5% of the exact shield. These results demonstrate that substantial reductions in shield representation complexity are possible while retaining both formal safety guarantees and agent performance. Future work will investigate automated predicate generation, including the use of generative AI to propose simpler and semantically meaningful predicates that improve the interpretability of the resulting DTs.

AI use statement

We used generative AI tools to provide feedback on parts of our research methodology and experiments, and to implement parts of our codebase, mostly the evaluation and plotting/aggregation pipeline used to produce our results (the core DT-construction code was written by the authors). We used these tools to interpret experimental results while iterating on the implementation, e.g. tracing

size/precision anomalies back to specific code paths using real run data. Proofs and the concepts of the algorithms were created by the authors. We did not use generative AI tools to generate synthetic data sets for our reported experiments, to clean or reformat our datasets, to support qualitative or thematic data analysis, or to assist with translation.

Additionally, we used generative AI to improve the paper's readability.

We have reviewed all AI-assisted work. At least two authors checked the code and text produced by these models. Code was furthermore verified by running it and comparing results over several stages of the development process. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Ethics statement

We do not think that our paper raises any questions regarding the Code of Ethics.

Reproducibility statement

To ensure reproducibility, the code that produced all plots of this paper was run within a docker container. The container and the code basis are attached to this submission in the supplementary material. Datasets are described in more detail in the appendix and base files are also included in the supplementary material.

References

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe Reinforcement Learning via Shielding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, pp. 2669–2678. AAAI Press, 2018. doi:10.1609/aaai.v32i1.11797.

Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008. ISBN 978-0-262-02649-9.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable Reinforcement Learning via Policy Extraction. In *Annual Conference on Neural Information Processing Systems 2018, (NeurIPS 2018)*, pp. 2499–2509, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/e6d8545daa42d5ced125a4bf747b3688-Abstract.html.

Leo Breiman, J. H. Friedman, Richard A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. ISBN 0-534-98053-8.

Ujjwal Das Gupta, Erik Talvitie, and Michael Bowling. Policy Tree: Adaptive Representation for Policy Gradient. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Feb. 2015. doi:10.1609/aaai.v29i1.9613.

Emir Demirović, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J. Stuckey. MurTree: Optimal Decision Trees via Dynamic Programming and Search. *Journal of Machine Learning Research*, 23(26):1–47, 2022. URL http://jmlr.org/papers/v23/20-520.html.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning, (CoRL 2017)*, volume 78 of *Proceedings of Machine Learning Research*, pp. 1–16. PMLR, 2017. URL https://proceedings.mlr.press/v78/dosovitskiy17a.html.

Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Omer F. Rana, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, and Rajiv Ranjan. Explainable AI (XAI): Core Ideas, Techniques, and Solutions. *ACM Computing Surveys*, 55(9):194:1–194:33, 2023. doi:10.1145/3561048.

Daniel J. Fremont, Johnathan Chiu, Dragos D. Margineantu, Denis Osipychev, and Sanjit A. Seshia. Formal Analysis and Redesign of a Neural Network-Based Aircraft Taxiing System with VerifAI. In *International Conference on Computer Aided Verification (CAV 2020)*, volume 12224 of *Lecture Notes in Computer Science*, pp. 122–134. Springer, 2020. doi:10.1007/978-3-030-53288-8_6.

Dennis Gross, Nils Jansen, Sebastian Junges, and Guillermo A. Pérez. COOL-MC: A Comprehensive Tool for Reinforcement Learning and Model Checking. In Wei Dong and Jean-Pierre Talpin (eds.), *8th International Symposium on Dependable Software Engineering. Theories, Tools, and Applications, (SETTA 2022)*, volume 13649 of *Lecture Notes in Computer Science*, pp. 41–49. Springer, 2022. doi:10.1007/978-3-031-21213-0_3.

Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, and Alois Knoll. A Review of Safe Reinforcement Learning: Methods, Theories, and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):11216–11235, 2024. doi:10.1109/tpami.2024.3457538.

Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011. doi:10.1016/j.dss.2010.12.003.

Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. Safe Reinforcement Learning Using Probabilistic Shields. In Igor Konnov and Laura Kovács (eds.), *Proceedings of the 31st International Conference on Concurrency Theory, (CONCUR 2020)*, volume 171 of *LIPIcs*, pp. 3:1–3:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/lipics.concur.2020.3.

Tereza Kinská, Jan Křetínský, Tobias Meggendorfer, Sabine Rieder, and Maximilian Weininger. dtControl2+ε: Trading Optimality for Explainability in MDPs via Decision Trees. In *Conference on Formal Methods in Computer-Aided Design (FMCAD 2026)*, pp. 586–598. TU Wien Academic Press, 2026. doi:10.34727/2026/isbn.978-3-85448-093-8_61.

Hector Kohler, Quentin Delfosse, Riad Akrour, Kristian Kersting, and Philippe Preux. Interpretable and Editable Programmatic Tree Policies for Reinforcement Learning. In *European Workshop on Reinforcement Learning*, 2024. URL `https://openreview.net/forum?id=zafp5CwoTq`.

Bettina Könighofer, Roderick Bloem, Nils Jansen, Sebastian Junges, and Stefan Pranger. Shields for Safe Reinforcement Learning. *Communications of the ACM*, 68(11):80–90, 2025. ISSN 0001-0782. doi:10.1145/3715958.

R. Krishnan, G. Sivakumar, and P. Bhattacharya. Extracting decision trees from trained neural networks. *Pattern Recognition*, 32(12):1999–2009, 1999. doi:10.1016/s0031-3203(98)00181-2.

Stefan Pranger and Bettina Könighofer. tempestpy: Easy-to-Use Shielded Reinforcement Learning. In *International Symposium on Leveraging Applications of Formal Methods, Verification, and Validation (ISoLA 2026)*, 2026.

Stefan Pranger, Bettina Könighofer, Lukas Posch, and Roderick Bloem. TEMPEST - Synthesis Tool for Reactive Systems and Shields in Probabilistic Environments. In *19th International Symposium on Automated Technology for Verification and Analysis, (ATVA 2021)*, volume 12971 of *Lecture Notes in Computer Science*, pp. 222–228. Springer, 2021. doi:10.1007/978-3-030-88885-5_15.

Sabine Rieder, Stefan Pranger, Debraj Chakraborty, Jan Křetínský, and Bettina Könighofer. Explainably Safe Reinforcement Learning. In *Annual Conference on Neural Information Processing Systems, (NeurIPS 2025)*, 2025. URL `http://proceedings.nips.cc/paper_files/paper/2025/hash/e68ef1d02b01a7d62a3ff19f57fdc509-Abstract-Conference.html`.

Lior Rokach and Oded Maimon. *Data Mining with Decision Trees - Theory and Applications*, volume 81 of *Series in Machine Perception and Artificial Intelligence*. World Scientific, 2nd edition, 2014. ISBN 978-981-4590-07-5. doi:10.1142/9097.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning - An Introduction*. Adaptive computation and machine learning. MIT Press, 1998. ISBN 978-0-262-19398-6.

Nicholay Topin, Stephanie Milani, Fei Fang, and Manuela Veloso. Iterative Bounding MDPs: Learning Interpretable Policies via Non-Interpretable Methods. In *AAAI*, pp. 9923–9931. AAAI Press, 2021. doi:10.1609/AAAI.V35I11.17192.

Mark Towers, Ariel Kwiatkowski, Jordan K. Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *CoRR*, abs/2407.17032, 2024. doi:10.48550/arxiv.2407.17032.

A Environment Details

A.1 Windy City

Windy City (Könighofer et al., 2025) is a 25×20 MiniGrid gridworld in which a drone must deliver a package to a goal cell while avoiding crashes into houses. Scattered "windy" tiles push the agent stochastically off its intended direction, so a single misstep can carry it into a house several cells away. The environment is depicted in Fig. 3. The safety property φ requires the drone to steer clear of all houses (gray cells), with threshold $\eta = 0.05$, meaning the shield never permits an action whose risk of eventually crashing into a house exceeds 5%. The reward is $+1$ for reaching the goal, -1 for crashing into a house, and 0 if the agent has not reached the goal within 200 steps. Sizes and performance are reported in Tables 1 and 2.

A.2 Museum Heist

Museum Heist (see Fig. 4) is a 19×19 MiniGrid gridworld in which a thief must steal treasures and reach an exit while a guard actively pursues it. The safety property φ requires that the thief is never caught by the guard, with threshold $\eta = 0.001$. The reward is 0.2 per stolen treasure and 1 for escaping, but an agent that is caught receives an overall reward of -1, no matter how much it collected before. This environment clearly shows the trade-off between size and precision. In RQ1, the mean reward of agents trained under the 90%-precision approximation is not lower than under the exact shield, but the difference is within the confidence interval. In RQ2, the fine-tuned DT lowers the reward of the agent by 4.8%. A possible explanation for the first effect is that the trained RL agents are often reckless and get caught often. This is prohibited by the shield, and, since approximating shields makes them more restrictive, the approximate shield allows even less risky actions. Agents trained under approximations as coarse as 11% precision still obtain a reward of 1.6 (RQ1, Fig. 10), and even a DT consisting of a single node yields 1.55 for the fine-tuned agent (RQ2, Fig. 12), compared to -1 for an unshielded agent, which is always caught.

A.3 Pac-Man

Pac-Man (Gross et al., 2022) is an 11×7 maze gridworld in which the agent navigates a corridor layout while a ghost pursues it. For a visualization see Fig. 5 The ghost ignores the agent with probability 0.15 and moves randomly instead, making the pursuit imperfect rather than a fully optimal adversary. The safety property φ requires Pac-Man to never collide with the ghost, with threshold $\eta = 0.05$. The reward is $+1$ for every newly eaten pellet. Sizes and performance are reported in Tables 1 and 2. We observe no difference in reward between the approximation and the exact shield at 90% precision in RQ1 and at 95% weighted precision in RQ2, and the reward degrades gracefully rather than collapsing for smaller trees (Fig. 10 and 12) for smaller precisions.

We additionally evaluate a harder variant with two ghosts, which results in a larger state space (514,272 vs. 8,035 states). At 90% precision, `DT-Shield`'s tree has 11,099 nodes (a 75% reduction), and agents trained under it obtain a reward of 40.2 ± 2.9 (exact shield: 40.0 ± 3.9). For RQ2, the fine-tuned tree has 6,159 nodes at a weighted precision of 95% (an 86% reduction), with a reward of 40.0 ± 3.7. Full curves for this variant are in App. B.

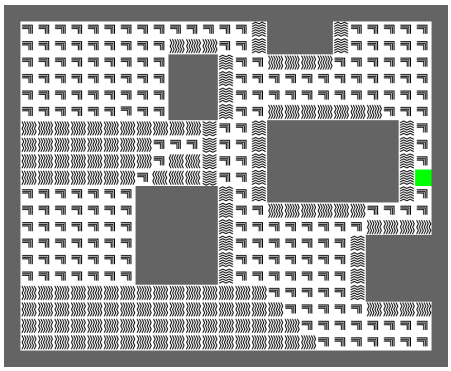

Figure 3: The Windy City environment. The green cell is the goal; the cell patterns encode the wind acting on the drone.

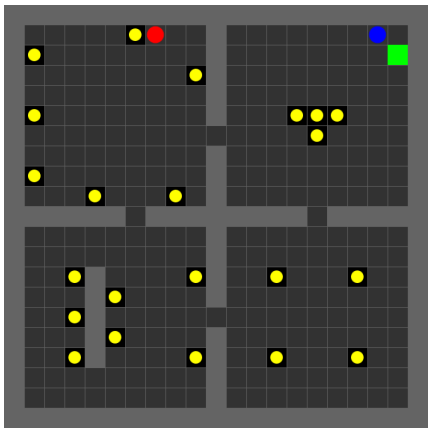

Figure 4: The Museum Heist environment: the agent (red circle) and the guard (blue circle), the treasures (yellow), and the exit (green).

A.4 LUNAR LANDER

Lunar Lander (Towers et al., 2024) is a continuous-control task in which an agent controls a descending lander using four actions (no-op and three thrusters) to keep its tilt angle and angular velocity within safe bounds. Unlike the gridworlds, states here are numeric (angle, angular velocity, ...) rather than discrete positions, so the shield's DT splits on these continuous features directly. The safety property φ requires that a bounded-unsafe tilt configuration is never reached within a 20-step horizon, with threshold $\eta = 0.05$. The reward is that of Gymnasium's LunarLander-v3 (a shaping reward for moving closer to the landing pad, slower and more upright, small fuel costs, and -100 for crashing or leaving the screen), with the change that an episode ends as soon as the first leg touches the ground, since the MDP model we used is a free-flight model and only valid until then. The final reward is the shaping term plus $+100$ if the tilt at touchdown is below 20° (the shield's safe-angle margin) and -100 otherwise. The exact tree (377 nodes) is already smaller than the baseline by (Rieder et al., 2025) (747 nodes). In Lunar Lander the size reduction leads to performance drop quickly. Already below about 60% precision, agents trained under the approximation lose reward quickly (RQ1, Fig. 10), and the fine-tuned trees of RQ2 reach the exact shield's reward with 21 nodes, but lose most of it when pruned further to 17 nodes (Fig. 12).

A.5 TAXINET

TaxiNet (Fremont et al., 2020) models an aircraft taxiing along a runway centerline, based on Boeing's vision-based taxiing benchmark. It is illustrated in Fig. 6. The state consists of the aircraft's cross-track error (lateral offset from the centerline) and its heading error. The agent steers left, right,

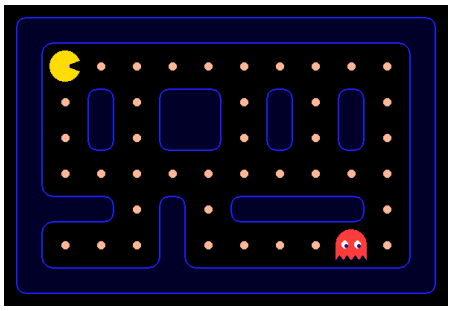

Figure 5: The Pac-Man environment with one ghost. This figure is AI-generated based on the underlying Prism file.



Figure 6: The TaxiNet setting (Fremont et al., 2020). This is the view of the aircraft's camera while taxiing along the runway centerline.

or holds course. The ground response to steering is stochastic: a steering command has a 90% chance of over-correcting. This setting is taken from Rieder et al. (2025). The safety property φ is not leaving the centerline corridor, with threshold $\eta = 0.05$. As for most environments, the exact tree (153 nodes) is smaller than the DTs of Rieder et al. (2025) (223 nodes). At 90% precision, `DT-Shield` reduces it to 63 nodes (Table 1, Fig. 2). TaxiNet has no trained agent in our pipeline as the simulator is not open source, so only the size results apply.

A.6 CARLA

CARLA (Dosovitskiy et al., 2017) is a high-fidelity driving simulator. We model a car approaching a marked crosswalk where a pedestrian is walking towards, and then across, the road. At every step the car either *Drives* (continues at a fixed speed) or *Brakes* (irreversibly enters a braking sequence), while the pedestrian's position updates stochastically, taking a small, medium, or large step each time. The safety property φ requires that the car does not hit the pedestrian within the next 20 steps (a 5 s horizon), with threshold $\eta = 0.05$. The exact tree produced by `DT-Construct` (65 nodes) is already $3\times$ smaller than the DTs of Rieder et al. (2025) (197 nodes) and approximation shrinks it further, to 17 nodes at 93% precision and as few as 5 nodes at 61% precision (a 92% size reduction; Fig. 2, rightmost panel). As with TaxiNet, no trained agent is evaluated here, so only RQ1 applies.

Model-Size Scaling. CARLA also lets us test how `DT-Shield` scales with model size, by varying two of its parameters, namely $LIMIT\_DX$, the initial distance (in meters) between the car and the pedestrian, and $LIMIT\_DC$, the initial distance from the pedestrian to the crosswalk. We scale both jointly with a factor k ($LIMIT\_DX = 9k$ m, $LIMIT\_DC = k$ m). Sweeping k from 14 to 130 yields models ranging from 1,277 to 64,892 reachable states. Fig. 7 shows that `DT-Construct`'s exact tree grows markedly slower than the DTs of Rieder et al. (2025) as the model grows, and the absolute gap widens with model size, indicating that our compact representation scales favorably even without approximation. When fixing the target precision at $\geq 90\%$ (the smallest tree at each k that still reaches it) we observe that approximation keeps the tree between 11 and 35 nodes across the *entire* range of model sizes above 1,300 states (where all trees consist of a single node), instead

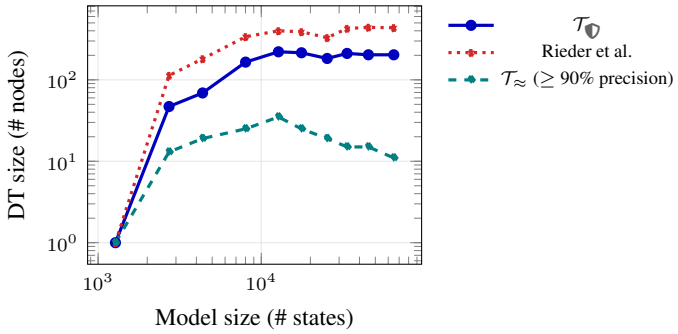

Figure 7: CARLA scaling: DT size vs. model size. The approximate DT is selected at $\geq$ 90% precision for each dataset size.



Figure 8: The CARLA scenario with a pedestrian crossing the road in front of the car.

of growing with the model like every exact representation does—4–19$\times$ smaller than our own exact tree, 9–40$\times$ smaller than L1+L2, and 7–30$\times$ smaller than L1.

B Additional Evaluation Results

This appendix contains the per-environment results behind Tables 1 and 2, including the Pac-Man variant with two ghosts. Figures 10 and 11 belong to RQ1 (performance of agents trained under approximate shields), Fig. 12 to RQ2 (agents trained under the exact shield and evaluated under a fine-tuned shield), and Figs. 13 and 14 to the ablation of the cost in RQ1 and RQ2. For the ablation, we compare RL performance against DT size since precision is measured under each variant's own cost and is therefore not comparable.

C Runtime

Table 3 reports the wall-clock time of shield synthesis and of the construction of the exact shield DT. The times are approximate as they were only measured once. Shield synthesis is model checking rather than DT construction and is not unique to our approach.

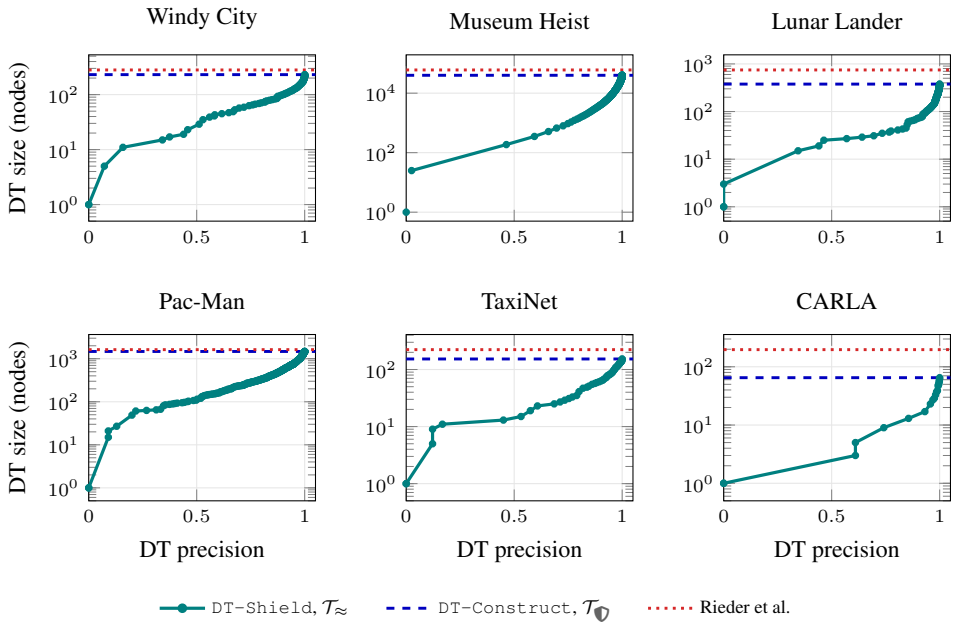

Figure 9: RQ1: size of $\mathcal{T}_{\approx}$ produced by `DT-Shield` at different weighted precisions for all environments. Dashed: exact shield DT; dotted: the baseline of Rieder et al. (2025).

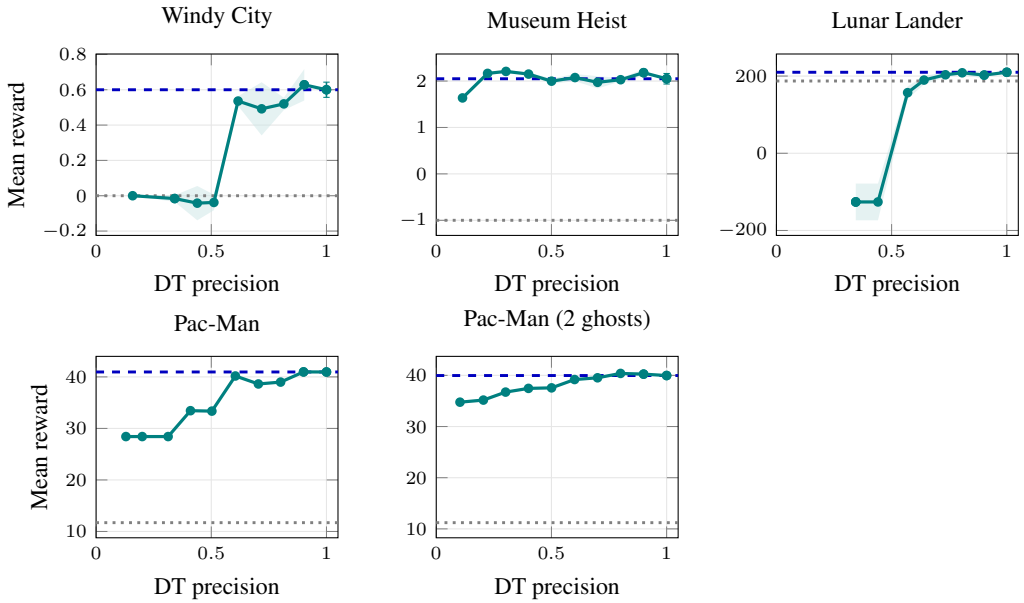

Figure 10: RQ1: mean reward of agents trained and evaluated under `DT-Shield`'s approximation, (checkpoints targeting 10%–90% precision; identical trees are shown once). The point at precision 1 is the agent under the exact shield. Each seed's mean over 100 episodes is one sample. Teal band and error bar: 95% confidence interval across 5 seeds; dotted: unshielded agent; dashed: agent under the exact shield.

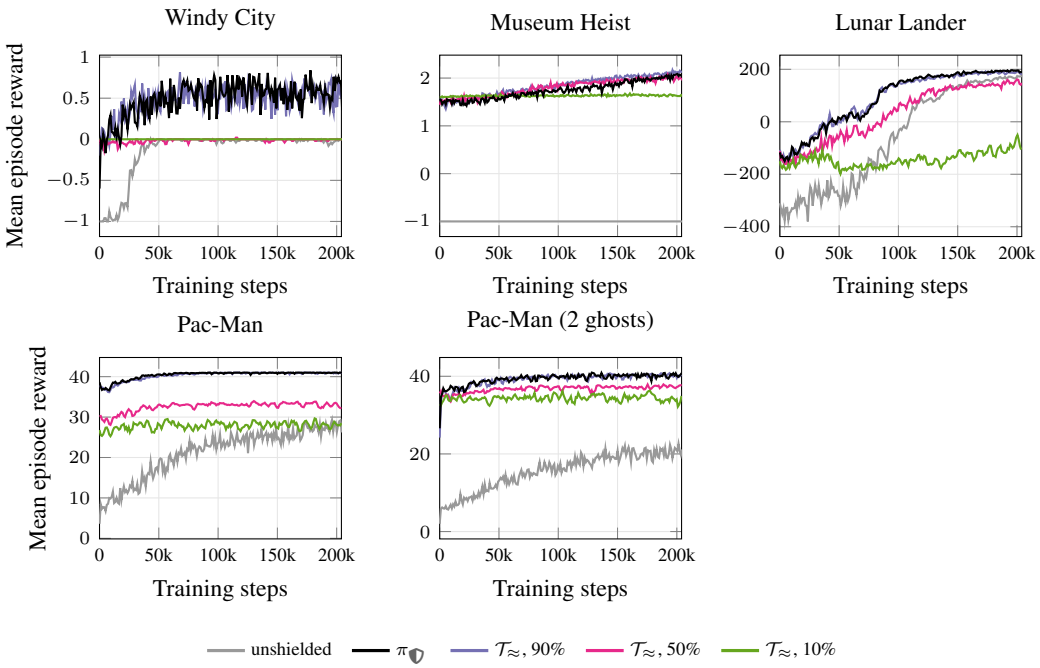

Figure 11: Training curves (mean episode reward over 5 seeds) of agents trained without a shield, under the exact shield $\pi_{\shield}$, and under `DT-Shield`'s unweighted approximations $\mathcal{T}_{\approx}$ with a precision of 90%, 50%, and 10%.

Table 3: Wall-clock time in seconds for shield synthesis, for constructing $\mathcal{T}_{\shield}$ with `DT-Construct` and a DT with a required precision of 1 with `DT-Shield`, and for the baseline of Rieder et al. (2025). Times are approximate.

			Exact shield DT		
Environment	$\lvert\mathcal{S}\rvert$	Synthesis	`DT-Construct`	`DT-Shield`	Rieder et al.
Windy City	1,000	5.03	0.07	0.05	1.84
Museum Heist	195,455	7.57	10.77	9.27	547.95
Lunar Lander	24,957	335.89	1.13	1.35	6.38
Pac-Man	8,035	0.21	0.44	0.34	11.22
Pac-Man (2 ghosts)	514,272	15.54	24.45	19.93	501.16
TaxiNet	587	0.02	0.04	0.04	1.39
CARLA	4,205	0.06	0.09	0.11	1.61

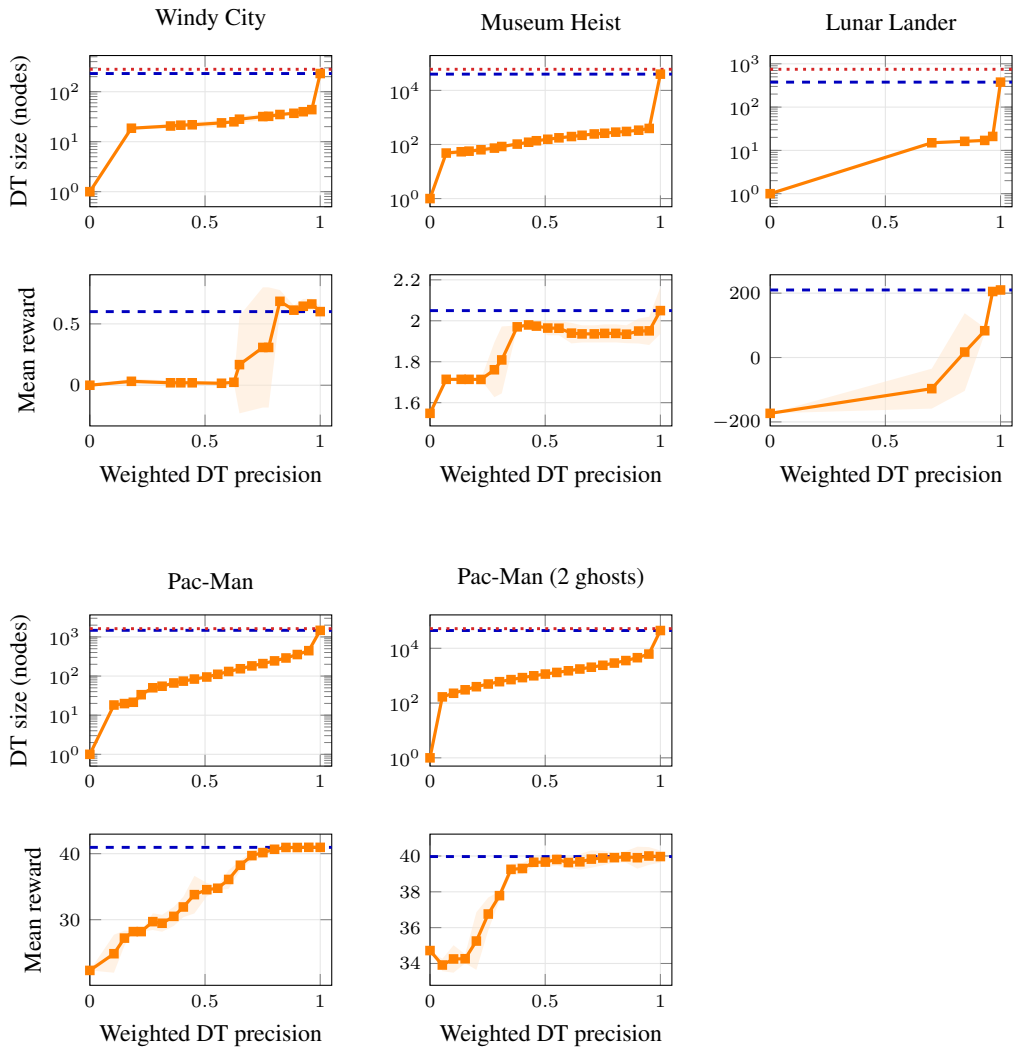

Figure 12: RQ2: DT size and mean reward of an agent trained under the exact shield and evaluated with the policy-weighted approximation $\mathcal{T}_{\text{🤖}}$, vs. the DT precision. Orange: mean with 95% confidence interval across 5 seeds (each seed's mean over 100 episodes); dashed: $\mathcal{T}_{\text{🛡}}$, dotted: the baseline of Rieder et al. (2025).

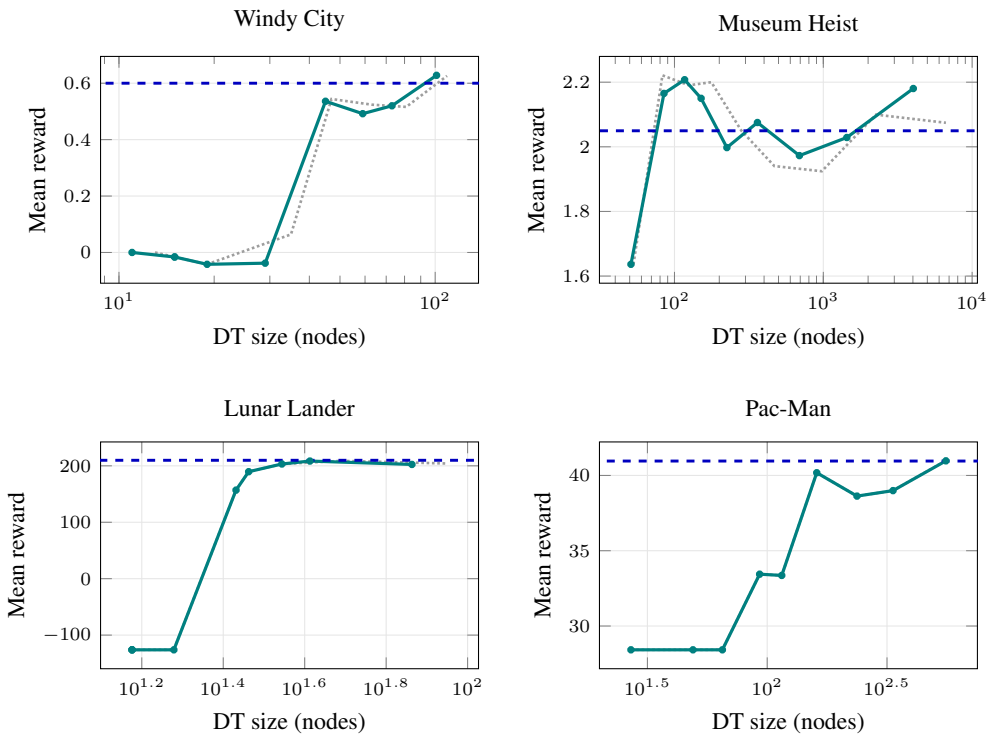

Figure 13: Ablation of the approximation cost in RQ1 (agents trained under the approximate shield): mean reward against DT size for the cost of `DT-Shield` (teal) and for a uniform cost (light gray, dotted). The dashed blue line is the reward under the exact shield. Precision is not comparable across the variants, as each is measured under its own cost.

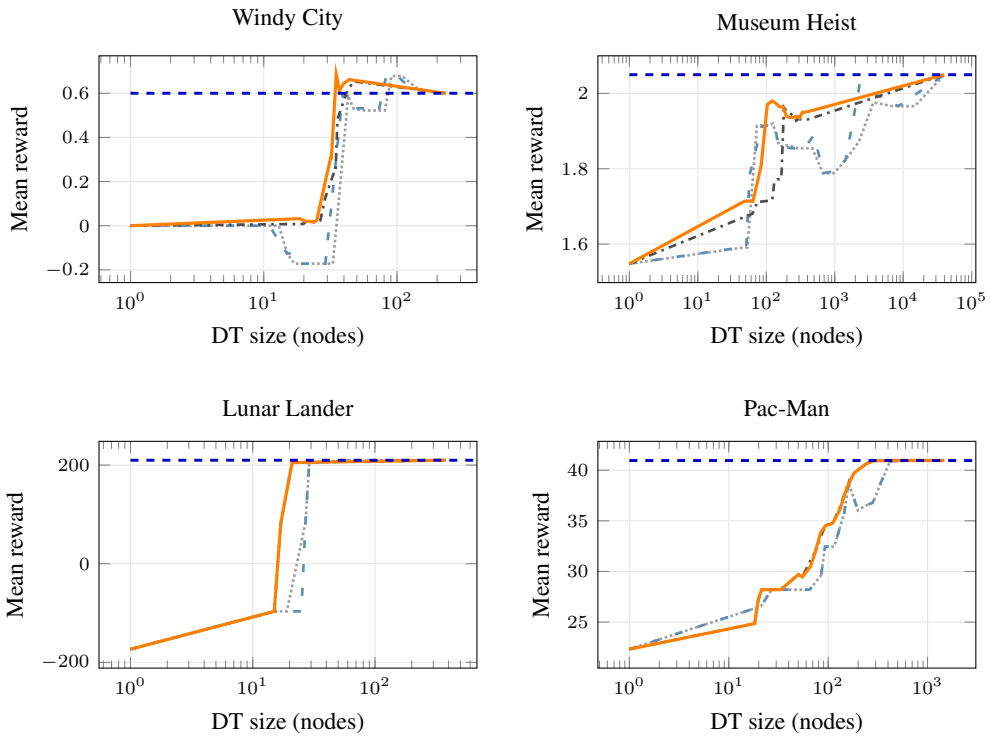

Figure 14: Ablation of the approximation cost in RQ2 (agents trained under the exact shield): mean reward against DT size for the cost of `DT-Shield`$_{\text{🤖}}$ (orange: policy weighting and safety margin), the weighting without c (dark gray, dash-dotted), cost c without the weighting (blue, dashed), and a uniform cost with neither (light gray, dotted). The dashed blue line is the reward under the exact shield. Precision is not comparable across the variants, as each is measured under its own cost.