COPA: CERTIFYING ROBUST POLICIES FOR OFFLINE REINFORCEMENT LEARNING AGAINST POISONING ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

As reinforcement learning (RL) has achieved near human-level performance in a variety of tasks, its robustness has raised great attention when applied to safety-critical domains such as autonomous driving. Recent studies have explored test-time attacks in RL and corresponding defenses, while the robustness of RL against training-time attacks remains largely unanswered. In this work, we focus on *certifying* the robustness of offline RL in the presence of poisoning attacks, where a subset of training trajectories could be *arbitrarily* manipulated. We propose the *first* certification framework, COPA to certify the number of poisoning trajectories that can be tolerated regarding different certification criteria. Given the complex structure of RL, we propose two certification criteria: *per-state action stability* and *cumulative reward bound*. To tighten the certification, we also propose different partition and aggregation protocols to train robust policies. We further prove that some of the proposed certification methods are theoretically tight and some are NP-Complete problems. We conduct a thorough evaluation of COPA on different games trained with different offline RL algorithms: (1) The proposed temporal aggregation in COPA significantly improves certified robustness; (2) Our certifications for both per-state action stability and cumulative reward bound are efficient and tight; (3) The certification for different training algorithms and games are different, implying their intrinsic robustness properties.

1 INTRODUCTION

Reinforcement learning (RL) has been widely applied to a range of applications, including robotics (Kober et al., 2013; Deisenroth et al., 2013; Polydoros & Nalpantidis, 2017) and autonomous vehicles (Shalev-Shwartz et al., 2016; Sallab et al., 2017). In particular, offline RL (Levine et al., 2020) is proposed to leverage previously observed data to train the policy without requiring expensive interaction with environments or online data collection, and enables the reuse of training data (Agarwal et al., 2020). However, offline RL also raises a great safety concern on poisoning attacks (Kiourti et al., 2020; Wang et al., 2020; 2021). A recent survey of industry reports that data poisoning is significantly more concerning than other threats (Kumar et al., 2020). For offline RL, the situation is even worse, and a recent theoretical study shows that the robust offline RL against poisoning attacks is a strictly harder problem than online RL (Zhang et al., 2021).

Although there are several empirical and certified defenses for classification tasks against poisoning attacks (Peri et al., 2020; Levine & Feizi, 2021; Weber et al., 2020), it is challenging and shown ineffective to directly apply them to RL given its complex structure (Kiourti et al., 2020). Thus, robust offline RL against poisoning attacks remains largely unexplored with no mention of robustness certification. In addition, though some theoretical analyses provide general robustness bounds for RL, they either assume a bounded distance between the learned and Bellman optimal policies or are limited to linear MDPs (Zhang et al., 2021). To the best of our knowledge, there is no robust RL method that is able to provide *practically computable certified robustness* against poisoning attacks. In this paper, we tackle this problem by proposing the first framework of Certifying robust policies for general offline RL against poisoning attacks (**COPA**).

Certification Criteria. One critical challenge in certifying robustness for offline RL is the certification criteria, since the prediction consistency is no longer the only goal as in classification tasks. We propose two criteria based on the properties of RL: per-state action stability and cumulative reward bound. The per-state action stability guarantees that at a specific time, the policy learned with COPA will predict the same action before and after attacks under certain conditions. This criterion is important for guaranteeing the safety of the policy at critical states, *e.g.*, stopping a vehicle when seeing pedestrians. For cumulative reward bound, a lower bound of the cumulative reward for the policy learned with COPA is guaranteed under certain poisoning conditions. This criterion directly guarantees the worst-case overall performance.

COPA Framework. COPA is composed of two components: policy partition and aggregation protocol and robustness certification method. We propose three policy partition aggregation protocols: PARL (Per-State Partition Aggregation), TPARL (Temporal Partition Aggregation), and DPARL (Dynamic Temporal Partition Aggregation), and propose certification methods for each of them corresponding to both proposed certification criteria. In addition, for *per-state action stability*, we prove that our certifications for PARL and TPARL are theoretically tight. For *cumulative reward bound*, we propose an adaptive tree search algorithm, where we will compute the possible action set for each state under certain poisoning conditions. For PARL, we propose a novel method to compute the precise action set. For TPARL and DPARL, we propose efficient algorithms to compute a superset of the possible action set which leads to sound certification. We further prove that for PARL our certification is theoretically tight, for TPARL the theoretically tight certification is NP-complete, and for DPARL it is open whether theoretically tight certification exists.

Technical Contributions. We take the first step towards certifying the robustness of offline RL against poisoning attacks, and we make contributions on both theoretical and practical fronts.

- We abstract and formulate the robustness certification for offline RL against poisoning attacks, and we propose two certification criteria: per-state action stability and cumulative reward bound.
- We propose the first framework COPA for certifying robustness of offline RL against poisoning attacks. COPA includes novel policy aggregation protocols and certification methods.
- We prove the tightness of the proposed certification methods for the aggregation protocol PARL. We also prove the computational hardness of the certification for TPARL.
- We conduct thorough experimental evaluation for COPA on different games with three offline RL algorithms, demonstrating the effectiveness of COPA, together with several interesting findings.

2 Certification Criteria of COPA

In this section, we propose two robustness certification criteria for offline RL against general poisoning attacks: per-state action stability and cumulative reward bound.

Offline RL. We model the RL environment by an episodic finite-horizon Markov decision process (MDP) $\mathcal{E} = (\mathcal{S}, \mathcal{A}, R, P, H, d_0)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of discrete actions, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the stochastic transition function with $\mathcal{P}(\cdot)$ defining the set of probability measures, H is the time horizon, and $d_0 \in \mathcal{P}(\mathcal{S})$ is the distribution of the initial state. At time step t, the RL agent is at state $s_t \in \mathcal{S}$. After choosing action $a_t \in \mathcal{A}$, the agent transitions to the next state $s_{t+1} \sim P(s_t, a_t)$ and receives reward $r_t = R(s_t, a_t)$. After all H time steps, the agent's cumulative reward $J = \sum_{t=0}^{H-1} r_t$. We denote a consecutive sequence of all states between time step l and r as $s_{l:r} := [s_l, s_{l+1}, \ldots, s_r]$.

Here we focus on offline RL, for which the threat of poisoning attacks is practical and more challenging to deal with (Zhang et al., 2021). Concretely, in offline RL, a training dataset $D = \{\tau_i\}_{i=1}^N$ consists of logged trajectories, where each trajectory $\tau = \{(s_j, r_j, a_j, s'_j)\}_{j=1}^l \in (\mathcal{S} \times \mathcal{A} \times \mathbb{R} \times \mathcal{S})^l$ consists of multiple tuples denoting the transitions (*i.e.*, starting from state s_j, taking the action a_j, receiving reward r_j, and transitioning to the next state s'_j).

Poisoning Attacks. Training dataset D can be poisoned in the following manner. For each trajectory $\tau \in D$, the adversary is allowed to replace it with an arbitrary trajectory $\widetilde{\tau}$, generating a manipulated dataset $\widetilde{D}$. We denote $D \ominus \widetilde{D} = (D \backslash \widetilde{D}) \bigcup (\widetilde{D} \backslash D)$ as the *symmetric difference* between two datasets D and $\widetilde{D}$. For instance, adding or removing one trajectory causes a symmetric difference of magnitude 1, while replacing one trajectory with a new one leads to a symmetric difference of magnitude 2. We refer to the size of the symmetric difference as the *poisoning size*.

Table 1: Overview of theoretical results in Section 3. "Certification" columns entail our certification theorems. "Analysis" columns entail the analyses of our certification bounds, where "tight" means our certification is theoretically tight, "NP-complete" means the tight certification problem is NP-complete, and "open" means the tight certification problem is still open. Theorems 8 and 13 and proposition 9 are in Appendices C and F.6.2.

Certification Criteria	Proposed Aggregation Protocol					
	PARL (π_{P}, Definition 7)		TPARL (π_{T}, Definition 8)		DPARL (π_{D}, Definition 3)	
	Certification	Analysis	Certification	Analysis	Certification	Analysis
Per-State Action	Theorem 8	tight (Proposition 9)	Theorem 1	tight (Proposition 2)	Theorem 3	open
Cumulative Reward	Theorem 4	tight (Theorem 13)	Theorem 5	NP-complete (Theorem 6)	Theorem 7	open

Certification Goal. To provide the robustness certification against poisoning attacks introduced above, we aim to certify the *test-time* performance of the trained policy in a clean environment. Specifically, in the *training phase*, the RL training algorithm and our aggregation protocol can be jointly modeled by $\mathcal{M} : \mathcal{D} \to (\mathcal{S}^\star \to \mathcal{A})$ which provides an aggregated policy, where $\mathcal{S}^\star$ denotes the set of all consecutive state sequences. Our **goal** is to provide robustness certification for the poisoned aggregated policy $\tilde{\pi} = \mathcal{M}(\widetilde{D})$, given bounded poisoning size (*i.e.*, $|D \ominus \widetilde{D}| \leq \overline{K}$).

Robustness Certification Criteria: Per-State Action Stability. We first aim to certify the robustness of the poisoned policy in terms of the stability of per-state action during test time.

Definition 1 (Robustness Certification for Per-State Action Stability)**.** Given a clean dataset D, we define the robustness certification for *per-state action stability* as that for *any* $\widetilde{D}$ satisfying $|D \ominus \widetilde{D}| \leq \overline{K}$, the action predictions of the poisoned and clean policies for the state (or state sequence) s are the same, *i.e.*, $\tilde{\pi} = \mathcal{M}(\widetilde{D}), \pi = \mathcal{M}(D), \tilde{\pi}(s) = \pi(s)$, under the the tolerable poisoning threshold $\overline{K}$. In an episode, we denote the tolerable poisoning threshold for the state at step t by $\overline{K}_t$.

The definition encodes the requirement that, for a particular state, any poisoned policy will always give the same action prediction as the clean one, as long as the poisoning size is within $\overline{K}$ ($\overline{K}$ computed in Section 3). In this definition, s could be either a state or a state sequence, since our aggregated policy (defined in Section 2) may aggregate multiple recent states to make an action choice.

Robustness Certification Criteria: Lower Bound of Cumulative Reward. We also aim to certify poisoned policy's overall performance in addition to the prediction at a particular state. Here we measure the overall performance by the cumulative reward $J(\pi)$ (formally defined in Appendix A).

Now we are ready to define the robustness certification for cumulative reward bound.

Definition 2 (Robustness Certification for Cumulative Reward Bound)**.** Robustness certification for cumulative reward bound is the *lower bound* of *cumulative reward* $\underline{J}_K$ such that $\underline{J}_K \leq J(\tilde{\pi})$ for any $\tilde{\pi} = \mathcal{M}(\widetilde{D})$ where $|D \ominus \widetilde{D}| \leq K$, *i.e.*, $\tilde{\pi}$ is trained on poisoned dataset $\widetilde{D}$ within poisoning size K.

3 Certification Process of COPA

In this section, we introduce our framework COPA, which is composed of *training protocols*, *aggregation protocols*, and *certification methods*. The training protocol combined with an offline RL training algorithm provides subpolicies. The aggregation protocol aggregates the subpolicies as an aggregated policy. The certification method certifies the robustness of the aggregated policy against poisoning attacks corresponding to different certification criteria provided in Section 2.

Overview of Theoretical Results. In Table 1, we present an overview of our theoretical results: For each proposed aggregation protocol and certification criteria, we provide the corresponding certification method and core theorems, and we also provide the tightness analysis for each certification.

3.1 Partition-Based Training Protocol

COPA's training protocol contains two stages: partitioning and training. We denote D as the entire offline RL training dataset. We abstract an offline RL training algorithm (*e.g.*, DQN) by $\mathcal{M}_0 : 2^D \to \Pi$, where 2^D is the power set of D, and $\Pi = \{\pi : \mathcal{S} \to \mathcal{A}\}$ is the set of trained policies. Each trained policy in Π is a function mapping a given state to the predicted action.

Partitioning Stage. In this stage, we separate the training dataset D into u partitions $\{D_i\}_{i=0}^{u-1}$ that satisfy $\bigcup_{i=0}^{u-1} D_i = D$ and $\forall i \neq j, D_i \cap D_j = \emptyset$. Concretely, when performing partitioning, for each trajectory $\tau \in D$, we *deterministically* assign it to one unique partition. The assignment

is only dependent on the trajectory τ itself, and not impacted by any modification to other parts of the training set. One design choice of such a deterministic assignment is using a deterministic hash function h to compute the assignment, *i.e.*, $D_i = \{\tau \in D \mid h(\tau) \equiv i \pmod u\}, \forall i \in [u]$.

Training Stage. In this stage, for each training data partition D_i, we independently apply the chosen RL algorithm $\mathcal{M}_0$ to train a policy $\pi_i = \mathcal{M}_0(D_i)$. Hereinafter, we call these trained polices as *subpolicies* to distinguish from the aggregated policies. Concretely, let $[u] := \{0, 1, \ldots, u-1\}$. For these subpolicies, the *policy indicator* $\mathbf{1}_{i,a} : \mathcal{S} \to \{0, 1\}$ is defined by $\mathbf{1}_{i,a}(s) := \mathbf{1}[\pi_i(s) = a]$, indicating whether subpolicy π_i chooses action a given current state s. The *aggregated action count* $n_a : \mathcal{S} \to \mathbb{N}_{\geq 0}$ is the number of votes across all the subpolicies for action a given current state s: $n_a(s) := |\{i \mid \pi_i(s) = a, i \in [u]\}| = \sum_{i=0}^{u-1} \mathbf{1}_{i,a}(s)$. Specifically, we denote $n_a(s_{l:r})$ for $\sum_{j=l}^{r} n_a(s_j)$, *i.e.*, the sum of votes for all states between time step l and r. A detailed algorithm description of the training protocol is in Appendix E.

Now we are ready to introduce the proposed aggregation protocols in COPA (PARL, TPARL, DPARL) that generate *aggregated policies* based on subpolicies, and corresponding certification.

3.2 Aggregation Protocols: PARL, TPARL, DPARL

With u learned subpolicies $\{\pi_i\}_{i=0}^{u-1}$, we propose three different aggregation protocols in COPA to form three types of aggregated policies for each certification criteria: PARL, TPARL, and DPARL.

Per-State Partition Aggregation (PARL). Inspired by aggregation in classification (Levine & Feizi, 2021), PARL aggregates subpolicies by choosing actions with the highest votes. We denote the PARL aggregated policy by $\pi_{\mathrm{P}} : \mathcal{S} \to \mathcal{A}$. When there are multiple highest voting actions, we break ties deterministically by returning the "smaller" ($<$) action, which can be defined by numerical order, lexicographical order, etc. Throughout the paper, we assume $\arg\max$ over $\mathcal{A}$ always uses $<$ operator to break ties. The formal definition of the protocol is in Appendix A.

The intuition behind PARL is that the poisoning attack within size K can change at most K subpolicies. Therefore, as long as the margin between the votes for top and runner-up actions is larger than $2K$ for the given state, after poisoning, we can guarantee that the aggregated PARL policy will not change its action choice. We will formally state the robustness guarantee in Section 3.3.

Temporal Partition Aggregation (TPARL). In the sequential decision making process of RL, it is likely that certain important states are much more vulnerable to poisoning attacks, which we refer to as *bottleneck states*. Therefore, the attacker may just change the action predictions for these bottleneck states to deteriorate the overall performance, say, the cumulative reward. For example, in Pong game, we may lose the round when choosing an immediate bad action when the ball is closely approaching the paddle. Thus, to improve the overall certified robustness, we need to focus on improving the tolerable poisoning threshold for these bottleneck states. Given such intuition and goal, we propose Temporal Partition Aggregation (TPARL) and the aggregated policy is denoted as π_{T}, which is formally defined in Definition 8 in Appendix A.

TPARL is based on two insights: (1) Bottleneck states have lower tolerable poisoning threshold, which is because the vote margin between the top and runner-up actions is smaller at such state; (2) Some RL tasks satisfy temporal continuity (Legenstein et al., 2010; Veerapaneni et al., 2020), indicating that good action choices are usually similar across states of adjacent time steps, *i.e.*, adjacent states. Hence, we leverage the subpolicies' votes from adjacent states to enlarge the vote margin, and thus increase the tolerable poisoning threshold. To this end, in TPARL, we predetermine a *window size* W, and choose the action with the highest votes across recent W states.

Dynamic Temporal Partition Aggregation (DPARL). The TPARL uses a fixed window size W across all states. Since the specification of the window size W requires certain prior knowledge, plus that the same fixed window size W may not be suitable for all states, it is preferable to perform *dynamic temporal aggregation* by using a flexible window size. Therefore, we propose Dynamic Temporal Partition Aggregation (DPARL), which dynamically selects the window size W towards maximizing the tolerable poisoning threshold *per step*. Intuitively, DPARL selects the window size W such that the average vote margin over selected states is maximized. To guarantee that only recent states are chosen, we further constrain the maximum window size ($W_{\max}$).

Definition 3 (Dynamic Temporal Partition Aggregation)**.** Given subpolicies $\{\pi_i\}_{i=0}^{u-1}$ and maximum window size $W_{\max}$, at time step t, the Dynamic Temporal Partition Aggregation (DPARL) defines

an aggregated policy $\pi_{\mathrm{D}} : \mathcal{S}^{\min\{t+1, W_{\max}\}} \to \mathcal{A}$ such that

$$\pi_{\mathrm{D}}(s_{\max\{t-W_{\max}+1,0\}:t}) := \arg\max_{a\in\mathcal{A}} n_a(s_{t-W'+1:t}), \text{ where } W' = \arg\max_{1\le W\le \min\{W_{\max},t+1\}} \Delta_t^W. \quad (1)$$

In the above equation, n_a is defined in Section 2 and Δ_t^W is given by

$$\Delta_t^W := \frac{1}{W}\left(n_{a_1}(s_{t-W+1:t}) - n_{a_2}(s_{t-W+1:t})\right), \quad \text{where} \quad a_1 = \arg\max_{a\in\mathcal{A}} n_a(s_{t-W+1:t}), a_2 = \arg\max_{a\in\mathcal{A}, a\neq a_1} n_a(s_{t-W+1:t}). \quad (2)$$

In the above definition, Δ_t^W encodes the average vote margin between top action a_1 and runner-up action a_2 if choosing window size W. Thus, W' locates the window size with maximum average vote margin, and its corresponding action is selected. Again, we use the mechanism described in PARL to break ties. Robustness certification methods for DPARL are in Sections 3.3 and 3.4.

In Appendix B, we use a concrete example to illustrate bottleneck states and non-bottleneck states, and demonstrate how different aggregation protocols induce different tolerable poisoning thresholds.

3.3 Certification of Per-State Action Stability

In this section, we present our robustness certification theorems and methods for *per-state action*. For each of the aggregation protocols (PARL, TPARL, and DPARL), at each time step t, we will compute a valid *tolerable poisoning threshold* $\overline{K}$ as defined in Definition 1, such that the chosen action at step t does not change as long as the poisoning size $K \le \overline{K}$.

Certification for PARL. Due to the space limit, we defer the robustness certification method for PARL to Appendix C. The certification method is based on Theorem 8. We further show the theoretical tightness of the certification in Proposition 9. All the theorem statements are in Appendix C.

Certification for TPARL. We certify the robustness of TPARL following Theorem 1.

Theorem 1. *Let D be the clean training dataset; let $\pi_i = \mathcal{M}_0(D_i), 0 \le i \le u-1$ be the learned subpolicies according to Section 3.1 from which we define n_a (Section 2); and let π_{T} be the Temporal Partition Aggregation policy: $\pi_{\mathrm{T}} = \mathcal{M}(D)$ where $\mathcal{M}$ abstracts the whole training-aggregation process. $\widetilde{D}$ is a poisoned dataset and $\widetilde{\pi_{\mathrm{T}}}$ is the poisoned policy: $\widetilde{\pi_{\mathrm{T}}} = \mathcal{M}(\widetilde{D})$.*

For a given state s_t encountered at time step t during test time, let $a := \pi_{\mathrm{T}}(s_{\max\{t-W+1,0\}:t})$, then at time step t the tolerable poisoning threshold (see Definition 1)

$$\overline{K}_t = \min_{a'\neq a, a'\in\mathcal{A}} \max\left\{ p \,\middle|\, \sum_{i=1}^{p} h_{a,a'}^{(i)} \le \delta_{a,a'} \right\} \quad (3)$$

where $\{h_{a,a'}^{(i)}\}_{i=1}^{u}$ is a nonincreasing permutation of

$$\left\{ \sum_{j=0}^{\min\{W-1,t\}} \mathbf{1}_{i,a}(s_{t-j}) + \min\{W, t+1\} - \sum_{j=0}^{\min\{W-1,t\}} \mathbf{1}_{i,a'}(s_{t-j}) \right\}_{i=0}^{u-1} =: \{h_{i,a,a'}\}_{i=0}^{u-1}, \quad (4)$$

and $\delta_{a,a'} := n_a(s_{\max\{t-W+1,0\}:t}) - (n_{a'}(s_{\max\{t-W+1,0\}:t}) + \mathbf{1}[a' < a])$. Here, $\mathbf{1}_{i,a}(s) = \mathbf{1}[\pi_i(s) = a]$ (Section 2), and W is the window size.

Proof sketch. Let $t_s = \max\{t-W+1, 0\}$. We consider the condition of successful poisoning, which is to enable a' to satisfy $\widetilde{n}_{a'}(s_{t_s:t}) + \mathbf{1}[a' < a] > \widetilde{n}_a(s_{t_s:t})$ after the poisoning. We figure out the optimal attack strategy towards satisfying this constraint and negation of this optimal attack brings the theorem statement. We defer the detailed proof to Appendix F.2. □

Remark. The theorem provides a valid per-state action certification for TPARL. The detailed algorithm is in Algorithm 3 (Appendix E.2). The certification time complexity per state is $O(|\mathcal{A}|u(W + \log u))$ and can be further optimized to $O(|\mathcal{A}|u \log u)$ with proper prefix sum caching across time steps. We prove the certification for TPARL is theoretically tight in Proposition 2 with proofs in Appendix F.3. We also prove that directly extending Theorem 8 for TPARL (Corollary 10) is loose in Appendix F.4.

Proposition 2. *Under the same condition as Theorem 1, for any time step t, there exists an RL learning algorithm $\mathcal{M}_0$, and a poisoned dataset $\widetilde{D}$, such that $|D \ominus \widetilde{D}| = \overline{K}_t + 1$, and $\widetilde{\pi_{\mathrm{T}}}(s_{\max\{t-W+1,0\}:t}) \neq \pi_{\mathrm{T}}(s_{\max\{t-W+1,0\}:t})$.*

Certification for DPARL. Theorem 3 provides certification for DPARL.

Theorem 3. *Let D be the clean training dataset; let $\pi_i = \mathcal{M}_0(D_i), 0 \le i \le u-1$ be the learned subpolicies according to Section 3.1 from which we define n_a (see Section 2); and let π_{D} be the Dynamic Temporal Partition Aggregation: $\pi_{\mathrm{D}} = \mathcal{M}(D)$ where $\mathcal{M}$ abstracts the whole training-aggregation process. $\widetilde{D}$ is a poisoned dataset and $\widetilde{\pi_{\mathrm{D}}}$ is the poisoned policy: $\widetilde{\pi_{\mathrm{D}}} = \mathcal{M}(\widetilde{D})$.*

For a given state s_t encountered at time step t during test time, let $a := \pi_{\mathrm{D}}(s_{\max\{t-W_{\max}+1,0\}:t})$ and W' be the chosen time window (according to Equation (1)), then tolerable poisoning threshold

$$\overline{K}_t^D = \min\left\{\overline{K}_t, \min_{1\le W^*\le \min\{W_{\max},t+1\}, W^*\neq W', a'\neq a, a''\neq a} L_{a',a''}^{W^*,W'}\right\} \tag{5}$$

where $\overline{K}_t$ is defined by Equation (3) with W as W' and $L_{a,a''}^{W^*,W'}$ defined by the below Definition 4.

Definition 4 (L in Theorem 3)**.** Under the same condition as Theorem 3, for given W^*, W', a, a', a'', we let $a^{\#} := \arg\max_{a_0\neq a', a_0\in\mathcal{A}} n_{a_0}(s_{t-W^*+1:t})$, then

$$L_{a',a''}^{W^*,W'} := \max\left\{p \,\middle|\, \sum_{i=1}^{p} g^{(i)} + W'(n_{a'}^{W^*} - n_{a^{\#}}^{W^*}) - W^*(n_{a}^{W'} - n_{a''}^{W'}) - \mathbf{1}[a' > a] < 0\right\} \tag{6}$$

where n_a^w is a shorthand of $n_a(s_{t-w+1:t})$ and $\{g^{(i)}\}_{i=1}^{u}$ is a nonincreasing permutation of $\{g_i\}_{i=0}^{u-1}$. Each g_i is defined by

$$g_i := \sum_{w=0}^{\max\{W^*,W'\}} \max_{a_0\in\mathcal{A}} \sigma^w(a_0) - \sigma^w(\pi_i(s_{t-w})), \text{ where} \tag{7}$$

$$\sigma^w(a_0) := W'\mathbf{1}[a_0 = a', w \le W^*] - W'\mathbf{1}[a_0 = a^{\#}, w \le W^*] - W^*\mathbf{1}[a_0 = a, w \le W'] + W^*\mathbf{1}[a_0 = a'', w \le W'].$$

Proof sketch. A successful poisoning attack should change the chosen action from a to an another a'. If after attack, the chosen window size is still W', the poisoning size should at least larger than $\overline{K}_t$ according to Theorem 1. If the chosen window size is not W', we find out that the poisoning size is at least $\min_{a''\neq a} L_{a',a''}^{W^*,W'} + 1$ from a greedy-based analysis. Formal proof is in Appendix F.5. □

Remark. The theorem provides a valid per-state action certification for DPARL policy. The detailed algorithm is in Algorithm 4 (Appendix E.2). The certification time complexity per state is $O(W_{\max}^2|\mathcal{A}|^2u + W_{\max}|\mathcal{A}|^2u\log u)$, which in practice adds similar overhead compared with TPARL certification (see Appendix H.3). Unlike certification for PARL and TPARL, the certification given by Theorem 3 is not theoretically tight. An interesting future work would be providing a tighter per-state action certification for DPARL.

3.4 Certification of Cumulative Reward Bound

In this section, we present our robustness certification methods for *cumulative reward bound*. We assume that the RL environment is deterministic throughout the cumulative reward certification for convenience, *i.e.*, the transition function is $P : \mathcal{S}\times\mathcal{A} \to \mathcal{S}$ and the initial state is a fixed $s_0 \in \mathcal{S}$. The certification goal, as listed in Definition 6, is to obtain a lower bound of cumulative reward under poisoning attacks, as long as the poisoning size is within K. The cumulative reward certification is based on a novel adaptive tree search algorithm COPA-Search inspired from Wu et al. (2021). We tailor the algorithm to certify against poisoning attacks instead of adversarial state perturbations as in Wu et al. (2021). A detailed discussion and complexity analysis of the algorithm is in Appendix E.3.

COPA-Search Algorithm Description. The pseudocode is in Algorithm 5 (Appendix E.3). The method starts from the base case: when the poisoning threshold $K_{\mathrm{cur}} = 0$, the lower bound of cumulative reward $\underline{J}_{K_{\mathrm{cur}}}$ is exactly the reward without poisoning. The method then gradually increases the poisoning threshold K_{cur}, by finding the immediate larger $K' > K_{\mathrm{cur}}$ that can expand the possible action set along the trajectory. With the increase of $K_{\mathrm{cur}} \leftarrow K'$, the attack may cause the poisoned policy $\tilde{\pi}$ to take different actions at some states, thus resulting in new trajectories. We need to figure out *a set of all possible actions* to exhaustively traverse all possible trajectories. We will introduce theorems to compute this set of possible actions. With this set, the method effectively explores these new trajectories by formulating them as expanded branches of a trajectory tree. Once all new trajectories are explored, the method examines all leaf nodes of the tree and figures out the minimum reward among them, which is the new lower bound of cumulative reward $\underline{J}_{K'}$ under new poisoning size K'. We then repeat this process of increasing poisoning size from K' and expanding with new trajectories until we reach a predefined threshold for poisoning size K.

Definition 5 (Possible Action Set)**.** Given previous states $s_{0:t}$, the subpolicies $\{\pi_i\}_{i=0}^{u-1}$, the aggregation protocol (PARL, TPARL, or DPARL), and the poisoning size K, the *possible action set* A

at step t is a subset of action space: $A \subseteq \mathcal{A}$, such that for any poisoned policy $\widetilde{\pi}$, as long as the poisoning size is within K, the chosen action at step t will always be in A, *i.e.*, $a_t = \widetilde{\pi}(s_{0:t}) \in A$.

Possible Action Set for PARL. The following theorem gives the possible action set for PARL.

Theorem 4 (Tight PARL Action Set)**.** *Under the condition of Definition 5, suppose the aggregation protocol is PARL as defined in Definition 7, then the* possible action set *at step t*

$$A^T(K) = \left\{ a \in \mathcal{A} \,\middle|\, \sum_{a' \in \mathcal{A}} \max\{n_{a'}(s_t) - n_a(s_t) - K + \mathbf{1}[a' < a], 0\} \leq K \right\}. \tag{8}$$

We defer the proof to Appendix F.6.1. Furthermore, in Appendix F.6.2 we show that: 1) The theorem gives theoretically tight possible action set; 2) In contrast, directly extending PARL's per-state certification gives loose certification.

Possible Action Set for TPARL. The following theorem gives the possible action set for TPARL.

Theorem 5. *Under the condition of Definition 5, suppose the aggregation protocol is TPARL as defined in Definition 8, then the* possible action set *at step t*

$$A(K) = \left\{ a \in \mathcal{A} \,\middle|\, \sum_{i=1}^{K} h_{a',a}^{(i)} > \delta_{a',a}, \forall a' \neq a \right\}, \tag{9}$$

where $h_{a',a}^{(i)}$ and $\delta_{a',a}$ follow the definition in Theorem 1.

We defer the proof to Appendix F.7. The possible action set here is no longer theoretically tight. Indeed, the problem of computing a possible action set with minimum cardinality for TPARL is NP-complete as we shown in the following theorem (proved in Appendix F.8), where we reduce computing theoretically tight possible action set to the set cover problem (Karp, 1972). This result can be viewed as the hardness of targeted attack. In other words, the optimal *untargeted* attack on TPARL can be found in polynomial time, while the optimal *targeted* attack on TPARL is NP-complete, which indicates the robustness property of proposed TPARL.

Theorem 6. *Under the condition of Definition 5, suppose the aggregation protocol is TPARL as defined in Definition 8, then computing a possible action set $A(K)$ such that any possible action set S satisfies $|A(K)| \leq |S|$ is* NP*-complete.*

Possible Action Set for DPARL. The following theorem gives the possible action set for DPARL.

Theorem 7. *Under the condition of Definition 5, suppose the aggregation protocol is TPARL as defined in Definition 3, then the* possible action set *at step t*

$$\begin{aligned} A(K) = \{a_t\} &\cup \left\{ a' \in \mathcal{A} \,\middle|\, \min_{1 \leq W^* \leq \min\{W_{\max}, t+1\}, W^* \neq W', a'' \neq a_t} L_{a',a''}^{W^*,W'} \leq K \right\} \\ &\cup \left\{ a \in \mathcal{A} \,\middle|\, \sum_{i=1}^{K} h_{a',a}^{(i)} > \delta_{a',a}, \forall a' \neq a \right\} \end{aligned} \tag{10}$$

where $a_t = \pi_{\mathrm{D}}(s_{\max\{t-W_{\max}+1,0\}:t})$ is the clean policy's chosen action, W' is defined by Equation (1)*, $L_{a',a''}^{W^*,W'}$ is defined by Definition 4 with a being replaced by a_t, and $h_{a',a}^{(i)}, \delta_{a',a}$ is defined in Theorem 1 with W replaced by W'.*

We prove the theorem in Appendix F.8. As summarized in Table 1, we further prove the tightness or hardness of certification for PARL and TPARL, while for DPARL it is an interesting open problem on whether theoretical tight certification is possible in polynomial time.

4 EXPERIMENTS

In this section, we present the evaluation for our COPA framework, specifically, the aggregation protocols (Section 3.2) and the certification methods under different certification criteria (Section 3.3 and 3.4). We defer the description of the offline RL algorithms (DQN (Mnih et al., 2013), QR-DQN (Dabney et al., 2018), and C51 (Bellemare et al., 2017)) used for training the subpolicies to Appendix G.1, and the concrete experimental procedures to Appendix G.2. As a summary, we obtain similar conclusions for both per-state action certification and cumulative reward certification: 1) QR-DQN and C51 are more certifiably robust than DQN; 2) temporal aggregation (PARL and TPARL) achieves higher certification for games explicitly satisfying temporal continuity, *e.g.*, Freeway; 3) larger partition number ($u = 50$) improves the certified robustness; 4) Freeway is a more stable and robust game than Breakout. More interesting discussions are deferred to Appendix H.

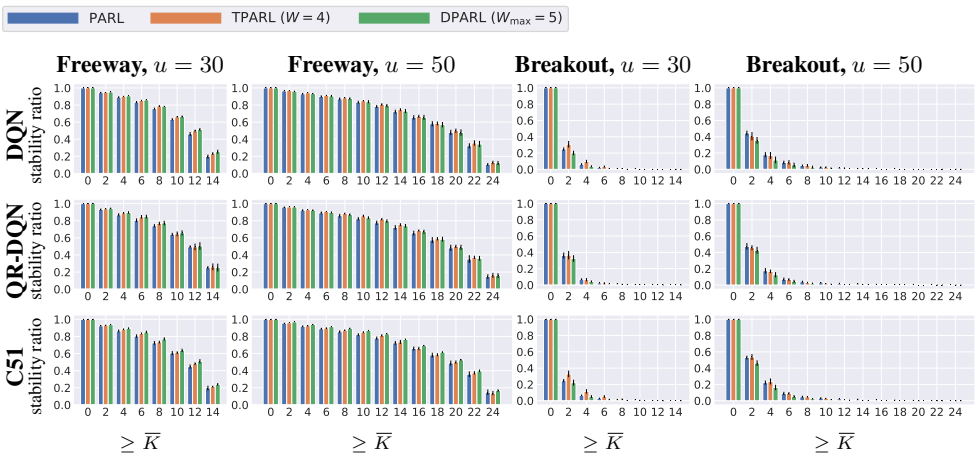


Figure 1: **Robustness certification for per-state action stability.** We plot the cumulative histogram of the *tolerable poisoning size* $\overline{K}$ for all time steps. We provide the certification for different aggregation protocols (PARL, TPARL, DPARL) on two games (Freeway and Breakout) and two #partitions ($u \in \{30, 50\}$). The results are averaged over 20 runs with the vertical bar on top denoting the standard deviation.

4.1 Evaluation of Robustness Certification for Per-State Action Stability

We provide the robustness certification for per-state action stability based on Section 3.3.

Experimental Setup and Metrics. We evaluate the aggregated policies π_{P}, π_{T}, and π_{D} following Section 3.2. Basically, in each run, we run one trajectory (of maximum length H) using the derived policy, and compute $\overline{K}_t$ at each time step t. Given $\{\overline{K}_t\}_{t=0}^{H-1}$, we obtain a cumulative histogram—for each threshold $\overline{K}$, we count the time steps that achieve a threshold no smaller than it and then normalize, *i.e.*, $\sum_{t=0}^{H-1} \mathbf{1}[\overline{K}_t \geq \overline{K}]/H$. We call this quantity *stability ratio* since it reflects the per-state action stability w.r.t. given poisoning thresholds. We also compute an *average tolerable poisoning thresholds* for a trajectory, defined as $\sum_{t=0}^{H-1} \overline{K}_t/H$. More details are deferred to Appendix G.2.

Evaluation Results. We present the comparison of per-state action certification for different RL methods and certification methods in Figure 1. We plot partial poisoning thresholds on the x-axes here, and omit the full set of results in Appendix H.6, where we also report the average tolerable poisoning thresholds. In the Appendix, we additionally report benign empirical reward in H.1 and compare with standard training in H.2. We include more analytical statistics for DPARL in H.3, and report the maximum tolerable poisoning threshold vs. the total number of trajectories in H.4.

The cumulative histograms in Figure 1 can be compared in different levels. Basically, we compare the *stability ratio* at each tolerable poisoning thresholds $\overline{K}$—*higher ratio at larger poisoning size indicates stronger certified robustness*. On the **RL algorithm** level, QR-DQN and C51 consistently outperform the baseline DQN algorithm. On the **aggregation protocol** level, we observe different behaviors on different games. On Freeway, methods with temporal aggregation (TPARL and DPARL) achieve higher robustness, and DPARL achieves the highest certified robustness in most cases; while on Breakout, the single-step aggregation PARL is oftentimes better. This difference is due to the different properties of games. Our temporal aggregation is developed based on the assumption of consistent action selection in adjacent time steps. This assumption is true in Freeway while violated in Breakout. A more detailed explanation of different game properties is omitted to Appendix H.5. On the **partition number** level, a larger partition number ($u = 50$) generally allows larger tolerable poisoning thresholds. Finally, on the **game** level, Freeway achieves much higher certified robustness for per-state action stability than Breakout, implying that Freeway is a game that accommodates more stable and robust policies than Breakout.

4.2 Evaluation of Robustness Certification for Cumulative Reward Bound

We provide the robustness certification for cumulative reward bound according to Section 3.4.

Experimental Setup and Metrics. We evaluate the aggregated policies π_{P}, π_{T}, and π_{D} following Theorem 4, Theorem 5 and Theorem 7. We compute the lower bounds of the cumulative reward $\underline{J}_K$ w.r.t. the poisoning size K using the COPA-Search algorithm introduced in Section 3.4. We adopt the trajectory length $H = 400$ for evaluating Freeway and $H = 75$ for Breakout.

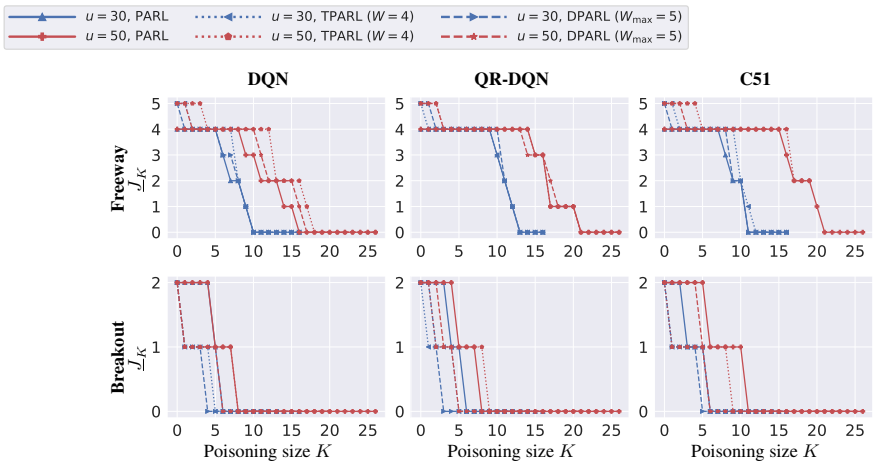


Figure 2: **Robustness certification for cumulative reward.** We plot the *lower bound of cumulative reward bound* $\underline{J}_K$ w.r.t. *poisoning size* K under three aggregation protocols (PARL, TPARL ($W = 4$), DPARL ($W_{\max} = 5$)) with two #partitions ($u \in \{30, 50\}$). Each column corresponds to one RL algorithm.

Evaluation Results. We present the comparison of reward certification for different RL algorithms and certification methods in Figure 2. Essentially, at each poisoning size K, we compare the lower bound of cumulative reward achieved by different RL algorithms and certification methods—*higher value of the lower bound implies stronger certified robustness*. On the **RL algorithm** level, similar to the conclusion in Section 3.4, QR-DQN and C51 invariably outperform the baseline DQN algorithm. On the **aggregation protocol** level, on Freeway, methods with temporal aggregation consistently surpass the single-step aggregation PARL, while PARL achieves better performance on Breakout, which is due to the similar analysis in Section 4.2. In addition, we note that DPARL is sometimes not as robust as TPARL. We hypothesize two reasons: 1) the dynamic mechanism is more susceptible to the attack, *e.g.*, the selected optimal window size is prone to be manipulated; 2) the lower bound is looser for DPARL given the difficulty of computing the possible action set in DPARL as discussed in Theorem 7. On the **partition number** level, a larger partition number ($u = 50$) demonstrates higher robustness. On the **game** level, Freeway can tolerate a larger poisoning size than Breakout. More results and discussions are omitted to Appendix H.7.

5 Related Work

Poisoning attacks (Nelson et al., 2008; Diakonikolas et al., 2016) are critical threats in machine learning, which are claimed to be more concerning than other threats (Kumar et al., 2020). Poisoning attacks widely exist in classification (Schwarzschild et al., 2021), and both empirical defenses (Liu et al., 2018; Chacon et al., 2019; Peri et al., 2020; Steinhardt et al., 2017) and certified defenses (Weber et al., 2020; Levine & Feizi, 2021) have been proposed. Robust RL against reward poisoning is studied in Banihashem et al. (2021), but robust RL against general poisoning is less explored.

From the attack side, Kiourti et al. (2020) show that effective poisoning attacks, in particular backdoor attacks, exist in RL. A recent work theoretically and empirically validates the existence of reward poisoning in online RL (Zhang et al., 2020b). Furthermore, Zhang et al. (2020a) theoretically prove that the offline RL is more difficult to be robustified against poisoning than online RL considering linear MDP. From the defense side, Zhang et al. (2020a) propose robust variants of the Least-Square Value Iteration algorithm that provides probabilistic robustness guarantees under linear MDP assumption. In contrast, we aim to provide the first certified robustness for general offline RL algorithms against poisoning attacks. We discuss broader related work in Appendix I.

6 Conclusions

In this paper, we proposed **COPA**, the first framework for certifying robust policies for offline RL against poisoning attacks. COPA proposes three different policy aggregation protocols: PARL, TPARL, and DPARL. For each aggregation protocol, COPA can provide a sound certification for both per-state action stability and cumulative reward bound. Our experimental evaluation on different games and different offline RL training algorithms shows the effectiveness of our robustness certification in a wide range of scenarios.

Ethics Statement. In this paper, we prove the first framework for certifying robust policies for offline RL against poisoning attacks. On the one hand, such framework provides rigorous guarantees in practice RL tasks and thus significantly alleviates the security vulnerabilities of offline RL algorithms against training-phase poisoning attacks. The evaluation of different RL algorithms also provides a better understanding about the different degrees of security vulnerabilities across different RL algorithms. On the other hand, the robustness guarantee provided by our framework only holds under specific conditions of the attack. Specifically, we require the attack to change only a bounded number of training trajectories. Therefore, users should be aware of such limitations of our framework, and should not blindly trust the robustness guarantee when the attack can change a large number of training instances or modify the training algorithm itself. As a result, we encourage researchers to understand the potential risks, and evaluate whether our constraints on the attack align with their usage scenarios when applying our COPA to real-world applications. We do not expect any ethics issues raised by our work.

Reproducibility Statement. All theorem statements are substantiated with rigorous proofs in Appendix F. In Appendix E, we list the pseudocode for all key algorithms. Our experimental evaluation is conducted with publicly available OpenAI Gym toolkit (Brockman et al., 2016). We introduce all experimental details in both Section 4 and Appendix G. Specifically, we build the code upon the open-source code base of Agarwal et al. (2020), and we upload the source code as the supplementary material for reproducibility purpose.

REFERENCES

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 2020.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Kiarash Banihashem, Adish Singla, and Goran Radanovic. Defense against reward poisoning attacks in reinforcement learning. *arXiv preprint arXiv:2102.05776*, 2021.

Vahid Behzadan and Arslan Munir. Whatever does not kill deep reinforcement learning, makes it stronger. *arXiv preprint arXiv:1712.09344*, 2017.

Vahid Behzadan and Arslan Munir. Mitigation of policy manipulation attacks on deep q-networks with parameter-space noise. In *International Conference on Computer Safety, Reliability, and Security*, pp. 406–417. Springer, 2018.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458. PMLR, 2017.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Henry Chacon, Samuel Silva, and Paul Rad. Deep learning poison data attack detection. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 971–978. IEEE, 2019.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pp. 1310–1320. PMLR, 2019.

Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and trends in Robotics*, 2(1-2):388–403, 2013.

Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high dimensions without the computational intractability. In *57th Annual Symposium on Foundations of Computer Science*, pp. 655–664. Institute of Electrical and Electronics Engineers (IEEE), 2016.

Michael Everett, Björn Lütjens, and Jonathan P How. Certifiable robustness to adversarial state uncertainty in deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Marc Fischer, Matthew Mirman, Steven Stalder, and Martin Vechev. Online robustness training for deep reinforcement learning. *arXiv preprint arXiv:1911.00887*, 2019.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15262–15271, 2021.

Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Yunhan Huang and Quanyan Zhu. Deceptive reinforcement learning under adversarial manipulations on cost signals. In *International Conference on Decision and Game Theory for Security*, pp. 217–237. Springer, 2019.

Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pp. 85–103. Springer, 1972.

Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. Trojdrl: evaluation of backdoor attacks on deep reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2020.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning-industry perspectives. In *2020 IEEE Security and Privacy Workshops (SPW)*, pp. 69–75. IEEE, 2020.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Robert Legenstein, Niko Wilbert, and Laurenz Wiskott. Reinforcement learning on slow features of high-dimensional input streams. *PLoS computational biology*, 6(8):e1000894, 2010.

Alexander Levine and Soheil Feizi. Deep partition aggregation: Provable defenses against general poisoning attacks. In *International Conference on Learning Representations*, 2020.

Alexander Levine and Soheil Feizi. Deep partition aggregation: Provable defenses against general poisoning attacks. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=YUGG2tFuPM`.

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

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 273–294. Springer, 2018.

Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Xiaojin Zhu. Policy poisoning in batch reinforcement learning and control. *Advances in Neural Information Processing Systems*, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pp. 3578–3586. PMLR, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, Udam Saini, Charles Sutton, J Doug Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter. *LEET*, 8:1–9, 2008.

Tuomas Oikarinen, Tsui-Wei Weng, and Luca Daniel. Robust deep reinforcement learning through adversarial loss. *arXiv preprint arXiv:2008.01976*, 2020.

Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *17th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2018*, pp. 2040–2042. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2018.

Neehar Peri, Neal Gupta, W Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P Dickerson. Deep k-nn defense against clean-label data poisoning attacks. In *European Conference on Computer Vision*, pp. 55–70. Springer, 2020.

Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *International Conference on Machine Learning*, pp. 7974–7984. PMLR, 2020.

Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.

Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sébastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 11292–11303, 2019.

Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In *International Conference on Machine Learning*, pp. 9389–9398. PMLR, 2021.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 3520–3532, 2017.

Yanchao Sun, Da Huo, and Furong Huang. Vulnerability-aware poisoning mechanism for online {rl} with unknown dynamics. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=9r30XCjf5Dt`.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Rishi Veerapaneni, John D Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. In *Conference on Robot Learning*, pp. 1439–1456. PMLR, 2020.

Lun Wang, Zaynah Javed, Xian Wu, Wenbo Guo, Xinyu Xing, and Dawn Song. Backdoorl: Backdoor attack against competitive reinforcement learning. *arXiv preprint arXiv:2105.00579*, 2021.

Yue Wang, Esha Sarkar, Wenqing Li, Michail Maniatakos, and Saif Eddin Jabari. Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems. *arXiv preprint arXiv:2003.07859*, 2020.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Maurice Weber, Xiaojun Xu, Bojan Karlas, Ce Zhang, and Bo Li. Rab: Provable robustness against backdoor attacks. *arXiv preprint arXiv:2003.08904*, 2020.

Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pp. 5276–5285. PMLR, 2018.

Fan Wu, Linyi Li, Zijian Huang, Yevgeniy Vorobeychik, Ding Zhao, and Bo Li. Crop: Certifying robust policies for reinforcement learning through functional smoothing. *arXiv preprint arXiv:2106.09292*, 2021.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34, 2021.

Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21024–21037. Curran Associates, Inc., 2020a. URL `https://proceedings.neurips.cc/paper/2020/file/f0eb6568ea114ba6e293f903c34d7488-Paper.pdf`.

Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning*, pp. 11225–11234. PMLR, 2020b.

Xuezhou Zhang, Yiding Chen, Jerry Zhu, and Wen Sun. Corruption-robust offline reinforcement learning. *arXiv preprint arXiv:2106.06630*, 2021.

A OMITTED DEFINITIONS

A.1 CUMULATIVE REWARD

Definition 6 (Cumulative Reward)**.** Let $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ be the transition function with $\mathcal{P}(\cdot)$ defining the set of probability measures. Let R, d_0, H be the reward function, initial state distribution, and time horizon. We denote $J(\pi)$ as the *cumulative reward* of the given policy π:

$$J(\pi) := \mathbb{E} \sum_{t=0}^{H-1} R(s_t, a_t), \text{ where } a_t = \pi(s_t), s_{t+1} \sim P(s_t, a_t), s_0 \sim d_0, \tag{11}$$

It is natural to adapt this definition when there is a discount factor. If policy π considers recent n_t states instead of only s_t to make action predictions, we can change $a_t = \pi(s_t)$ to $a_t = \pi(s_{t-n_t+1:t})$ in Equation (11).

A.2 PARL (PER-STATE PARTITION AGGREGATION) PROTOCOL

Definition 7 (Per-State Partition Aggregation)**.** Given subpolicies $\{\pi_i\}_{i=0}^{u-1}$, the Per-State Partition Aggregation (PARL) protocol defines an aggregated policy $\pi_{\mathrm{P}} : \mathcal{S} \to \mathcal{A}$ such that

$$\pi_{\mathrm{P}}(s) := \arg\max_{a \in \mathcal{A}} n_a(s), \tag{12}$$

where $n_a(s)$ is defined in Section 2.

As mentioned in Section 3, when the $\arg\max$ in Equation (12) returns multiple elements, we break ties deterministically by returning the "smaller" ($<$) action, which can be defined by numerical order, lexicographical order, etc.

A.3 TPARL (TEMPORAL PARTITION AGGREGATION) PROTOCOL

Definition 8 (Temporal Partition Aggregation)**.** Given subpolicies $\{\pi_i\}_{i=0}^{u-1}$ and window size W, at time step t, the Temporal Partition Aggregation (TPARL) defines an aggregated policy $\pi_{\mathrm{T}} : \mathcal{S}^{\min\{t+1,W\}} \to \mathcal{A}$ such that

$$\pi_{\mathrm{T}}(s_{\max\{t-W+1,0\}:t}) = \arg\max_{a \in \mathcal{A}} n_a(s_{\max\{t-W+1,0\}:t}), \tag{13}$$

where $s_{l:r}$ and n_a are defined in Section 2.

In the above definition, at time step t, the input of policy π_{T} contains all states between time step $\max\{t - W + 1, 0\}$ and t within window size W; and the output is the policy prediction with the highest votes across these states. Recall that in Section 2, we define $n_a(s_{l:r}) = \sum_{j=l}^{r} n_a(s_j)$. We break ties in the same way as in PARL.

B ILLUSTRATION OF COPA AGGREGATION PROTOCOLS

We use a concrete example to illustrate bottleneck states and non-bottleneck states, and demonstrate how different aggregation protocols induce different tolerable poisoning thresholds.

Suppose the action space contains two actions $\mathcal{A} = \{a_1, a_2\}$, and there are six subpolicies $\{\pi_i\}_{i=0}^{u-1}$ where $u = 6$. We are at time step $t = 7$ now. When there is no poisoning, the subpolicies' action predictions for state s_0 to s_7 are shown by Table 2. After aggregation, all aggregated policies will choose action a_1 at $t = 7$.

Bottleneck States and Non-Bottleneck States. According to our illustration in Section 3.2, bottleneck states are those where subpolicies vote for diverse actions but the best action is the same as previous states, and non-bottleneck states are those where subpolicies mostly vote for the same action. As we can see, s_0 to s_6 are non-bottleneck states since the sub-policies almost unanimously vote for one action. In contrast, s_7 is a bottleneck state.

PARL (Definition 7). The PARL aggregation protocol only uses the current state s_7 to aggregate the votes. On s_7, three subpolicies choose action a_1 and three others choose action a_2. By breaking the tie with $a_1 < a_2$, the PARL policy $\pi_{\mathrm{P}}(s_7) = a_1$. The tolerable poisoning threshold $\overline{K}_t = 0$,

Table 2: A concrete example of action predictions, where "1" means action a_1 and "2" means action a_2. When there is no poisoning attack, the corresponding time window spans by PARL, TPARL, and DPARL are shown by green, blue, and pink respectively. All aggregated policies choose action a_1, but have different tolerable poisoning thresholds as shown in the last column.

Action Prediction for	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	$\overline{K}$ at s_7
π_0	1	1	1	1	1	1	1	1	
π_1	1	1	1	1	1	1	1	1	
π_2	1	1	1	1	1	1	1	1	
π_3	1	1	1	1	1	1	1	2	
π_4	1	1	1	1	1	1	1	2	
π_5	1	1	1	2	1	2	2	2	
PARL (π_{P}, Definition 7)									0
TPARL with $W = 7$ (π_{T}, Definition 8)									2
DPARL with $W_{\max} = 8$ (π_{D}, Definition 3)									1

because one action flipping from a_1 to a_2 by poisoning only one subpolicy can change the aggregated policy to a_2.

TPARL (Definition 8). The TPARL aggregation protocol uses window size $W = 7$. This implies that, we aggregate all states $s_{1:7}$ to count the votes and decide the action. Since $n_{a_1}(s_{1:7}) = 36$ and $n_{a_2}(s_{1:7}) = 6$, after poisoning two subpolicies, we still $\widetilde{n}_{a_1}(s_{1:7}) \geq \widetilde{n}_{a_2}(s_{1:7})$. Thus, we achieve an tolerable poisoning threshold $\overline{K}_t = 2$.

Compared with PARL, the temporal aggregation in TPARL increases the vote margin between a_1 and a_2 and thus improve the tolerable poisoning threshold at current state.

DPARL (Definition 3). The DPARL aggregation protocol chooses the window size $W' = 8$ since this window size gives the largest average vote margin $\Delta_t^{W'}$ (defined in Equation (1)). We can easily find out that with poisoning size $K = 1$, we will still choose $\widetilde{W}' = 8$ as the window size and the resulting votes for a_1 can be kept higher than votes for a_2. However, when it comes to $K = 2$, if we totally flip subpolicies π_0 and π_1 to let them always predict action a_2, then the DPARL aggregated policy will turn to choose $W' = 1$ as the window size and then choose action a_2 as the action output. Thus, we achieve a tolerable poisoning threshold $\overline{K}_t = 1$ this time.

In summary, we can observe that for the bottleneck state s_7, both TPARL and DPARL exploit temporal aggregation to boost the certified robust poisoning threshold (from 0 to 2 and from 0 to 1 respectively), thus demonstrating the effectiveness of TPARL and DPARL on improving certified robustness. On the other hand, for non-bottleneck states s_0 to s_6, we can easily see that different aggregation protocols do not differ much in terms of provided certified robustness levels.

More Discussion about Bottleneck and Non-Bottleneck States. We provide additional discussions regarding why bottleneck states are vulnerable and why our temporal aggregation strategy can effectively alleviate the problem. We first explain the existence of the bottleneck states. Given the property of the bottleneck states that there is high disagreement among different sub-policies on such states, they may lie close to the decision boundary. This may be a result of poisoning, or simply due to the intrinsic difficulty of the state. On the other hand, such states naturally exist in the rollouts (like natural adversarial examples (Hendrycks et al., 2021)), and may induce high instability of the rollouts during testing. We next discuss additional intuitions for our temporal aggregation. Essentially, temporal aggregation effectively leverages the adjacent non-bottleneck states to rectify the decisions at the bottleneck states, based on the assumption of temporal continuity which has been revealed and utilized in several previous works (Legenstein et al., 2010; Veerapaneni et al., 2020; Ye et al., 2021).

C Certification of Per-State Action Stability for PARL

We certify the robustness of PARL following Theorem 8.

Theorem 8. *Let D be the clean training dataset; let $\pi_i = \mathcal{M}_0(D_i), 0 \leq i \leq u-1$ be the learned subpolicies according to Section 3.1 from which we define n_a (see Section 2); and let π_{P} be the Per-State Partition Aggregation policy: $\pi_{\mathrm{P}} = \mathcal{M}(D)$ where $\mathcal{M}$ abstracts the whole training-aggregation process. $\widetilde{D}$ is a poisoned dataset and $\widetilde{\pi_{\mathrm{P}}}$ is the poisoned policy: $\widetilde{\pi_{\mathrm{P}}} = \mathcal{M}(\widetilde{D})$.*

For a given state s_t encountered at time step t during test time, let $a := \pi_{\mathrm{P}}(s_t)$, then

$$\overline{K}_t = \left\lfloor \frac{n_a(s_t) - \max_{a' \neq a} \left(n_{a'}(s_t) + \mathbf{1}[a' < a]\right)}{2} \right\rfloor. \tag{14}$$

Remark. The theorem provides a valid per-state action certification for PARL, since according to Definition 1, as long as the poisoning size is smaller or equal to $\overline{K}_t$, *i.e.*, $|D \ominus \widetilde{D}| \leq \overline{K}_t$, the action of the poisoned policy is the same as the clean policy: $\widetilde{\pi_{\mathrm{P}}}(s_t) = \pi_{\mathrm{P}}(s_t) = a$. To compute $\overline{K}_t$, according to Equation (14), we rely on the aggregated action count n_a for state s_t from subpolicies. The $a' < a$ is the "smaller-than" operator introduced following Definition 7.

Proof of Theorem 8. Suppose the poisoning size is within K, then after poisoning, the aggregated action count $\widetilde{n}_a(s_t) \in [n_a(s_t) - K, n_a(s_t) + K]$ where $n_a(s_t)$ is such count before poisoning. Thus, to ensure the chosen action does not change, a necessary condition is that $\widetilde{n}_a(s_t) \geq \widetilde{n}_{a'}(s_t) + \mathbf{1}[a' < a]$ for any other $a' < a$. Solving K yields Equation (14). □

We use the theorem to compute the per-state action certification for each time step t along the trajectory, and the detailed algorithm is in Algorithm 2 (Appendix E). Furthermore, we prove that this certification is theoretically tight as the following proposition shows. The proof is in Appendix F.1.

Proposition 9. *Under the same condition as Theorem 8, for any time step t, there exists an RL learning algorithm $\mathcal{M}_0$, and a poisoned dataset $\widetilde{D}$, such that $|D \ominus \widetilde{D}| = \overline{K}_t + 1$, and $\widetilde{\pi_{\mathrm{P}}}(s_t) \neq \pi_{\mathrm{P}}(s_t)$.*

D A Table of Possible Action Set

Table 3: Expressions of possible action set $A(K)$ given poisoning threshold K for different aggregation protocols. Full theorem statements are in Theorems 4, 5 and 7 (in Section 3.4).

Protocol	$A(K)$
PARL	$\left\{a \in \mathcal{A} \,\middle\vert\, \sum_{a' \in \mathcal{A}} \max\{n_{a'}(s_t) - n_a(s_t) - K + \mathbf{1}[a' < a], 0\} \leq K\right\}$
TPARL	$\left\{a \in \mathcal{A} \,\middle\vert\, \sum_{i=1}^{K} h_{a',a}^{(i)} > \delta_{a',a}, \forall a' \neq a\right\}$
DPARL	$\{a_t\} \cup \left\{a' \in \mathcal{A} \,\middle\vert\, \min_{1 \leq W^* \leq \min\{W_{\max}, t+1\}, W^* \neq W', a'' \neq a_t} L_{a',a''}^{W^*,W'} \leq K\right\} \cup \left\{a \in \mathcal{A} \,\middle\vert\, \sum_{i=1}^{K} h_{a',a}^{(i)} > \delta_{a',a}, \forall a' \neq a\right\}$

For all three aggregation protocols, we summarize the expressions used for computing the possible action set $A(K)$ given tolerable poisoning threshold K in Table 3.

E Algorithm Pseudocode and Discussion

E.1 COPA Training Protocol

Algorithm 1: COPA training protocol.

Input: training dataset D, number of partitions k, deterministic hash function h
Output: COPA subpolicies $\{\pi_i\}_{i=0}^{k-1}$

for $i \in [k]$ **do**
  $D_i \leftarrow \{\tau \in D \mid h(\tau) \equiv i \pmod{k}\}$ ▷ `Separate the training data` D `into` k `partitions`
for *each partition* D_i **do**
  $\pi_i \leftarrow \mathcal{M}_0(D_i)$ ▷ `Subpolicy trained on partition` D_i `with offline RL algorithm` $\mathcal{M}_0$
return $\{\pi_i\}_{i=0}^{k-1}$

E.2 COPA PER-STATE ACTION CERTIFICATION

Per-State Partition Aggregation (PARL).

Algorithm 2: COPA per-state certification algorithm for Per-State Partition Aggregation (PARL).

Input: environment $\mathcal{E} = (\mathcal{S}, \mathcal{A}, R, P, H, d_0)$, subpolicies $\{\pi_i\}_{i=0}^{k-1}$
Output: COPA robust size at each time step $\{\overline{K}_t\}_{t=0}^{H-1}$

$s_0 \sim d_0$ ▷ `sample initial state`
for t *from* 0 *to* $H-1$ **do**
  for *each* $a \in \mathcal{A}$ **do**
    Compute $n_a(s_t)$ from subpolicies' $\{\pi_i\}_{i=0}^{k-1}$ decisions ▷ n_a `is defined in Section 2`
  Determine the chosen action $a_t \leftarrow \pi_{\mathrm{P}}(s_t)$ according to PARL (Definition 7)
  Compute $\overline{K}_t$ according to Equation (14) in Theorem 8
  $s_{t+1} \sim P(s_t, a_t)$
return $\{\overline{K}_t\}_{t=0}^{H-1}$

Temporal Partition Aggregation (TPARL).

Algorithm 3: COPA per-state certification algorithm for Temporal Partition Aggregation (TPARL).

Input: environment $\mathcal{E} = (\mathcal{S}, \mathcal{A}, R, P, H, d_0)$, subpolicies $\{\pi_i\}_{i=0}^{k-1}$, window size W
Output: COPA robust size at each time step $\{\overline{K}_t\}_{t=0}^{H-1}$

$s_0 \sim d_0$ ▷ `sample initial state`
for t *from* 0 *to* $H-1$ **do**
  for *each* $a \in \mathcal{A}$ **do**
    Compute $n_a(s_t)$ from subpolicies' $\{\pi_i\}_{i=0}^{k-1}$ decisions ▷ n_a `is defined in Section 2`
  Determine the chosen action $a_t \leftarrow \pi_{\mathrm{T}}(s_t)$ according to TPARL (Definition 8)
  $p_{\min} \leftarrow \infty$
  for *each* $a' \in \mathcal{A}, a' \neq a_t$ **do**
    for i *from* 0 *to* $k-1$ **do**
      Compute $h_{i,a_t,a'}$ according to Equation (4)
    $\{h^{(i)}_{a_t,a'}\}_{i=1}^{k} \leftarrow \mathsf{sorted}(\{h_{i,a_t,a'}\}_{i=0}^{k-1}, \texttt{reverse} = \texttt{True})$
    Compute $\delta_{a_t,a'}$
    $\texttt{sum} \leftarrow 0, p \leftarrow 0$
    for j *from* 1 *to* k **do**
      if $\texttt{sum} + h^{(j)}_{a_t,a'} > \delta_{a_t,a'}$ **then**
        $p \leftarrow j-1$
        break
      $p \leftarrow j, \texttt{sum} \leftarrow \texttt{sum} + h^{(j)}_{a_t,a'}$
    $p_{\min} \leftarrow \min\{p_{\min}, p\}$
  $\overline{K}_t \leftarrow p_{\min}$
  $s_{t+1} \sim P(s_t, a_t)$
return $\{\overline{K}_t\}_{t=0}^{H-1}$

Dynamic Temporal Partition Aggregation (DPARL).

Algorithm 4: COPA per-state certification algorithm for Dynamic Temporal Partition Aggregation (DPARL).

Input: environment $\mathcal{E} = (\mathcal{S}, \mathcal{A}, R, P, H, d_0)$, subpolicies $\{\pi_i\}_{i=0}^{k-1}$, maximum window size $W_{\max}$
Output: COPA robust size at each time step $\{\overline{K}_t\}_{t=0}^{H-1}$

$s_0 \sim d_0$ ▷ `sample initial state`
for t *from* 0 *to* $H-1$ **do**
  for *each* $a \in \mathcal{A}$ **do**
    Compute $n_a(s_t)$ from subpolicies' $\{\pi_i\}_{i=0}^{k-1}$ decisions given s_t ▷ n_a `is defined in Section 2`
  Determine the chosen action $a_t \leftarrow \pi_{\mathrm{D}}(s_t)$ and chosen window size W' according to DPARL (Definition 3)
  $p_{\min} \leftarrow \infty$
  $\overline{K}_t \leftarrow$ Algorithm 3 ▷ `use Algorithm 3 with` W `replaced by` W' `to compute` $\overline{K}_t$
  $\overline{K}_t^D \leftarrow \overline{K}_t$
  for *each* $a' \in \mathcal{A}, a' \neq a_t$ **do**
    for *each* $a'' \in \mathcal{A}, a'' \neq a_t$ **do**
      for W^* *from* 1 *to* $\min\{W_{\max}, t+1\}$ **do**
        $a^{\#} \leftarrow \arg\max_{a_0 \neq a', a_0 \in \mathcal{A}} n_{a_0}(s_{t-W^*+1:t})$
        for w *from* 1 *to* $\max\{W', W^*\}$ **do**
          Compute $\max_{a_0 \in \mathcal{A}} \sigma^w(a_0)$ according to Equation (7)
        for i *from* 0 *to* $k-1$ **do**
          for w *from* 1 *to* $\max\{W', W^*\}$ **do**
            Compute $\sigma^w(\pi_i(s_{t-w}))$ according to Equation (7)
          Compute g_i according to Equation (7)
        $\{g^{(i)}\}_{i=1}^{k} \leftarrow \mathtt{sorted}(\{g_i\}_{i=0}^{k-1}, \mathtt{reverse} = \mathtt{True})$
        Compute $n_{a'}^{W^*}, n_{a^{\#}}^{W^*}, n_{a_t}^{W'}, n_{a''}^{W'}$
        $\mathtt{tmp} \leftarrow W'(n_{a'}^{W^*} - n_{a^{\#}}^{W^*}) - W^*(n_{a_t}^{W'} - n_{a''}^{W'}) - \mathbf{1}[a' > a_t]$
        $\mathtt{sum} \leftarrow 0, p \leftarrow 0$
        for j *from* 1 *to* k **do**
          if $\mathtt{sum} + \mathtt{tmp} \geq 0$ **then**
            $p \leftarrow j-1$
            break
          $p \leftarrow j, \mathtt{sum} \leftarrow \mathtt{sum} + g^{(j)}$
        $L_{a',a''}^{W^*,W'} \leftarrow p, \overline{K}_t^D \leftarrow \min\{\overline{K}_t^D, L_{a',a''}^{W^*,W'}\}$
  $s_{t+1} \sim P(s_t, a_t)$
return $\{\overline{K}_t^D\}_{t=0}^{H-1}$

E.3 COPA Cumulative Reward Certification

COPA-Search alternately executes the procedure of *trajectory exploration and expansion* and *poisoning threshold growth*. In trajectory exploration and expansion, COPA-Search organizes all possible trajectories in the form of search tree and progressively grows it. For each node (representing a game state), we leverage Theorems 4, 5 and 7 to compute the Possible Action Set. We then expand the tree branches corresponding to the actions in the derived set. In poisoning threshold growth, when all trajectories for the current poisoning threshold are explored, we increase K' to seek for certification under larger poisoning sizes, via maintaining a priority queue of all poisoning sizes during the expansion of the tree. The iterative procedures end when the priority queue becomes empty.

The highlighted line in the following algorithm needs to inject different algorithms based on the aggregation protocol: for PARL (π_{P}), use Theorem 4; for TPARL (π_{T}), use Theorem 5; for DPARL (π_{D}), use Theorem 7.

Algorithm 5: COPA-SEARCH: adaptive tree search for cumulative reward certification.

Input: environment $\mathcal{E} = (\mathcal{S}, \mathcal{A}, R, P, H, d_0)$, subpolicies $\{\pi_i\}_{i=0}^{k-1}$, aggregated policy π_{P} or π_{T} (with W) or π_{T} (with $W_{\max}$)

Output: a map M that maps poisoning size K to corresponding certified lower bound of cumulative reward $\underline{J}_K$

▷ Initialize global variables

p_que $\leftarrow \emptyset$ ▷ initialize an empty priority queue containing tuples of (state history $s_{0:t}$, action a, poisoning size K, reward J) sorted by increasing K

$J_{\mathrm{global}} \leftarrow \infty$ ▷ initialize global minimum reward

Function GETACTIONS $(s_{0:t}, K_{\mathrm{lim}}, J_{\mathrm{cur}})$:

  $A \leftarrow$ possibleActions$(s_{0:t}, \{\pi_i\}_{i=1}^{k-1}, \pi_*, K_{\mathrm{lim}})$ ▷ compute the possible action set given state history, subpolicies, aggregation policy π_*, and poisoning size according to theorems in Section 3.4

  a_list $\leftarrow \emptyset$

  for *each action* $a \in \mathcal{A}$ **do**

    if $P(s, a) = \perp$ **then**

      continue ▷ game terminate here, no possible larger or lower cumulative reward to search

    a_list $\leftarrow$ a_list $\cup \{a\}$ ▷ record possible actions to expand

  if $A \neq \mathcal{A}$ **then**

    $K' \leftarrow \min_{\text{possibleActions}(s_{0:t}, \{\pi_i\}_{i=0}^{k-1}, \pi_*, K) \neq A} K$

    $A_{\mathrm{new}} \leftarrow$ possibleActions$(s_{0:t}, \{\pi_i\}_{i=1}^{k-1}, \pi_*, K') \setminus A$ ▷ compute the immediate actions that will possibly be chosen if enlarging the poisoning size

    for *each action* $a \in A_{\mathrm{new}}$ **do**

      p_que.push$((s_{0:t}, a, K', J_{\mathrm{cur}}))$

  return a_list

Procedure EXPAND $(s_{0:t}, K_{\mathrm{lim}}, J_{\mathrm{cur}})$:

  if $J_{\mathrm{cur}} \geq J_{\mathrm{global}} \wedge$ (*step reward is non-negative*) **then**

    return ▷ pruning

  a_list $\leftarrow$ GETACTIONS $(s_{0:t}, \{\pi_i\}_{i=0}^{k-1}, \pi_*, K_{\mathrm{lim}})$ ▷ compute according to theorems in Section 3.4

  if a_list $= \emptyset$ **then**

    $J_{\mathrm{global}} \leftarrow \min\{J_{\mathrm{global}}, J_{\mathrm{cur}}\}$

    return

  for $a \in$ a_list **do**

    $s_{t+1} \leftarrow P(s_t, a)$

    if $s_{t+1} = \perp$ **then**

      $J_{\mathrm{global}} \leftarrow \min\{J_{\mathrm{global}}, J_{\mathrm{cur}}\}$

    else

      EXPAND $(s_{0:t+1}, K_{\mathrm{lim}}, J_{\mathrm{cur}} + R(s_t, a))$

$M \leftarrow \emptyset$

$s_0 \leftarrow d_0$ ▷ initial state is s_0

EXPAND $(s_0, K_{\mathrm{lim}} = 0, J_{\mathrm{cur}} = 0)$ ▷ expand initial trajectory

while True **do**

  if p_que $= \emptyset$ **then**

    break ▷ no state to expand

  $(s_{0:t}, a, K, J) \leftarrow$ p_que.pop() ▷ pop out the first element

  $(_, _, K', _) \leftarrow$ p_que.top() ▷ examine the next element

  $M(K) \leftarrow J_{\mathrm{global}}$ ▷ obtain one new point of certification result

  $s_{t+1} \leftarrow P(s_t, a)$

  EXPAND $(s_{0:t+1}, K', J + R(s_t, a))$ ▷ expand the tree from the new node

return M ▷ indeed the algorithm can terminate at any time within the while loop

Time Complexity. The complexity of COPA-SEARCH is $O(H|\mathcal{S}_{\text{explored}}|(\log|\mathcal{S}_{\text{explored}}| + |\mathcal{A}|T))$, where $|\mathcal{S}_{\text{explored}}|$ is the number of explored states throughout the search procedure, which is no larger than cardinality of state set $\mathcal{S}$, H is the horizon length, $|\mathcal{A}|$ is the cardinality of action set, and T is the time complexity of per-state action certification. The main bottleneck of the algorithm is the large number of possible states, which is in the worse case exponential to state dimension. However, to provide a deterministic worst-case certification agnostic to game properties, exploring all possible states may be inevitable.

Relation with Wu et al. (2021). The COPA-SEARCH algorithm is inspired from Wu et al. (2021, Algorithm 3), which also leverages tree search to explore all possible states and thus derive the robustness guarantee. However, the major distinction is that their algorithm tries to derive a probabilistic guarantee of RL robustness against state perturbations, while COPA-SEARCH derives deterministic guarantee of RL robustness against poisoning attacks. We think this general tree search methodology can be further extended to provide certification beyond evasion attack (Wu et al., 2021) and poisoning attack (our work) and we leave it as future work.

F PROOFS

F.1 TIGHTNESS OF PER-STATE ACTION CERTIFICATION IN PARL

Proof of Proposition 9. We prove by construction. Given the state s_t, we first locate the subpolicies whose chosen action is a, and denote the set of them as

$$B = \{i \in [u] \mid \pi_i(s_t) = a = \pi_{\text{P}}(s_t)\}.$$

We also denote $a' = \arg\max_{a' \neq a} n_{a'}(s_t) + \mathbf{1}[a' < a]$. According to $\overline{K}_t$'s definition (Equation (14)),

$$|B| = n_a(s_t) > n_a(s_t)/2 \geq \overline{K}_t.$$

We now pick an arbitrary subset $B' \subseteq B$ such that $|B'| = \overline{K}_t + 1$. For each $i \in B'$, we locate its corresponding parititioned dataset D_i for training subpolicy π_i. We insert one point p_i to D_i, such that our chosen learning algorithm $\mathcal{M}_0$ can train a subpolicy $\tilde{\pi}_i = \mathcal{M}_0(D_i \cup \{p_i\})$ that satisfies $\tilde{\pi}_i(s_t) = a'$. For example, the point could be $p_i = \{(s_t, a', s', \infty)\}$ and $\mathcal{M}_0$ learns the action with maximum reward for the memorized nearest state, where s' can be adjusted to make sure the point is hashed to parition i.[1] Then, we construct the poisoned dataset

$$\widetilde{D} = \left(\bigcup_{i \in B'} D_i \cup \{p_i\} \right) \cup \left(\bigcup_{i \in [u] \setminus B'} D_i \right).$$

Therefore, we have $\widetilde{D} \ominus D = \cup_{i \in B'} p_i = |B'| = \overline{K}_t + 1$.

On this poisoned dataset, we train subpolicies $\{\tilde{\pi}_i\}_{i=0}^{u-1}$ and get the aggregated policy $\widetilde{\pi_{\text{P}}}$. To study $\widetilde{\pi_{\text{P}}}(s_t)$, we compute the aggregated action count $\widetilde{n}_a$ on these poisoned subpolicies. We found that $\widetilde{n}_a(s_t) = n_a(s_t) - |B'|$ and $\widetilde{n}_{a'}(s_t) = n_{a'}(s_t) + |B'|$. Therefore,

$$\begin{aligned}
\widetilde{n}_a(s_t) - \widetilde{n}_{a'}(s_t) &= n_a(s_t) - n_{a'}(s_t) - 2(\overline{K}_t + 1) \\
&= n_a(s_t) - n_{a'}(s_t) - 2\left(\left\lfloor \frac{n_a(s_t) - n_{a'}(s_t) - \mathbf{1}[a' < a]}{2} \right\rfloor + 1 \right) \\
&< n_a(s_t) - n_{a'}(s_t) - (n_a(s_t) - n_{a'}(s_t) - \mathbf{1}[a' < a]) = \mathbf{1}[a' < a].
\end{aligned}$$

Therefore, if $a' < a$, $\widetilde{n}_a(s_t) \leq \widetilde{n}_{a'}(s_t)$, and a' has higher priority to be chosen than a; if $a' > a$, $\widetilde{n}_a(s_t) \leq \widetilde{n}_{a'}(s_t) - 1$, and a' still has higher priority to be chosen than a. Hence, $\widetilde{\pi_{\text{P}}}(s_t) \neq a = \pi_{\text{P}}(s_t)$. To this point, we conclude the proof with a feasible construction. □

[1] Strictly speaking, we need to choose a determinisitc hash function h for dataset partitioning such that adjustment on s' and reward (currently ∞, but can be an arbitrary large enough number) can make the point being partitioned to i, *i.e.*, $h(p_i) \equiv i \pmod{u}$. Since our adjustment space is infinite due to infinite number of large enough reward, such assumption can be easily achieved. Same applies to other attack constructions in the following proofs.

F.2 PER-STATE ACTION CERTIFICATION IN TPARL

Proof of Theorem 1. For ease of notation, we let $w = \min\{W, t+1\}$ so w is the actual window size used at step t. We let $t_0 = t - w + 1$, *i.e.*, t_0 is the actual start time step for TPARL aggregation at step t. Now we can write the chosen action at step t without poisoning as $a = \pi_{\mathrm{T}}(s_{t_0:t})$.

We prove the theorem by contradiction: We assume that there is a poisoning attack whose poisoning size $K \leq \overline{K}_t$ where $\overline{K}_t$ is defined by Equation (3) in the theorem, and after poisoning, the chosen action is $a^T \neq a$. We denote $\{\tilde{\pi}_i\}_{i=0}^{u-1}$ to the poisoned subpolicies and $\widetilde{\pi_{\mathrm{T}}}$ to the poisoned TPARL aggregated policy. From the definition of $\overline{K}_t$, we have

$$\sum_{i=1}^{K} h_{a,a^T}^{(i)} \leq \sum_{i=1}^{\overline{K}_t} h_{a,a^T}^{(i)} \leq \delta_{a,a^T}, \tag{15}$$

since $K \leq \overline{K}_t$, and each $h_{a,a^T}^{(i)}$ is an element of h_{i',a,a^T} for some i' where

$$h_{i',a,a^T} = \sum_{j=0}^{w-1} \mathbf{1}_{i',a}(s_{t-j}) + w - \sum_{j=0}^{w-1} \mathbf{1}_{i',a^T}(s_{t-j}) \geq 0$$

by Equation (4) in the theorem. Since the poisoning attack within threshold K can at most affect K subpolicies, we let B be the set of affected policies and assume $|B| = K$ without loss of generality. Formally, $B = \{i \in [u] \mid \exists t' \in [t_0, t], \tilde{\pi}_i(s_{t'}) \neq \pi_i(s_{t'})\}$. Therefore, according to the monotonicity of $h_{a,a^T}^{(i)}$, from Equation (15),

$$\sum_{i \in B} h_{i,a,a^T} \leq \sum_{i=1}^{K} h_{a,a^T}^{(i)} \leq \delta_{a,a^T}. \tag{16}$$

According to the assumption of successfully poisoning attack, after attack, the sum of aggregated vote for a^T plus $\mathbf{1}[a^T < a]$ should be larger then that of a. Formally, after the poisoning,

$$\widetilde{n}_{a^T}(s_{t_0:t}) + \mathbf{1}[a^T < a] > \widetilde{n}_a(s_{t_0:t})$$

or equivalently

$$\widetilde{n}_a(s_{t_0:t}) - (\widetilde{n}_{a^T}(s_{t_0:t}) + \mathbf{1}[a^T < a]) < 0.$$

From the statement of Theorem 1,

$$\delta_{a,a^T} = n_a(s_{t_0:t}) - (n_{a^T}(s_{t_0:t}) + \mathbf{1}[a^T < a]).$$

Take the difference of the above two equations, we know that the attack satisfies

$$n_a(s_{t_0:t}) - \widetilde{n}_a(s_{t_0:t}) - (n_{a^T}(s_{t_0:t}) - \widetilde{n}_{a^T}(s_{t_0:t})) > \delta_{a,a^T}. \tag{17}$$

Since the attack only changes the subpolicies in B, we have

$$\begin{aligned}
n_a(s_{t_0:t}) - \widetilde{n}_a(s_{t_0:t}) &= \sum_{i \in B} \left(\sum_{j=0}^{w-1} \mathbf{1}[\pi_i(s_{t-j}) = a] - \sum_{j=0}^{w-1} \mathbf{1}[\tilde{\pi}_i(s_{t-j}) = a] \right) \\
&\leq \sum_{i \in B} \sum_{j=0}^{w-1} \mathbf{1}_{i,a}(s_{t-j}), \\
n_{a^T}(s_{t_0:t}) - \widetilde{n}_{a^T}(s_{t_0:t}) &= \sum_{i \in B} \left(\sum_{j=0}^{w-1} \mathbf{1}[\pi_i(s_{t-j}) = a^T] - \sum_{j=0}^{w-1} \mathbf{1}[\tilde{\pi}_i(s_{t-j}) = a^T] \right) \\
&\geq \sum_{i \in B} \left(\sum_{j=0}^{w-1} \mathbf{1}_{i,a^T}(s_{t_j}) - w \right).
\end{aligned}$$

Inject them into Equation (17) yields

$$\sum_{i \in B} \underbrace{\left(\sum_{j=0}^{w-1} \mathbf{1}_{i,a}(s_{t-j}) + w - \sum_{j=0}^{w-1} \mathbf{1}_{i,a^T}(s_{t-j}) \right)}_{h_{i,a,a^T}} > \delta_{a,a^T}$$

which contradicts Equation (16) and thus concludes the proof. □

F.3 TIGHTNESS OF PER-STATE ACTION CERTIFICATION IN TPARL

Proof of Proposition 2. For ease of notation, we let $w = \min\{W, t+1\}$ so w is the actual window size used at step t. We let $t_0 = t - w + 1$, *i.e.*, t_0 is the actual start time step for TPARL aggregation at step t. Now we can write the chosen action at step t without poisoning as $a = \pi_{\mathrm{T}}(s_{t_0:t})$.

We prove by construction, *i.e.*, we construct an poisoning attack with poisoning size $\overline{K}_t + 1$ that deviates the prediction of the poisoned policy. Specifically, we aim to craft a poisoned dataset $\widetilde{D}$ with poisoning size $|D \ominus \widetilde{D}| = \overline{K}_t + 1$, such that for certain learning algorithm $\mathcal{M}_0$, after partitioning and learning on the poisoned dataset, the poisoned subpolicies $\tilde{\pi}_i = \mathcal{M}_0(\widetilde{D}_i)$ can be aggregated to produce different action prediction: $\widetilde{\pi_{\mathrm{T}}}(s_{t_0:t}) \neq a$.

Before construction, we first show that $\overline{K}_t$ given by Equation (3) satisfies $\overline{K}_t < u$. If $\overline{K}_t = u$, it means that for arbitrary $a' \neq a$,

$$\sum_{i=1}^{u} h_{a,a'}^{(i)} = \sum_{i=0}^{u-1} h_{i,a,a'} = \sum_{i=0}^{u-1} \left(\sum_{j=0}^{w-1} \mathbf{1}_{i,a}(s_{t-j}) + w - \sum_{j=0}^{w-1} \mathbf{1}_{i,a'}(s_{t-j}) \right) = n_a(s_{t_0:t}) + uw - n_{a'}(s_{t_0:t})$$

$$\leq \delta_{a,a'} = n_a(s_{t_0:t}) - n_{a'}(s_{t_0:t}) - \mathbf{1}[a' < a],$$

which implies $uw \leq 0$ contracting $uw > 0$. Now, we know $\overline{K}_t < u$, so $\overline{K}_t + 1$, our poisoning size, is smaller or equal to u.

We start our construction by choosing an action a^T:

$$a^T = \operatorname*{arg\,min}_{a' \neq a, a' \in \mathcal{A}} \operatorname*{arg\,max}_{\sum_{i=1}^{p} h_{a,a'}^{(i)} \leq \delta_{a,a'}} p.$$

According to the definition of $\overline{K}_t$ in Equation (3), for a^T we have

$$\sum_{i=1}^{\overline{K}_t + 1} h_{a,a^T}^{(i)} > \delta_{a,a^T}. \tag{18}$$

We locate the subpolicies to poison as

$$B = \{i \in [u] \mid h_{a,a^T}^{i} \text{ is } h_{a,a^T}^{(j)} \text{ in the nonincreasing permutation in Equation (4)}, j \leq \overline{K}_t + 1\} \tag{19}$$

Therefore, $|B| = \overline{K}_t + 1$ and

$$\sum_{i \in B} h_{i,a,a^T} > \delta_{a,a^T} \tag{20}$$

by Equation (18). For each of these subpolicies $i \in B$, we locate its corresponding partitioned dataset D_i for training subpolicy π_i, and insert one trajectory p_i to D_i, such that our chosen learning algorithm $\mathcal{M}_0$ can train a subpolicy $\tilde{\pi}_i = \mathcal{M}(D_i \cup \{p_i\})$ satisfying $\tilde{\pi}_i(s_{t'}) = a^T$ for any $t' \in [t_0, t]$. For example, the trajectory could be $p_i = \{(s_{t'}, a^T, s', \infty)\}_{t'=t_0}^{t}$ and $\mathcal{M}_0$ learns the action with maximum reward for the memorized nearest state, where s' can be adjusted to make sure the trajectory is hashed to partition i. Then, we construct the poisoned dataset

$$\widetilde{D} = \left(\bigcup_{i \in B'} D_i \cup \{p_i\} \right) \cup \left(\bigcup_{i \in [u] \setminus B'} D_i \right).$$

Therefore, we have $\widetilde{D} \ominus D = \cup_{i \in B'} p_i = |B'| = \overline{K}_t + 1$. Now, we compare the aggregated votes for a and a^T before and after poisoning:

$$\widetilde{n}_a(s_{t_0:t}) - n_a(s_{t_0:t}) = \sum_{i \in B} \left(\sum_{j=0}^{w-1} \mathbf{1}[\widetilde{\pi}_i(s_{t-j}) = a] - \sum_{j=0}^{w-1} \mathbf{1}[\pi_i(s_{t-j}) = a] \right)$$

$$= - \sum_{i \in B} \sum_{j=0}^{w-1} \mathbf{1}[\pi_i(s_{t-j}) = a],$$

$$\widetilde{n}_{a^T}(s_{t_0:t}) - n_{a^T}(s_{t_0:t}) = \sum_{i \in B} \left(\sum_{j=0}^{w-1} \mathbf{1}[\widetilde{\pi}_i(s_{t-j}) = a^T] - \sum_{j=0}^{w-1} \mathbf{1}[\pi_i(s_{t-j}) = a^T] \right)$$

$$= \sum_{i \in B} \left(w - \sum_{j=0}^{w-1} \mathbf{1}[\pi_i(s_{t-j}) = a^T] \right).$$

Now we compare the margin between aggregated votes for a and a^T after poisoning:

$$\begin{aligned}
&\widetilde{n}_{a^T}(s_{t_0:t}) - \widetilde{n}_a(s_{t_0:t}) + \mathbf{1}[a^T < a] \\
=&n_{a^T}(s_{t_0:t}) - n_a(s_{t_0:t}) + \mathbf{1}[a^T < a] + \widetilde{n}_{a^T}(s_{t_0:t}) - n_{a^T}(s_{t_0:t}) - (\widetilde{n}_a(s_{t_0:t}) - n_a(s_{t_0:t})) \\
=& - \delta_{a,a^T} + \sum_{i \in B} \left(\sum_{j=0}^{w-1} \mathbf{1}[\pi_i(s_{t-j}) = a] + w - \sum_{j=0}^{w-1} \mathbf{1}[\pi_i(s_{t-j}) = a^T] \right) \\
=& - \delta_{a,a^T} + \sum_{i \in B} h_{i,a,a^T} > 0.
\end{aligned}$$

As a result, after poisoning, a^T has higher priority to be chosen than a, *i.e.*, $\widetilde{\pi_{\mathrm{T}}}(s_{t_0:t}) \neq a = \pi_{\mathrm{T}}(s_{t_0:t})$. To this point, we conclude the proof with a feasible attack construction. □

F.4 Loose Per-State Action Certification in TPARL and Comparision with Tight One

The following corollary of Theorem 8 states a loose per-state action certification.

Corollary 10. *Under the same condition as Theorem 1,*

$$\overline{K}_t = \left\lfloor \frac{n_a(s_{\max\{t-W+1,0\}:t}) - \max_{a' \neq a} \left(n_{a'}(s_{\max\{t-W+1,0\}:t}) + \mathbf{1}[a' < a] \right)}{2 \min\{W, t+1\}} \right\rfloor \tag{21}$$

is a tolerable poisoning threshold *at time step t in Definition 1, where W is the window size.*

Proof of Corollary 10. We let $w = \min\{t+1, W\}$ to be the actual aggregation window size at time step t. After poisoning with size K, at most K subpolicies are affected and each affected policy can only make $\pm w$ changes to the aggregated action count. This implies that, after poisoning, for any action $a' \in \mathcal{A}$, the aggregated vote count $\widetilde{n}_{a'}(s_{t-w+1:t}) \in [n_{a'}(s_{t-w+1:t}) - uw, n_{a'}(s_{t-w+1:t}) + uw]$. Thus, when $K \leq \overline{K}_t$ where $\overline{K}_t$ is defined in Theorem 1, for any $a' \neq a$, we have

$$\begin{aligned}
&\tilde{n}_a(s_{t-w+1:t}) - \tilde{n}_{a'}(s_{t-w+1:t}) - \mathbf{1}[a' < a] \\
=&n_a(s_{t-w+1:t}) - n_{a'}(s_{t-w+1:t}) - \mathbf{1}[a' < a] - 2Kw \\
\geq&n_a(s_{t-w+1:t}) - n_{a'}(s_{t-w+1:t}) - \mathbf{1}[a' < a] - 2w \cdot \overline{K}_t \\
\geq&n_a(s_{t-w+1:t}) - n_{a'}(s_{t-w+1:t}) - \mathbf{1}[a' < a] - \left(n_a(s_{t-w+1:t}) - \max_{a'' \neq a}(n_{a''}(s_{t-w+1:t}) + \mathbf{1}[a'' < a]) \right) \\
=& \max_{a'' \neq a}(n_{a''}(s_{t-w+1:t}) + \mathbf{1}[a'' < a]) - (n_{a'}(s_{t-w+1:t}) + \mathbf{1}[a' < a]) \geq 0.
\end{aligned}$$

From the definition of TPARL protocol, the poisoned policy still chooses action a, which implies $\overline{K}_t$ is a tolerable poisoning threshold. □

In the main text, we mention that the certification from Corollary 10 is looser than that from Theorem 1. This assertion is based on the following two facts:

1. According to Proposition 2, the certification given by Theorem 1 is theoretically tight, which means that any other certification can only be as tight as Theorem 1 or looser than it.
2. There exists examples where the computed $\overline{K}_t$ from Theorem 1 is larger than that from Corollary 10.
   For instance, suppose $W = 5$, action set $\mathcal{A} = \{a_1, a_2\}$, and there are three subpolicies. At time step $t = 4$, π_i for s_0 to s_4 are $[a_1, a_1, a_1, a_1, a_2]$ for all subpolicies (*i.e.*, $i \in [3]$). Thus, the benign policy $\pi_{\mathrm{T}}(s_{0:4}) = a_1$. By computation, the $\overline{K}_t$ from Theorem 1 is 1; while the $\overline{K}_t$ from Corollary 10 is 0.

Indeed, Corollary 10 can be viewed as using $2w$ to upper bound $h_{i,a,a'} = w + \sum_{j=0}^{w-1} \mathbf{1}_{i,a}(s_{t-j}) - \sum_{j=0}^{w-1} \mathbf{1}_{i,a'}(s_{t-j})$. Intuitively, Corollary 10 assumes every subpolicy can provide $2w$ vote margin

shrinkage, and Theorem 1 uses $h_{i,a,a'}$ to capture the precise worse-case margin shrinkage and thus provides a tighter certification.

F.5 PER-STATE ACTION CERTIFICATION IN DPARL

Proof of Theorem 3. Without loss of generality, we assume $W_{\max} \leq t+1$ and otherwise we let $W_{\max} \leftarrow \min\{W_{\max}, t+1\}$. We let $t_0 = \max\{t - W_{\max} + 1, 0\}$ be the start time step of the maximum possible window. To prove the theorem, our general methodology is to enumerate all possible cases of a successful attack, and derive the tolerable poisoning threshold for each case respectively. Taking a minimum over these tolerable poisoning thresholds gives the required result.

Specifically, we denote $\mathcal{P}$ to the predicate of robustness under poisoning attack: $\mathcal{P} = [\widetilde{\pi_{\mathrm{D}}}(s_{t_0:t}) = a]$, and denote K to the poisoning attack size. Therefore, we can decompose $\mathcal{P}$ as such:

$$\mathcal{P} = \mathcal{P}(W') \wedge \bigwedge_{\substack{1 \leq W^* \leq W_{\max}, W^* \neq W' \\ a' \neq a}} \neg \mathcal{Q}(W^*, a'). \tag{22}$$

Recall that W' is the chosen window size by the protocol DPARL with unattacked subpolicies Equation (1). In Equation (22), the predicate $\mathcal{P}(W')$ means that after poisoning attack, whether the prediction under window size W' is still a; the predicate $\mathcal{Q}(W^*, a')$ means that after poisoning attack, whether the chosen action is a' at window size W^* and average vote margin is larger (or equal if $a' < a$) at window size W^* compared to W'. Formally, let $\widetilde{n}_a$ be the aggregated action count after poisoning and $\widetilde{\Delta}_t^W$ be the average vote margin after poisoning at window W (see Equation (2)),

$$\begin{aligned} \mathcal{P}(W') &= \left(\arg\max_{a \in \mathcal{A}} \widetilde{n}_a(s_{t-W'+1:t}) = a \right), \\ \mathcal{Q}(W^*, a') &= \left(\arg\max_{a \in \mathcal{A}} \widetilde{n}_a(s_{t-W^*+1:t}) = a' \right) \wedge \\ &\quad \left(\left(\left(\widetilde{\Delta}_t^{W^*} \geq \widetilde{\Delta}_t^{W'} \right) \wedge (a' < a) \right) \vee \left(\left(\widetilde{\Delta}_t^{W^*} > \widetilde{\Delta}_t^{W'} \right) \wedge (a' > a) \right) \right). \end{aligned} \tag{23}$$

According to Theorem 1,

$$K \leq \overline{K}_t \Longrightarrow \mathcal{P}(W')$$

where $\overline{K}_t$ is defined by Equation (3) with W replaced by W'. The following Lemma 11 shows a sufficient condition for $Q(W^*, a')$. We then aggregate these conditions together with minimum to obtain a sufficient condition for $\mathcal{P}$:

$$K \leq \min \left\{ \overline{K}_t, \min_{1 \leq W^* \leq \min\{W_{\max}, t+1\}, W^* \neq W', a' \neq a, a'' \neq a} L_{a',a''}^{W^*,W'} \right\}$$

and thus conclude the proof. □

Lemma 11. *Let $\mathcal{Q}(W^*, a')$, K, W' be the same as defined in proof of Theorem 3, then*

$$K \leq \min_{a'' \neq a} L_{a',a''}^{W^*,W'} \Longrightarrow \neg \mathcal{Q}(W^*, a'), \tag{24}$$

where $L_{a',a''}^{W^*,W'}$ is defined in Definition 4.

Proof. We prove the equivalent form:

$$\mathcal{Q}(W^*, a') \Longrightarrow K > \min_{a'' \neq a} L_{a',a''}^{W^*,W'}. \tag{25}$$

Suppose a poisoning attack can successfully achieve $Q(W^*, a')$, we now induce the requirement on its poisoning size K. First, we notice that

$$\begin{aligned} &\left(\left(\widetilde{\Delta}_t^{W^*} \geq \widetilde{\Delta}_t^{W'} \right) \wedge (a' < a) \right) \vee \left(\left(\widetilde{\Delta}_t^{W^*} > \widetilde{\Delta}_t^{W'} \right) \wedge (a' > a) \right) \\ \Longleftrightarrow\ & W^* W' \widetilde{\Delta}_t^{W^*} \geq W^* W' \widetilde{\Delta}_t^{W'} + \mathbf{1}[a' > a] \end{aligned} \tag{26}$$

since $W^* W' \widetilde{\Delta}_t^{W*/W^*}$ is an integer by definition. According to the definition and $\mathcal{Q}(W^*, a')$'s assumption $\arg\max_{a \in \mathcal{A}} \widetilde{n}_a(s_{t-W^*+1:t}) = a'$,

$$W^* W' \widetilde{\Delta}_t^{W^*} \leq W' \left(\widetilde{n}_{a'}(s_{t-W^*+1:t}) - \widetilde{n}_{a^\#}(s_{t-W^*+1:t}) \right),$$

where "$\leq$" comes from the fact that the margin in $\widetilde{\Delta}_t^{W^*}$ should be with respect to the runner-up class after poisoning, and computing with respect to any other class provides an upper bound. Here we choose $a^\# = \arg\max_{a_0 \neq a', a_0 \in \mathcal{A}} n_{a_0}(s_{t-W^*+1:t})$ (see Definition 4), the runner-up class before

poisoning, to empirically shrink the gap between the bound and actual margin. On the other hand,

$$W^* W' \widetilde{\Delta}_t^{W'} \geq W^* \left(\widetilde{n}_a(s_{t-W'+1:t}) - \max_{a'' \neq a} \widetilde{n}_{a''}(s_{t-W'+1:t}) \right),$$

where "$\geq$" comes from the fact that the margin in $\widetilde{\Delta}_t^{W'}$ should use the top class after poisoning, and computing with any other class provides a lower bound. Thus, from Equation (26) and the above two relaxations, we get

$$\begin{aligned}
&Q(W^*, a') \\
\Longrightarrow & W' \left(\widetilde{n}_{a'}(s_{t-W^*+1:t}) - \widetilde{n}_{a^\#}(s_{t-W^*+1:t}) \right) \geq W^* \left(\widetilde{n}_a(s_{t-W'+1:t}) - \max_{a'' \neq a} \widetilde{n}_{a''}(s_{t-W'+1:t}) \right) + \mathbf{1}[a' > a] \\
\Longrightarrow & \exists a'' \neq a, \\
& W' \left(\widetilde{n}_{a'}(s_{t-W^*+1:t}) - \widetilde{n}_{a^\#}(s_{t-W^*+1:t}) \right) \geq W^* \left(\widetilde{n}_a(s_{t-W'+1:t}) - \widetilde{n}_{a''}(s_{t-W'+1:t}) \right) + \mathbf{1}[a' > a].
\end{aligned}$$

For each $a'' \neq a$, now we use the last equation as the condition, and show that $K > L_{a',a''}^{W^*,W'}$ is a necessary condition. This proposition is equivalent to

$$\begin{aligned}
&K \leq L_{a',a''}^{W^*,W'} \\
\Longrightarrow & W' \left(\widetilde{n}_{a'}(s_{t-W^*+1:t}) - \widetilde{n}_{a^\#}(s_{t-W^*+1:t}) \right) < W^* \left(\widetilde{n}_a(s_{t-W'+1:t}) - \widetilde{n}_{a''}(s_{t-W'+1:t}) \right) + \mathbf{1}[a' > a].
\end{aligned} \tag{27}$$

Suppose a poisoning attack within poisoning size K changes the subpolicies in set $B \subseteq [u]$. Note that $|B| \leq K$. We inspect the objective in Equation (27):

$$\begin{aligned}
& W' \left(\widetilde{n}_{a'}(s_{t-W^*+1:t}) - \widetilde{n}_{a^\#}(s_{t-W^*+1:t}) \right) - W^* \left(\widetilde{n}_a(s_{t-W'+1:t}) - \widetilde{n}_{a''}(s_{t-W'+1:t}) \right) - \mathbf{1}[a' > a] \\
=& W' \left(n_{a'}(s_{t-W^*+1:t}) - n_{a^\#}(s_{t-W^*+1:t}) \right) - W^* \left(n_a(s_{t-W'+1:t}) - n_{a''}(s_{t-W'+1:t}) \right) - \mathbf{1}[a' > a] \\
& + \sum_{i \in B} \Bigg(W' \sum_{w=0}^{W^*-1} \mathbf{1}[\tilde{\pi}_i(s_{t-w+1}) = a'] - W' \sum_{w=0}^{W^*-1} \mathbf{1}[\tilde{\pi}_i(s_{t-w+1}) = a^\#] \\
& \qquad - W^* \sum_{w=0}^{W'-1} \mathbf{1}[\tilde{\pi}_i(s_{t-w+1}) = a] + W^* \sum_{w=0}^{W'-1} \mathbf{1}[\tilde{\pi}_i(s_{t-w+1}) = a''] \Bigg) \\
& - \sum_{i \in B} \Bigg(W' \sum_{w=0}^{W^*-1} \mathbf{1}[\pi_i(s_{t-w+1}) = a'] - W' \sum_{w=0}^{W^*-1} \mathbf{1}[\pi_i(s_{t-w+1}) = a^\#] \\
& \qquad - W^* \sum_{w=0}^{W'-1} \mathbf{1}[\pi_i(s_{t-w+1}) = a] + W^* \sum_{w=0}^{W'-1} \mathbf{1}[\pi_i(s_{t-w+1}) = a''] \Bigg) \\
=& W' \left(n_{a'}(s_{t-W^*+1:t}) - n_{a^\#}(s_{t-W^*+1:t}) \right) - W^* \left(n_a(s_{t-W'+1:t}) - n_{a''}(s_{t-W'+1:t}) \right) - \mathbf{1}[a' > a] \\
& + \sum_{i \in B} \sum_{w=0}^{\max\{W^*,W'\}} \sigma^w(\tilde{\pi}_i(s_{t-w})) - \sigma^w(\pi_i(s_{t-w})) \\
\leq & W' \left(n_{a'}(s_{t-W^*+1:t}) - n_{a^\#}(s_{t-W^*+1:t}) \right) - W^* \left(n_a(s_{t-W'+1:t}) - n_{a''}(s_{t-W'+1:t}) \right) - \mathbf{1}[a' > a] \\
& + \sum_{i \in B} \underbrace{\sum_{w=0}^{\max\{W^*,W'\}} \max_{a_0 \in \mathcal{A}} \sigma^w(a_0) - \sigma^w(\pi_i(s_{t-w}))}_{g_i} \\
\overset{(a)}{\leq} & W' \left(n_{a'}(s_{t-W^*+1:t}) - n_{a^\#}(s_{t-W^*+1:t}) \right) - W^* \left(n_a(s_{t-W'+1:t}) - n_{a''}(s_{t-W'+1:t}) \right) - \mathbf{1}[a' > a] + \sum_{i=1}^{K} g^{(i)} \\
\overset{(b)}{\leq} & W' \left(n_{a'}(s_{t-W^*+1:t}) - n_{a^\#}(s_{t-W^*+1:t}) \right) - W^* \left(n_a(s_{t-W'+1:t}) - n_{a''}(s_{t-W'+1:t}) \right) - \mathbf{1}[a' > a] + \sum_{i=1}^{L_{a',a''}^{W^*,W'}} g^{(i)} \\
\overset{(c)}{<} & 0.
\end{aligned}$$

Thus, Equation (27) is proved. Therefore, $K > L_{a',a''}^{W^*,W'}$ is a necessary condition for $Q(W^*, a')$, *i.e.*, Equation (25).

In the above deriviation, the definitions of g^i, g^i, and σ^w are from Equation (7). (a) comes from the facts that $\{g^{(i)}\}_{i=1}^{u}$ is a nondecreasing permutation of $\{g_i\}_{i=0}^{u-1}$, $g_i \geq 0$, and $|B| \leq K$. (b) comes from the assumption $K \leq L_{a,a''}^{W^*,W'}$ and also $g^{(i)} \geq 0$. (c) comes from the definition in Equation (6). □

F.6 Possible Action Set in PARL and Comparison

F.6.1 Certification

Proof of Theorem 4. According to the definition of possible action set, we only need to prove the contrary: for any $a \in \mathcal{A} \setminus A^T(K)$, within poisoning size K, the poisoned policy cannot choose a: $\widetilde{\pi_{\mathrm{P}}}(s_t) \neq a$.

According to Equation (8), any $a \in \mathcal{A} \setminus A^T(K)$ satisfies

$$\sum_{a' \in \mathcal{A}} \max\{n_{a'}(s_t) - n_a(s_t) - K + \mathbf{1}[a' < a], 0\} > K. \tag{28}$$

Given poisoning size K, since each poisoning size can affect only one subpolicy, we know

$$\widetilde{n}_a(s_t) \leq n_a(s_t) + K$$

where $\widetilde{n}_a$ denotes to the poisoned aggregated action count. We suppose the attack could be successful, then for $a' < a$, $\widetilde{n}_{a'}(s_t) \leq \widetilde{n}_a(s_t) - 1$, and thus $n_{a'}(s_t) - \widetilde{n}_{a'}(s_t) \geq n_{a'}(s_t) - n_a(s_t) - K + 1$. Similarly, for $a' > a$, $\widetilde{a'}(s_t) \leq \widetilde{n}_a(s_t)$, and thus $n_{a'}(s_t) - \widetilde{n}_{a'}(s_t) \geq n_{a'}(s_t) - n_a(s_t) - K$. Also, for any $a' \neq a$ after poisoning $n_{a'}(s_t) - \widetilde{n}_{a'}(s_t) \geq 0$; otherwise deviating the difference subpolicies' decisions' from a' to a is strictly no-worse. Given these facts, the amount of votes that need to be reduced is the LHS of Equation (28) which is larger than K. However, we only have K poisoning size, *i.e.*, K votes that can be reduced. As a result, our assumption that the attack could be successful is falsified and the poisoned policy cannot choose a. □

Corollary 12 (Loose PARL Action Set)**.** *Under the condition of Definition 5, suppose the aggregation protocol is PARL as defined in Definition 7, then the* possible action set *at step t*

$$A^L(K) = \left\{ a \in \mathcal{A} \,\middle|\, \max_{a' \in \mathcal{A}} n_{a'}(s_t) - n_a(s_t) \leq 2K - \mathbf{1}[a > \arg\max_{a' \in \mathcal{A}} n_{a'}(s_t)] \right\}. \tag{29}$$

Proof of Corollary 12. Again, we prove the contrary, for any $a \in \mathcal{A} \setminus A^L(K)$, within poisoning size K, the poisoned policy cannot choose a: $\widetilde{\pi_{\mathrm{P}}}(s_t) \neq a$.

According to Equation (29), let $a_m = \arg\max_{a' \in \mathcal{A}} n_{a'}(s_t)$, then any $a \in \mathcal{A} \setminus A^L(K)$ satisfies

$$n_{a_m}(s_t) - n_a(s_t) > 2K - \mathbf{1}[a > a_m].$$

After poisoning, we thus have

$$n_{a_m}(s_t) - n_a(s_t) > -\mathbf{1}[a > a_m] \Longrightarrow n_{a_m}(s_t) - n_a(s_t) \geq 1 - \mathbf{1}[a > a_m]$$

From the definition, $a \notin A^L(K)$ so $a \neq a_m$. If $a_m < a$, $n_{a_m}(s_t) \geq n_a(s_t)$; if $a_m > a$, $n_{a_m}(s_t) > n_a(s_t)$. In both cases, a_m has higher priority to be chosen than a, and thus the poisoned policy cannot choose a. □

F.6.2 Comparison

Theorem 13. *Under the condition of Definition 5, suppose the aggregation protocol is PARL as defined in Definition 7, $A^T(K), A^L(K)$ are defined according to Equations* (8) *and* (29) *accordingly, then*

1. *$A^T(K) \subseteq A^L(K)$; and there are subpolicies $\{\pi_i\}_{i=0}^{u-1}$ and state s_t such that $A^T(K) \subsetneq A^L(K)$.*
2. *Given subpolicies $\{\pi_i\}_{i=0}^{u-1}$ and state s_t, for any $a \in A^T(K)$, there exists a poisoned training set $\widetilde{D}$ whose poisoning size $|D \ominus \widetilde{D}| \leq K$ and some RL training mechanism, such that $\widetilde{\pi_{\mathrm{P}}}(s_t) = a$ where $\widetilde{\pi_{\mathrm{P}}}$ is the poisoned PARL policy trained on $\widetilde{D}$.*

Proof of Theorem 13. We prove the two arguments separately.

1. We first prove $A^T(K) \subseteq A^L(K)$. For any $a \in A^T(K)$, let $a_m = \arg\max_{a' \in \mathcal{A}} n_{a'}(s_t)$, then
$$\begin{aligned}&\sum_{a' \in \mathcal{A}} \max\{n_{a'}(s_t) - n_a(s_t) - K + \mathbf{1}[a' < a], 0\} \le K\\ \Longrightarrow &n_{a_m}(s_t) - n_a(s_t) - K + \mathbf{1}[a_m < a] \le K\\ \Longrightarrow &n_{a_m}(s_t) - n_a(s_t) \le 2K - \mathbf{1}[a > a_m]\end{aligned}$$
where the last proposition is exactly the set selector of $A^L(K)$ so $a \in A^L(K)$.

   We then prove that $A^T(K) \subsetneq A^L(K)$ can happen by construction. Suppose that there are three actions in the action space: $\mathcal{A} = \{a_1, a_2, a_3\}$. We construct subpolicies for current state s_t such that the aggregated action counts are
$$n_{a_1}(s_t) = 10, n_{a_2}(s_t) = 9, n_{a_3}(s_t) = 1.$$
Given poisoning size $K = 5$, we find that
$$\begin{aligned}&n_{a_1}(s_t) - n_{a_3}(s_t) = 9 \le 10 - \mathbf{1}[a_3 \ge a_1] = 9 && \Longrightarrow a_3 \in a^L(K),\\ &(n_{a_1}(s_t) - n_{a_3}(s_t) - K + \mathbf{1}[a_1 < a_3]) + \\ &(n_{a_2}(s_t) - n_{a_3}(s_t) - K + \mathbf{1}[a_2 < a_3]) = 9 > K = 5 && \Longrightarrow a_3 \notin a^T(K).\end{aligned}$$
Therefore, $A^T(K) \subsetneq A^L(K)$ for these subpolicies and state s_t.

2. We prove by construction. For any $a \in A^T(K)$, we construct the set of subpolicies to poison $B^a \subseteq [u]$ such that $|B^a| \le K$, then describe the corresponding poisoned dataset $\widetilde{D}^a$, and finally prove that the poisoned policy $\widetilde{\pi_{\mathrm{P}}}^a(s_t) = a$.

   For $a \in A^T(K)$, by definition (Equation (8)), we know
$$\sum_{a' \in \mathcal{A}} \max\{\underbrace{n_{a'}(s_t) - n_a(s_t) - K + \mathbf{1}[a' < a]}_{:= t_{a,a'}}, 0\} \le K. \tag{30}$$
We now define a set of actions $C^a \subseteq \mathcal{A}$ such that
$$C^a = \{a' \in \mathcal{A} \mid t_{a,a'} > 0\}.$$
According to this definition, $a \notin C^a$ since $t_{a,a} \le 0$.

   Fact F.1. *For $a \in A^T(K)$ and any $a' \in \mathcal{A}$, $n_a(s_t) + K - \mathbf{1}[a' < a] \ge 0$.*

   Proof of Fact F.1. Suppose $n_a(s_t) + K - \mathbf{1}[a' < a] \le 0$, then $n_a(s_t) = K = 0$ and $a' < a$. Then
$$\sum_{a' \in \mathcal{A}} \max\{n_{a'}(s_t) - n_a(s_t) - K + \mathbf{1}[a' < a], 0\} \ge \sum_{a' \in \mathcal{A}} n_{a'}(s_t) - n_a(s_t) - K = \sum_{a' \in \mathcal{A}} n_{a'}(s_t) = u > 0$$
which contradicts the requirement that the LHS of the above inequality should be $\le K = 0$. □

   Give Fact F.1, for any $a' \in C^a$, $n_{a'}(s_t) \ge t_{a,a'}$. Notice that $n_{a'}(s_t)$ is the number of subpolicies that vote for action a' at state s_t. Therefore, we can pick an arbitrary subset of those subpolicies whose cardinality is $t_{a,a'}$. We denote $B^a_{a'}$ to such subset:
$$B^a_{a'} \subseteq \{i \in [u] \mid \pi_i(s_t) = a'\}, |B^a_{a'}| = t_{a,a'}.$$
Now define B^a_α: $B^a_\alpha = \bigcup_{a' \in C^a} B^a_{a'}$. We construct B^a_β to be an arbitrary subset of those subpolicies whose prediction is not a and who are not in B^a_α, and limit the B^a_β's cardinality:
$$B^a_\beta \subseteq \{i \in [u] \mid \pi_i(s_t) \ne a\} \setminus B^a_\alpha, |B^a_\beta| = \min\{K - |B^a_\alpha|, u - n_a(s_t) - |B^a_\alpha|\}.$$
Such B^a_β can be selected, because:

   - From definition, $B^a_\alpha \subseteq \{i \in [u] \mid \pi_i(s_t) \ne a\}$, where the cardinality of $\{i \in [u] \mid \pi_i(s_t) \ne a\}$ is $u - n_a(s_t)$. So $0 \le u - n_a(s_t) - |B^a_\alpha|$.
   - Since
$$|B^a_\alpha| \le \sum_{a' \in C^a} |B^a_{a'}| = \sum_{a' \in C^a} t_{a,a'} = \sum_{a' \in \mathcal{A}, t_{a,a'} > 0} t_{a,a'} \overset{(*)}{\le} K,$$
$K - |B^a_\alpha| \ge 0$, and thus $|B^a_\beta| \ge 0$. Here $(*)$ is due to Equation (30).
   - The superset $\{i \in [u] \mid \pi_i(s_t) \ne a\} \setminus B^a_\alpha$ has cardinality $u - n_a(s_t) - |B^a_\alpha|$ and $|B^a_\beta| \le u - n_a(s_t) - |B^a_\alpha|$.

   To this point, we can define the set of subpolicies to poison:
$$B^a := B^a_\alpha \cup B^a_\beta,$$

and $|B^a| = |B^a_\alpha| + |B^a_\beta| \leq K$.

In a similar fashion as the attack construction in proof of Proposition 9, for each $i \in B^a$, we locate its corresponding partitioned dataset D_i for training subpolicy π^a_i. We inset one trajectory p^a_i to D_i such that our chosen learning algorithm $\mathcal{M}_0$ can train a subpolicy $\tilde{\pi}^a_i = \mathcal{M}_0(D_i \cup \{p^a_i\})$ such that $\tilde{\pi}^a_i(s_t) = a$. For example, the trajectory could be $p^a_i = \{(s_t, a, s', \infty)\}$ where s' is an arbitrary state that guarantees p^a_i is hashed to partition i; and $\mathcal{M}_0$ learns the action with maximum reward for the memorized nearest state. Then, the poisoned dataset

$$\widetilde{D}^a = \left(\bigcup_{i \in B^a} D_i \cup \{p^a_i\} \right) \cup \left(\bigcup_{i \in [u] \setminus B'} D_i \right).$$

Thus, $\widetilde{D}^a$ is $|D \ominus \widetilde{D}^a| = |B^a| \leq K$, *i.e.*, the constructed attack's poisoning size is within K.

We now analyze the action prediction of the poisoned policy $\widetilde{\pi_{\mathrm{P}}}$. For action a, after poisoning, $\widetilde{n}_a(s_t) = n_a(s_t) + |B^a| = n_a(s_t) + \min\{K, u - n_a(s_t)\} = \min\{n_a(s_t) + K, u\}$. If $\widetilde{n}_a(s_t) = u$, then all subpolicies vote for a, apparently $\widetilde{\pi_{\mathrm{P}}}^a(s_t) = a$; otherwise, $\widetilde{n}_a(s_t) = n_a(s_t) + K$. In this case, for any action $a' \in C^a$, since we at least choose subpolicies in $B^a_{a'}$ and change their action prediction to a, the aggregated action count after poisoning is $\widetilde{n}_{a'}(s_t) \leq n_{a'}(s_t) - |B^a_{a'}| = n_{a'}(s_t) - t_{a,a'} = n_{a'}(s_t) - n_{a'}(s_t) + n_a(s_t) + K - \mathbf{1}[a' < a] = n_a(s_t) + K - \mathbf{1}[a' < a] = \tilde{n}_a(s_t) - \mathbf{1}[a' < a]$. Thus, a' has lower priority to be chosen than a. For any action $a' \notin C^a$ and $a' \neq a$, by the definition of C^a, $t_{a,a'} = n_{a'}(s_t) - n_a(s_t) - K + \mathbf{1}[a' < a] \leq 0$. After poisoning, the vote of a' does not increase, *i.e.*, $\widetilde{n}_{a'}(s_t) \leq n_{a'}(s_t) \leq n_a(s_t) + K - \mathbf{1}[a' < a] = \widetilde{n}_a(s_t) - \mathbf{1}[a' < a]$. Thus, a' also has lower priority to be chosen than a. In conclusion, we have $\widetilde{\pi_{\mathrm{P}}}^a(s_t) = a$.

To this point, for any $a \in A^T(K)$, we successfully construct the corresponding poisoned dataset $\widetilde{D}^a$ within poisoning size K such that $\widetilde{\pi_{\mathrm{P}}}^a(s_t) = a$, thus concludes the proof.

□

F.7 Possible Action Set in TPARL

Proof of Theorem 5. For ease of notation, we let $w = \min\{W, t+1\}$, so w is the actual window size used at step t. We let $t_0 = t - w + 1$, *i.e.*, t_0 is the actual start time step for TPARL aggregation at our current step t. Now we can write chosen action at step t without poisoning as $\pi_{\mathrm{T}}(s_{t_0:t})$.

We only need to prove the contrary: for any $a \in \mathcal{A} \setminus A(K)$, within poisoning size K, the poisoned policy cannot choose a: $\widetilde{\pi_{\mathrm{T}}}(s_{t_0:t}) \neq a$.

According to Equation (9), for such a, there exists $a' \neq a$ such that

$$\sum_{i=1}^{K} h^{(i)}_{a',a} \leq \delta_{a',a} = n_{a'}(s_{t_0:t}) - n_a(s_{t_0:t}) - \mathbf{1}[a < a']. \tag{31}$$

Suppose there exists such poisoning attack that lets $\widetilde{\pi_{\mathrm{T}}}(s_{t_0:t}) = a$. This implies that for a', after poisoning, we have

$$\widetilde{n}_{a'}(s_{t_0:t}) - \widetilde{n}_{a'}(s_{t_0:t}) - \mathbf{1}[a' < a] < 0, \tag{32}$$

where $\widetilde{n}_a$ is the aggregated action count after poisoning. Since the poisoning size is within K, it can affect at most K subpolicies. We let $B \subseteq [u], |B| \leq K$ to represent the affected subpolicy set. Therefore,

$$\begin{aligned}
&\widetilde{n}_{a'}(s_{t_0:t}) - \widetilde{n}_a(s_{t_0:t}) - \mathbf{1}[a' < a] \\
=&\underbrace{n_{a'}(s_{t_0:t}) - n_a(s_{t_0:t}) - \mathbf{1}[a' < a]}_{\delta_{a',a}} + (\widetilde{n}_{a'}(s_{t_0:t}) - n_{a'}(s_{t_0:t})) - (\widetilde{n}_a(s_{t_0:t}) - n_a(s_{t_0:t})) \\
=&\delta_{a',a} + \sum_{i \in B} \left(\sum_{j=0}^{w-1} \mathbf{1}[\tilde{\pi}_i(s_{t-j}) = a', \pi_i(s_{t-j}) \neq a'] - \sum_{j=0}^{w-1} \mathbf{1}[\tilde{\pi}_i(s_{t-j}) \neq a', \pi_i(s_{t-j}) = a'] \right) \\
&- \sum_{i \in B} \left(\sum_{j=0}^{w-1} \mathbf{1}[\tilde{\pi}_i(s_{t-j}) = a, \pi_i(s_{t-j}) \neq a] - \sum_{j=0}^{w-1} \mathbf{1}[\tilde{\pi}_i(s_{t-j}) \neq a, \pi_i(s_{t-j}) = a] \right)
\end{aligned}$$

$$
\begin{aligned}
&\geq \delta_{a',a} - \sum_{i\in B}\left(\sum_{j=0}^{w-1}\mathbf{1}[\tilde{\pi}_i(s_{t-j})\neq a', \pi_i(s_{t-j})=a'] + \sum_{j=0}^{w-1}\mathbf{1}[\tilde{\pi}_i(s_{t-j})=a, \pi_i(s_{t-j})\neq a]\right)\\
&\geq \delta_{a',a} - \sum_{i\in B}\left(\sum_{j=0}^{w-1}\mathbf{1}[\pi_i(s_{t-j})=a'] + \sum_{j=0}^{w-1}\mathbf{1}[\pi_i(s_{t-j})\neq a]\right)\\
&= \delta_{a',a} - \sum_{i\in B}\left(\sum_{j=0}^{w-1}\mathbf{1}_{i,a'}(s_{t-j}) + w - \sum_{j=0}^{w-1}\mathbf{1}_{i,a}(s_{t-j})\right)\\
&= \delta_{a',a} - \sum_{i\in B} h_{i,a',a} \overset{(a)}{\geq} \delta_{a',a} - \sum_{i=1}^{|B|} h^{(i)}_{a',a} \overset{(b)}{\geq} \delta_{a',a} - \sum_{i=1}^{K} h^{(i)}_{a',a} \overset{(c)}{\geq} 0.
\end{aligned}
$$

This contradicts with Equation (32), and thus the assumption is falsified, *i.e.*, there is no such poisoning attack that let $\widetilde{\pi_{\mathrm{T}}}(s_{t_0:t}) = a$. In the above equations, (a) is due to the fact that $h^{(i)}_{a',a}$ is a nonincreasing permutation of $\{h_{i,a',a}\}_{i=0}^{u-1}$. (b) is due to the facts that $|B| \leq K$ and $h^{(i)}_{a',a} \geq 0$. (c) comes from Equation (31). □

F.8 Hardness for Computing Tight Possible Action Set in TPARL

Proof of Theorem 6. By Definition 5, the possible action set with minimum cardinality (called minimal possible action set hereinafter) is unique. Otherwise, suppose A and B are both minimal possible action set, but $A \neq B$, then $A \cap B$ is a smaller and valid possible action set. Therefore, the oracle that returns the minimal possible action set, denoted by MINSET, can tell whether any action $a \in$ MINSET and thus whether any action a can be chosen by some poisoned policy $\widetilde{\pi}$ whose poisoning size is within K. In other words, the problem of determining whether an action a can be chosen by some poisoned policy $\tilde{\pi}$ whose poisoning size is within K, denoted by ATTKACT, is polynomially equivalent to MINSET: MINSET $\equiv_P$ ATTKACT. Now, we show a polynomial reduction from the set cover problem to ATTKACT, which implies that our MINSET problem is an NP-complete problem.

The decision version of the set cover problem (Karp, 1972), denoted by SETCOVER, is a well-known NP-complete problem and is defined as follows. The inputs are

1. a universal set of n elements: $\mathcal{U} = \{u_1, u_2, \cdots, u_n\}$;
2. a set of subsets of $\mathcal{U}$: $\mathcal{V} = \{V_1, \cdots, V_m\}, V_i \subseteq \mathcal{U}, 1 \leq i \leq m, \bigcup_{i=1}^m V_i = \mathcal{U}$;
3. a positive number $K \in \mathbb{R}_+$.

The output is a boolean variable b, indicating that whether there exists a subset $\mathcal{W} \subseteq \mathcal{V}, |\mathcal{W}| \leq K$, such that $\forall u_i \in \mathcal{U}, \exists V \in \mathcal{W}, u_i \in V$. Given an oracle to ATTKACT, we need to show SETCOVER can be solved in polynomial time, *i.e.*, SETCOVER $\leq_P$ ATTKACT.

- If $K \geq n$:
  We scan all sets $V_i \in \mathcal{V}$. To being with, we have a record set $S \leftarrow \emptyset$, and an answer set $\mathcal{W} \leftarrow \emptyset$. Whenever we encounter a set that contains a new element $u_i \notin S$, we put $\mathcal{W} \leftarrow \mathcal{W} \cup \{V_j\}$, and record this element $S \leftarrow S \cup \{u_i\}$. After one scan pass, $\mathcal{W}$ covers all elements of $\mathcal{U}$ (since $\bigcup_{j=1}^m V_j = \mathcal{U}$), and $|\mathcal{W}| \leq n \leq K$. Therefore, $\mathcal{W}$ is a valid set cover. Since we can always find such $\mathcal{W}$, we can directly answer `true`.
- If $K \geq m$:
  We can directly return $\mathcal{V}$ as a valid set cover, since $\bigcup_{j=1}^m V_j = \mathcal{U}$ and $|\mathcal{V}| \leq m \leq K$. Thus, we can answer `true`.
- If $K < \min\{n, m\}$:
  This is the general case which we need to handle. Now we construct the ATTKACT(K) problem so that we can trigger its oracle to solve SETCOVER.

1. The poisoning size is K.
2. The action space $\mathcal{A} = \mathcal{U} \cup \{b\} \cup \Gamma$ where $\Gamma := \{\#_1, \dots, \#_{m^2n}\}$. ($|\mathcal{A}| = n + 1 + m^2n$.) The sorting of actions is $u_1 < u_2 < \cdots < u_n < b < \#_1 < \cdots < \#_{m^2n}$.
3. The subpolicies are $\{\pi_j^a\}_{j=1}^m \cup \{\pi_i^b, \pi_{i,j}^c \mid 1 \le i \le n, 1 \le j \le K-1\}$, where π_j^a corresponds to $V_j \in \mathcal{V}$, and $\pi_i^b, \pi_{i,j}^c$ correspond to $u_i \in \mathcal{U}$. (Number of subpolicies $u = m + Kn \le m + n^2$.)
4. The current time step is $t = nm$, and the window size $W = nm$.
5. The input action is b, *i.e.*, asking whether b can be chosen by some poisoned TPARL policy, *i.e.*, $\widetilde{\pi_{\mathrm{T}}}(s_{1:t}) = b$, if the poisoning size is within K.

Now, we construct the states at each step t ($1 \le t \le nm$) so that the subpolicies' action predictions at these steps are as follows.

1. Count the appearing time of each u_i in $\mathcal{V}$, and denote it by c_i: $c_i = \sum_{j=1}^m \mathbf{1}[u_i \in V_j]$. For each u_i, select a V_{j_0} that contains u_i, and in the corresponding $\pi_{j_0}^a$, assign $m - c_i + 1$ steps to predict u_i; for all other V_j that contains u_i, in the corresponding π_j^a, assign one step to predict u_i.
2. After this process, each π_j^a at least has one time step whose action prediction is u_i for each $u_i \in V_j$. Among all $\{\pi_j^a\}_{j=1}^m$ and all time steps $1 \le t \le nm$, mn step-action cells are filled, and the remaining $(m^2n - mn)$ cells are filled by $\#_l \in \Gamma$ sequentially.
3. For each π_i^b, arbitrarily select $W - m$ time steps to assign action prediction as u_i; and fill in other m time steps by remaining $\#_l \in \Gamma$ sequentially.
4. For each $\pi_{i,j}^c$, for all time steps, let the action prediction be u_i.

As we can observe, the number of actions, the number of subpolicies, and the window size are all bounded by a polynomial of n and m. Therefore, such construction can be done in polynomial time.

We then show $\mathsf{SETCOVER} = \texttt{true} \iff \exists K', b \in \mathsf{ATTKACT}(K'), 1 \le K' \le K$.

- $\Longrightarrow$:
  Suppose the covering set is $\mathcal{W} \subseteq \mathcal{V}$, we denote K' to $|\mathcal{W}|$, and construct the $\mathsf{ATTKACT}$ problem with poisoning size K' as described above.
  We can construct a poisoning strategy to let $\widetilde{\pi_{\mathrm{T}}}(s_{1:nm}) = b$, The poisoning strategy is to find out π_j^a for each $V_j \in \mathcal{W}$, and to let them predict action b throughout all time steps: $\widetilde{\pi}_j^a(s'_t) = b, 1 \le t' \le nm$.
  After poisoning, the aggregated action count $\widetilde{n}_b(s_{1:nm}) = |\mathcal{W}| \times nm = K'nm$. Since $\mathcal{W}$ covers every $u_i \in \mathcal{U}$, for each $u_i \in \mathcal{U}$ there exists a set $V_j \ni u_i$, whose corresponding $\widetilde{\pi}_j^a$ is poisoned to predict b. Thus, $\widetilde{n}_{u_i}(s_{1:nm}) < n_{u_i}(s_{1:nm}) = m + (W - m) + W \times (K' - 1) = WK' = K'nm$. For any $\#_l \in \Gamma$, $\widetilde{n}_{\#_l}(s_{1:nm}) \le n_{\#_l}(s_{1:nm}) = 1$. In summary,
  $$\widetilde{n}_{u_i}(s_{1:nm}) < K'nm, \widetilde{n}_b(s_{1:nm}) = K'nm, \widetilde{n}_{\#_l}(s_{1:nm}) = 1.$$
  Thus, after TPARL aggregation, the poisoned policy $\widetilde{\pi_{\mathrm{T}}}(s_{1:nm}) = b$, and therefore $b \in \mathsf{ATTKACT}(K')$.
- $\Longleftarrow$:
  Suppose it is K' that let $b \in \mathsf{ATTKACT}(K')$, which implies that there exists such a poisoning attack within size K' that misleads the poisoned policy to b: $\widetilde{\pi_{\mathrm{T}}}(s_{1:nm}) = b$. Since the poisoning size is K', after poisoning the aggregated action count
  $$\widetilde{n}_b(s_{1:nm}) \le K'W = K'nm.$$
  For each $u_i \in \mathcal{U}$, since $n_{u_i}(s_{1:nm}) = m + (W - m) + W \times (K' - 1) = K'nm$, we always have
  $$\widetilde{n}_{u_i}(s_{1:nm}) \overset{(*)}{<} \widetilde{n}_b(s_{1:nm}) \le n_{u_i}(s_{1:nm}), \tag{33}$$
  where $(*)$ is due to the condition of successful attack. We denote the set of poisoned subpolicies by Π ($|\Pi| \le K'$). Therefore, Equation (33) implies that for each $u_i \in \mathcal{U}$, there exists at least one subpolicy
  $$\pi'_{u_i} \in \Pi, \pi'_{u_i} \in \{\pi_j^a \mid u_i \in V_j\} \cup \{\pi_i^b\} \cup \{\pi_{i,j}^c \mid 1 \le j \le K-1\} \tag{34}$$
  that is poisoned by the attack, otherwise the aggregated vote $\widetilde{n}_{u_i}$ cannot change.
  We partition Γ by Γ^a and Γ^{bc}, where
  $$\Gamma^a = \Gamma \cap \{\pi_j^a\}_{j=1}^m, \Gamma^b = \Gamma \cap \{\pi_i^b, \pi_{i,j}^c \mid 1 \le i \le n, 1 \le j \le K-1\}.$$

We construct additional poisoning set Γ^a_+ following this process: In the beginning, $\Gamma^a_+ \leftarrow \emptyset$. For each $u_i \in \mathcal{U}$, if $\Gamma^a \cap \{\pi^a_j \mid u_i \in V_j\}$ is not empty, skip. Otherwise, according to Equation (34), $\Gamma^b \cap \{\pi^b_i, \pi^c_{i,j} \mid 1 \le j \le K-1\}$ is not empty. In this case, we find an arbitrary covering set of u_i, namely $V_{j_0} \ni u_i$, and put $\pi^a_{j_0}$ into Γ^a_+. When the process terminates, we find that for each $u_i \in \mathcal{U}$,

$$(\Gamma^a \cup \Gamma^a_+) \cap \{\pi^a_j \mid u_i \in V_j\} \neq \emptyset. \tag{35}$$

Following the mapping $\{\pi^a_j\}_{j=1}^m \longleftrightarrow \{V_j \mid 1 \le j \le m\} = \mathcal{V}$, the subset $(\Gamma^a \cup \Gamma^a_+) \subseteq \{\pi^a_j\}_{j=1}^m$ can be mapped to $(\mathcal{W}^a \cup \mathcal{W}^a_+) \subseteq \mathcal{V}$. From Equation (35), $(\mathcal{W}^a \cup \mathcal{W}^a_+)$ is a valid set cover for $\mathcal{U}$. We now study the cardinality of this set cover. From the process, we know that every $V_{j_0} \in \mathcal{W}^a_+$ corresponds to a different set in Γ^b. Thus, $|\mathcal{W}^a_+| \le |\Gamma^b|$, which implies that

$$|\mathcal{W}^a \cup \mathcal{W}^a_+| \le |\mathcal{W}^a| + |\mathcal{W}^a_+| \le |\Gamma^a| + |\Gamma^b| = |\Gamma| \le K'.$$

To this point, we successfully construct a set cover within cardinally $K' \le K$ that covers $\mathcal{U}$, so $\mathsf{SETCOVER} = \texttt{true}$.

Therefore, we can check whether $\mathsf{SETCOVER} = \texttt{true}$ by iterating K' from 1 to K ($< \min\{n, m\}$ iterations), constructing the $\mathsf{ATTKACT}(K')$ problem, and querying the oracle. The whole process can be done in polynomial time assumping $O(1)$ computation time of the $\mathsf{ATTKACT}(K')$ oracle.

To this point, we have shown $\mathsf{SETCOVER} \le_P \mathsf{ATTKACT}$. On the other hand, an undeterminisitic Turing machine can try different poisoning strategies by branching on whether to poison current subpolicy and what actions to be assigned to each poisoned subpolicy. The decision of whether the poisoning is successful can be done in polynomial time and $b \in \mathsf{ATTKACT}(K)$ corresponds to the existence of successfully attacked branches. Thus, $\mathsf{ATTKACT} \in \mathsf{NP}$. Given that $\mathsf{SETCOVER}$ is an NP-complete problem, so does $\mathsf{ATTKACT}$ and MINSET (since $\mathsf{MINSET} \equiv_P \mathsf{ATTKACT}$). □

F.9 Possible Action Set in DPARL

Proof of Theorem 7. For ease of notation, we assume that $W_{\max} \le t+1$, and otherwise we let $W_{\max} \leftarrow t+1$. We let $t_0 = \max\{t - W_{\max} + 1, 0\}$ be the start time step of the maximum possible window.

We only need to prove the contrary: for any $a \in \mathcal{A} \setminus A(K)$, within poisoning size K, the poisoned policy cannot choose a: $\widetilde{\pi_{\mathrm{D}}}(s_{t_0:t}) \neq a$. We prove by contradiction: we assume that there exists such a poisoning attack within poisoning size K that lets $\widetilde{\pi_{\mathrm{D}}}(s_{t_0:t}) = a$. From the expression of $A(K)$ (Equation (10)), $a \neq a_t = \pi_{\mathrm{D}}(s_{t_0:t})$. Suppose the selected time window before the attack is W' (selected according to Equation (1) based on Δ^W_t and n_a), and the selected time window after the attack is $\widetilde{W}'$ (selected according to Equation (1) based on $\widetilde{\Delta}^W_t$ and $\widetilde{n}_a$).

- If $\widetilde{W}' = W'$:
  Suppose we use the TPARL aggregation policy with window size $W = W'$ instead of current DPARL aggregation policy, then we will have $\pi_{\mathrm{T}}(s_{t-W'+1:t}) = a_t$ and $\widetilde{\pi_{\mathrm{T}}}(s_{t-W'+1:t}) = a$. Thus, according to the definition of possible action set (Definition 5), $a \in A(K)$ where $A(K)$ is defined by Equation (9) in Theorem 5. This implies that $a \in A(K)$ where $A(K)$ is defined by Equation (10), which contradicts the assumption that $a \in \mathcal{A} \setminus A(K)$.

- If $\widetilde{W}' \neq W'$:
  According to the definition in Equation (10),
  $$\min_{1 \le W^* \le W_{\max}, W^* \neq W', a'' \neq a_t} L^{W^*, W'}_{a, a''} > K.$$
  We define $a^{\#}_t = \arg\max_{a_0 \neq a_t} \widetilde{n}_{a_0}(s_{t-W'+1:t})$. Then, the above equation implies that
  $$L^{\widetilde{W}', W'}_{a, a^{\#}_t} > K$$
  and thus
  $$\sum_{i=1}^{L^{\widetilde{W}', W'}_{a, a^{\#}_t}} g^{(i)} + W'(n^{\widetilde{W}'}_a - n^{\widetilde{W}'}_{a^{\#}}) - \widetilde{W}'(n^{W'}_{a_t} - n^{W'}_{a^{\#}_t}) - \mathbf{1}[a > a_t] < 0.$$

Since $g^{(i)} \geq 0$ by definition (Equation (7)),

$$\sum_{i=1}^{K} g^{(i)} + W'(n_a^{\widetilde{W}'} - n_{a^\#}^{\widetilde{W}'}) - \widetilde{W}'(n_{a_t}^{W'} - n_{a_t^\#}^{W'}) - \mathbf{1}[a > a_t] < 0, \tag{36}$$

where $a^\# = \arg\max_{a_0 \neq a', a_0 \in \mathcal{A}} n_{a_0}(s_{t-\widetilde{W}'+1:t})$ and n_a^w is a shorthand of $n_a(s_{t-w+1:t})$.

Following the derivation from Equation (27) to (a), we have

$$\sum_{i=1}^{K} g^{(i)} + W'(n_a^{\widetilde{W}'} - n_{a^\#}^{\widetilde{W}'}) - \widetilde{W}'(n_{a_t}^{W'} - n_{a_t^\#}^{W'}) - \mathbf{1}[a > a_t] \geq W'(\widetilde{n}_a^{\widetilde{W}'} - \widetilde{n}_{a^\#}^{\widetilde{W}'}) - \widetilde{W}'(\widetilde{n}_{a_t}^{W'} - \widetilde{n}_{a_t^\#}^{W'}) - \mathbf{1}[a > a_t].$$

Combined with Equation (36),

$$W'(\widetilde{n}_a^{\widetilde{W}'} - \widetilde{n}_{a^\#}^{\widetilde{W}'}) - \widetilde{W}'(\widetilde{n}_{a_t}^{W'} - \widetilde{n}_{a_t^\#}^{W'}) - \mathbf{1}[a > a_t] < 0. \tag{37}$$

On the other hand, the successful attack assumption, *i.e.*, $\widetilde{\pi_{\mathrm{D}}}(s_{t_0:t}) = a$ and $\pi_{\mathrm{D}}(s_{t_0:t}) = a_t$, implies that

$$\widetilde{\Delta}_t^{\widetilde{W}'} = \frac{\widetilde{n}_a^{\widetilde{W}'} - \widetilde{n}_{a^{(2)}}^{\widetilde{W}'}}{\widetilde{W}'} > \frac{\widetilde{n}_{a_{W'}}^{W'} - \widetilde{n}_{a_{W'}^{(2)}}^{W'}}{W'} = \widetilde{\Delta}_t^{W'}, \tag{38}$$

where "$>$" is "$\geq$" if $a < a_t$. In the above equation,

$$a^{(2)} = \underset{a_0 \neq a, a_0 \in \mathcal{A}}{\arg\max}\ \widetilde{n}_{a_0}^{\widetilde{W}'}, a_{W'} = \underset{a_0 \in \mathcal{A}}{\arg\max}\, \widetilde{n}_{a_0}^{W'}, a_{W'}^{(2)} = \underset{a_0 \neq a_{W'}, a_0 \in \mathcal{A}}{\arg\max}\ \widetilde{n}_{a_0}^{W'}.$$

Intuitively, after poisoning, $a^{(2)}$ is the runner-up action at window $\widetilde{W}'$, $a_{W'}$ is the action at window W', and $a_{W'}^{(2)}$ is the runner-up action at window W'. We rewrite Equation (38) to

$$W'(\widetilde{n}_a^{\widetilde{W}'} - \widetilde{n}_{a^{(2)}}^{\widetilde{W}'}) \geq \widetilde{W}'(\widetilde{n}_{a_{W'}}^{W'} - \widetilde{n}_{a_{W'}^{(2)}}^{W'}) + \mathbf{1}[a > a_t]. \tag{39}$$

We have the following two observations:

1. $\widetilde{n}_a^{\widetilde{W}'} - \widetilde{n}_{a^\#}^{\widetilde{W}'} \geq \widetilde{n}_a^{\widetilde{W}'} - \widetilde{n}_{a^{(2)}}^{\widetilde{W}'}$, since a is the top action and $a^{(2)}$ is the runner-up action and their margin should be the smallest.
2. $\widetilde{n}_{a_{W'}}^{W'} - \widetilde{n}_{a_{W'}^{(2)}}^{W'} \geq \widetilde{n}_{a_t}^{W'} - \widetilde{n}_{a_t^\#}^{W'}$, because 1) if $a_t = a_{W'}$, LHS equals to RHS; 2) if $a_t \neq a_{W'}$, LHS ≥ 0 and RHS ≤ 0.

Plugging these two observations to two sides of Equation (39), we get

$$W'(\widetilde{n}_a^{\widetilde{W}'} - \widetilde{n}_{a^\#}^{\widetilde{W}'}) \geq \widetilde{W}'(\widetilde{n}_{a_t}^{W'} - \widetilde{n}_{a_t^\#}^{W'}) + \mathbf{1}[a > a_t]. \tag{40}$$

This contradicts with Equation (37).

Since in both cases, we find contradictions. Now we can conclude that for any $a \in \mathcal{A} \setminus A(K)$, within poisoning size K, the poisoned policy cannot choose a: $\widetilde{\pi_{\mathrm{D}}}(s_{t_0:t}) \neq a$. □

G Additional Experimental Details

G.1 Details of the Offline RL algorithms and Implementations

We experimented with three offline RL algorithms: DQN (Mnih et al., 2013), QR-DQN (Dabney et al., 2018), and C51 (Bellemare et al., 2017). The first one is the standard baseline, while the latter two are distributional RL algorithms which show SOTA results in offline RL tasks. We first briefly introduce the algorithm ideas, followed by the implementation details.

Algorithm Ideas. The core of Q-learning (Watkins & Dayan, 1992) is the Bellman optimality equation (Bellman, 1966)

$$Q^\star(s,a) = \mathbb{E}R(s,a) + \gamma \mathbb{E}_{s' \sim P} \max_{a' \in \mathcal{A}} Q^\star(s', a'), \tag{41}$$

where a parameterized Q^θ is adopted to approximate the optimal $Q^\star$ and iteratively improved. In DQN (Mnih et al., 2013) specifically, the parameterization is achieved by using a convolutional neural network (LeCun et al., 1998). In contrast to estimating the mean action value $Q^\pi(s,a)$ in DQN, distributional RL algorithms estimate a density over the values of the state-action pairs. The distributional Bellman optimality can be expressed as follows:

$$Z^\star(s,a) \overset{D}{=} r + \gamma Z^\star(s', \mathrm{argmax}_{a' \in \mathcal{A}} Q^\star(s', a')) \text{ where } r \sim R(s,a), s' \sim P(\cdot \mid s, a). \tag{42}$$

Table 4: **Benign empirical performance** of three *aggregation protocols* (PARL, TPARL, and DPARL) applied on subpolicies trained using three *offline RL algorithms* (DQN, QR-DQN, and C51), with the number of subpolicies (*i.e.*, #partitions) u equal to 30 or 50. We report results averaged over 20 runs of varying randomness in the environment.

Freeway	$u = 30$					$u = 50$				
	PARL	TPARL ($W = 2$)	TPARL ($W = 3$)	TPARL ($W = 4$)	DPARL ($W_{\max} = 5$)	PARL	TPARL ($W = 2$)	TPARL ($W = 3$)	TPARL ($W = 4$)	DPARL ($W_{\max} = 5$)
DQN	10.90 $\pm$ 0.77	11.65 $\pm$ 0.57	11.25 $\pm$ 0.54	12.00 $\pm$ 0.45	11.45 $\pm$ 0.74	10.95 $\pm$ 0.97	11.60 $\pm$ 1.02	12.40 $\pm$ 0.58	11.65 $\pm$ 0.65	11.90 $\pm$ 0.89
QR-DQN	11.60 $\pm$ 0.66	11.85 $\pm$ 1.15	11.25 $\pm$ 0.62	11.80 $\pm$ 0.87	12.10 $\pm$ 0.99	11.50 $\pm$ 0.92	11.60 $\pm$ 1.02	12.15 $\pm$ 0.57	12.80 $\pm$ 0.51	11.90 $\pm$ 0.77
C51	11.20 $\pm$ 0.60	12.45 $\pm$ 0.50	12.55 $\pm$ 0.50	11.40 $\pm$ 0.66	12.40 $\pm$ 0.49	11.70 $\pm$ 1.27	11.80 $\pm$ 0.75	12.70 $\pm$ 0.46	11.50 $\pm$ 0.87	11.95 $\pm$ 0.92

Breakout	$u = 30$					$u = 50$				
	PARL	TPARL ($W = 2$)	TPARL ($W = 3$)	TPARL ($W = 4$)	DPARL ($W_{\max} = 5$)	PARL	TPARL ($W = 2$)	TPARL ($W = 3$)	TPARL ($W = 4$)	DPARL ($W_{\max} = 5$)
DQN	58.65 $\pm$ 40.83	38.05 $\pm$ 10.22	26.00 $\pm$ 12.79	13.50 $\pm$ 11.41	36.00 $\pm$ 13.39	60.25 $\pm$ 31.99	37.90 $\pm$ 11.55	25.90 $\pm$ 9.55	14.95 $\pm$ 7.24	45.00 $\pm$ 13.44
QR-DQN	76.50 $\pm$ 79.76	45.25 $\pm$ 42.70	22.05 $\pm$ 9.13	19.05 $\pm$ 6.91	42.05 $\pm$ 15.60	62.80 $\pm$ 28.71	32.10 $\pm$ 9.27	40.25 $\pm$ 52.97	17.30 $\pm$ 7.89	41.00 $\pm$ 13.87
C51	51.80 $\pm$ 10.74	37.60 $\pm$ 11.38	24.75 $\pm$ 12.77	13.55 $\pm$ 7.37	34.75 $\pm$ 10.91	60.55 $\pm$ 20.44	34.85 $\pm$ 11.92	26.15 $\pm$ 9.98	19.65 $\pm$ 7.30	39.90 $\pm$ 14.69

Concretely, QR-DQN (Dabney et al., 2018) approximates the density $D^\star$ with a uniform mixture of K Dirac delta functions, while C51 (Bellemare et al., 2017) approximates the density using a categorical distribution over a set of anchor points.

Implementation Details. For training the subpolicies using offline RL training algorithms, we use the code base of Agarwal et al. (2020). The configuration files containing the detailed hyperparameters can be found at their public repository `https://github.com/google-research/batch_rl/tree/master/batch_rl/fixed_replay/configs`. For the three methods DQN, QR-DQN, and C51, the names of the configuration files are `dqn.gin`, `quantile.gin`, and `c51.gin`, respectively. On each partition, we train the subpolicy for 50 epochs.

G.2 Concrete Experimental Procedures

We conduct experiments on two Atari 2600 games Freeway and Breakout from OpenAI Gym (Brockman et al., 2016) following Section 3. Concretely, in the **training** stage, we first partition the training dataset into u partitions ($u = 30$ or 50) by using the hash function $h(\tau)$ as the sum of all pixel values of all frames in the trajectory τ. We then train the subpolicies on the partitions with offline RL training algorithm $\mathcal{M}_0 \in \{$DQN (Mnih et al., 2013), QR-DQN (Dabney et al., 2018), C51 (Bellemare et al., 2017)$\}$. The detailed description of the algorithms can be found in Appendix G.1. In the **aggregation** stage, we apply the three proposed aggregation protocols (PARL, TPARL, and DPARL) on the trained u subpolicies and derive the aggregated policies π_{P}, π_{T}, and π_{D} accordingly. Finally, in the **certification** stage, for each aggregated policy, we provide the per-state action and cumulative reward certification following our theorems. When certifying the *per-state action*, we constrain the maximum trajectory length $H = 1000$ and report results averaged over 20 runs; for the *cumulative reward* certification, we adopt the trajectory length $H = 400$ for evaluating Freeway and $H = 75$ for Breakout.

H Additional Evaluation Results and Discussions

H.1 Benign Empirical Performance

We present the benign empirical cumulative rewards of the three aggregation protocols (PARL, TPARL, and DPARL) applied on subpolicies trained using three offline RL algorithms (DQN, QR-DQN, and C51) in Table 4. The cumulative reward is obtained by running a trajectory of maximum length H and accumulating the reward achieved at each time step.

We discuss the conclusions on multiple levels. We choose $H = 1000$ and report results averaged over 20 runs. On the **RL algorithm** level, we see that QR-DQN achieves the highest score in almost all cases. On the **aggregation protocol** level, temporal aggregation enhances the performance on Freeway while dampens the performance on Breakout. We particularly note that the results obtained by using temporal aggregation (TPARL and DPARL) gives significantly smaller variance compared

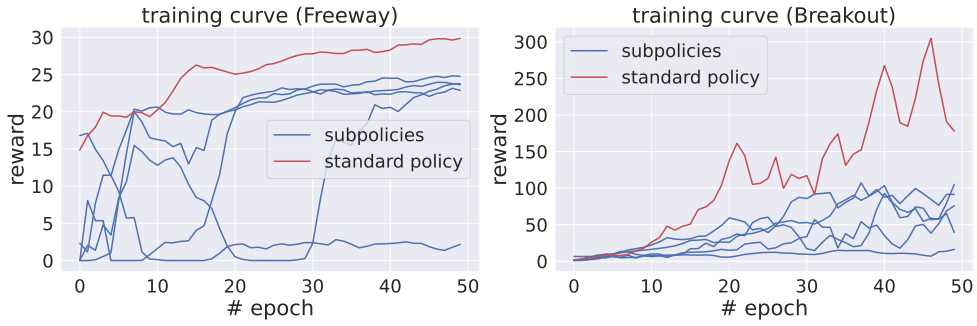


Figure 3: **The *convergence speed* of the proposed partition-based training compared with the standard training.** For *our proposed training*, we plot all training curves of the sampled 5 subpolicies in blue, where each subpolicy is trained on one partition of the dataset. (We do not plot all 50 curves for visual clarity). For *standard training*, we plot the training curve of the standard policy in red, where the single policy is trained on the entire dataset. We train each policy (or subpolicy) for a fixed number of 50 epochs, where each epoch consumes 1M randomly sampled training data. The offline RL algorithm used is DQN.

Table 5: **The *policy quality* measured by *empirical cumulative reward* of the proposed aggregation protocols (PARL, TPARL, and DPARL) compared with the standard training.** In our aggregation, we aggregate over u subpolicies with u equal to 30 or 50 . We report results averaged over 20 runs of varying randomness in the environment, where each run is an episode of length at most 1000. The offline RL algorithm used is DQN.

	standard testing	$u = 30$			$u = 50$		
		PARL	TPARL ($W = 4$)	DPARL ($W_{\max} = 5$)	PARL	TPARL ($W = 4$)	DPARL ($W_{\max} = 5$)
Freeway	12.00 $\pm$ 0.89	10.90 $\pm$ 0.77	12.00 $\pm$ 0.45	11.45 $\pm$ 0.74	10.95 $\pm$ 0.97	11.65 $\pm$ 0.65	11.90 $\pm$ 0.89
Breakout	97.60 $\pm$ 117.28	58.65 $\pm$ 40.83	13.50 $\pm$ 11.41	36.00 $\pm$ 13.39	60.25 $\pm$ 31.99	14.95 $\pm$ 7.24	45.00 $\pm$ 13.44

to the single-step aggregation PARL, especially in Breakout. This implies that temporal aggregation can help stabilize the policy. On the **partition number** level, a larger partition number can give slightly better results. On the game level, Freeway is simpler and more stable than Breakout.

H.2 Comparison of COPA with Standard Training

We provide additional experimental results on the comparison between the empirical performance of our COPA and the standard training in terms of the *convergence speed* (Figure 3) and *policy quality* (Table 5).

In Figure 3, we aim to show the comparison of the convergence speed. For *our proposed training*, we plot all training curves of the sampled 5 subpolicies in blue, where each subpolicy is trained on one partition of the dataset. (We do not plot all 50 curves for visual clarity). For *standard training*, we plot the training curve of the standard policy in red, where the single policy is trained on the entire dataset. We see that on **Freeway**, most subpolicies converge slower than the standard policy, but will reach similar convergence value as the standard training one; while there also exist a few subpolicies that fail to be trained. On **Breakout**, within 50 epochs, we observe substantial fluctuation for the training curves of all the policies, as well as large variance for the achieved reward at the last epoch. Thus, we cannot draw conclusions w.r.t. the convergence. Given more time and computational resources, we would be able to run for more epochs and draw more informative conclusions.

In Table 5, we aim to compare the policy quality of the aggregated policy derived in our COPA framework (*i.e.*, PARL, TPARL, and DPARL) with the policy obtained from standard training. We see that on **Freeway**, our three protocols achieve comparable results with the policy obtained by standard training on the entire dataset; on **Breakout**, although the quality of our obtained policies is lower, our policies are much more stable with significantly lower variance than the standard training.

Table 6: **Average window size** (*i.e.*, $\sum_{t=1}^{T} W_t/T$) for the *aggregation protocol* DPARL ($W_{\max} = 5$) applied on subpolicies trained using three *offline RL algorithms* (DQN, QR-DQN, and C51), with the number of subpolicies (*i.e.*, #partitions) u equal to 30 or 50. We report results averaged over all time steps in 20 runs.

	$u = 30$			$u = 50$		
	DQN	QR-DQN	C51	DQN	QR-DQN	C51
Freeway	2.69 $\pm$ 1.73	2.59 $\pm$ 1.71	2.64 $\pm$ 1.74	2.79 $\pm$ 1.73	2.70 $\pm$ 1.73	2.72 $\pm$ 1.73
Breakout	2.28 $\pm$ 1.46	2.44 $\pm$ 1.55	2.32 $\pm$ 1.48	2.33 $\pm$ 1.48	2.40 $\pm$ 1.52	2.39 $\pm$ 1.52

Table 7: **Runtime (unit: seconds)** of the *aggregation protocol* DPARL ($W_{\max} = 5$) applied on subpolicies trained using *offline RL algorithm* DQN, with the number of subpolicies (*i.e.*, partition number) u equal to 30 or 50. We compare with the *standard testing* which tests the runtime of a single trained DQN policy without using our framework. We report results averaged over 20 runs of varying randomness in the environment, where each run is an episode of length at most 1000.

	standard testing	DPARL ($u = 30$)	DPARL ($u = 50$)
Freeway	2.53 $\pm$ 0.50	56.37 $\pm$ 1.75	79.05 $\pm$ 4.27
Breakout	2.00 $\pm$ 0.56	72.05 $\pm$ 13.09	108.05 $\pm$ 8.71

H.3 More Analytical Statistics for DPARL

Selected Window Size. We provide the mean and variance for the selected window sizes in DPARL in Table 6. In our experiments, the *maximum window size* $W_{\max} = 5$, while the *average selected window sizes* are all around half of the maximum value. Thus, the average certification time is expected to be much smaller compared with the worst-case time complexity.

Running Time. We provide the running time in Table 7. Specifically, we compare the running time of DPARL ($W_{\max} = 5$) applied on $u = 30$ or 50 subpolicies trained using *offline RL algorithm* DQN with the *normal testing* which tests the runtime of a single trained DQN policy without using our framework. As we have shown in the remark of Theorem 3 in Section 3.3, the time complexity of DPARL is $O\left(W_{\max}^2|\mathcal{A}|^2 u + W_{\max}|\mathcal{A}|^2 u \log u\right)$. In Table 7, we also see that the runtime of DPARL scales roughly quasilinearly with the number of subpolicies u, and quadratically with the action set size $|\mathcal{A}|$, which is 3 for Freeway and 4 for Breakout.

H.4 Maximum Tolerable Poisoning Threshold vs. Total Trajectory Number

We provide the total number of trajectories in the offline dataset used for training the games Freeway and Breakout below, as well as the corresponding ratio of the maximum tolerable poisoning threshold w.r.t. the total number of trajectories. For **Freeway**, the total number of trajectories in the entire offline dataset is 121,892, and the ratio ($\max \overline{K}_t$ / # total trajectories) is 0.0002; for **Breakout**, the number is 209,049, and the ratio is 0.0001. We emphasize that, as shown in previous work on probable defense against poisoning in supervised learning (Levine & Feizi, 2020), the number of instances they can certify is also not high especially for challenging tasks (*e.g.*, certifying 20 instances on the dataset GTSRB in Levine & Feizi (2020), attaining the ratio 0.0005), which may imply the intrinsic difficulty of certifying against poisoning attacks. Given such difficulty, we already achieve reasonable certification as the first work on certified robust RL against poisoning, and we hope future works can further improve upon our results.

H.5 Comparison of Freeway and Breakout

Freeway and Breakout are two typical types of games with distinctive game properties. Freeway is a simple game where the agent aims to cross the road and avoid the traffic. In most of the time steps, the agent would take the "forward" action; only when there is a need to avoid the traffic will the agent stop and wait. In comparison, the game Breakout involves more complicated interactions between the paddle and the environment. The paddle position and velocity both play important roles

Table 8: **Average tolerable poisoning thresholds** of three *aggregation protocols* (PARL, TPARL, and DPARL) applied on subpolicies trained using three *offline RL algorithms* (DQN, QR-DQN, and C51), with the number of subpolicies (*i.e.*, #partitions) u equal to 30 or 50. We report results averaged over 20 runs of varying randomness in the environment.

Freeway	$u = 30$			$u = 50$		
	PARL	TPARL ($W = 4$)	DPARL ($W_{\max} = 5$)	PARL	TPARL ($W = 4$)	DPARL ($W_{\max} = 5$)
DQN	9.90 $\pm$ 0.14	10.21 $\pm$ 0.09	10.30 $\pm$ 0.09	16.78 $\pm$ 0.42	17.16 $\pm$ 0.29	16.88 $\pm$ 0.45
QR-DQN	9.87 $\pm$ 0.20	10.11 $\pm$ 0.30	10.16 $\pm$ 0.34	16.79 $\pm$ 0.52	17.31 $\pm$ 0.29	17.03 $\pm$ 0.40
C51	9.57 $\pm$ 0.24	9.83 $\pm$ 0.15	10.11 $\pm$ 0.19	16.82 $\pm$ 0.40	17.08 $\pm$ 0.34	17.61 $\pm$ 0.15

Breakout	$u = 30$			$u = 50$		
	PARL	TPARL ($W = 4$)	DPARL ($W_{\max} = 5$)	PARL	TPARL ($W = 4$)	DPARL ($W_{\max} = 5$)
DQN	1.08 $\pm$ 0.09	1.28 $\pm$ 0.15	0.85 $\pm$ 0.09	2.07 $\pm$ 0.22	1.92 $\pm$ 0.30	1.55 $\pm$ 0.28
QR-DQN	1.38 $\pm$ 0.09	1.33 $\pm$ 0.18	1.18 $\pm$ 0.10	2.12 $\pm$ 0.27	1.93 $\pm$ 0.14	1.71 $\pm$ 0.19
C51	1.10 $\pm$ 0.08	1.42 $\pm$ 0.19	0.93 $\pm$ 0.13	2.43 $\pm$ 0.21	2.37 $\pm$ 0.19	1.90 $\pm$ 0.19

in order to catch and bounce the ball at an appropriate angle, which requires a frequent switch of the paddle moving direction.

Given the properties of the games, we note that in Freeway, adjacent time steps may share consistent action selections, which is not necessarily true in the game Breakout. Thus, it would be expected to be beneficial to consider the past history for making the current decision in Freeway, while the past history may interfere with the action selection in Breakout.

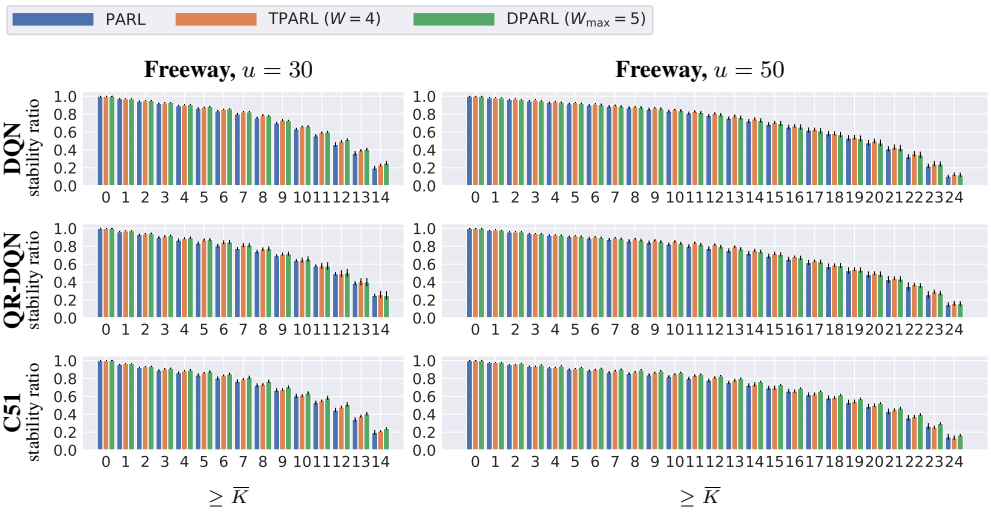
Bottleneck states in Breakout vs Freeway. In Breakout, the game goal is to control the paddle so that the ball is bounced to hit the brick. There are clearly very different stages in the Breakout game, *e.g.*, when the ball is flying towards the paddle and when it is bounced back. An example of the *bottleneck state* is when the ball is approaching the paddle. The poisoning behavior may lead to the disastrous effect that the policy learns to control the paddle to slide in the opposite direction of the ball at such bottleneck states, while other states may not be as vulnerable. In comparison, in Freeway, the game goal is to control the agent to cross the road while avoiding the traffic. The agent is faced with similar traffic conditions everywhere on the road, and can make stable choices regardless of the complex situation. Thus there are very few *bottleneck states*.

H.6 Full Results of Robustness Certification for Per-State Action Stability

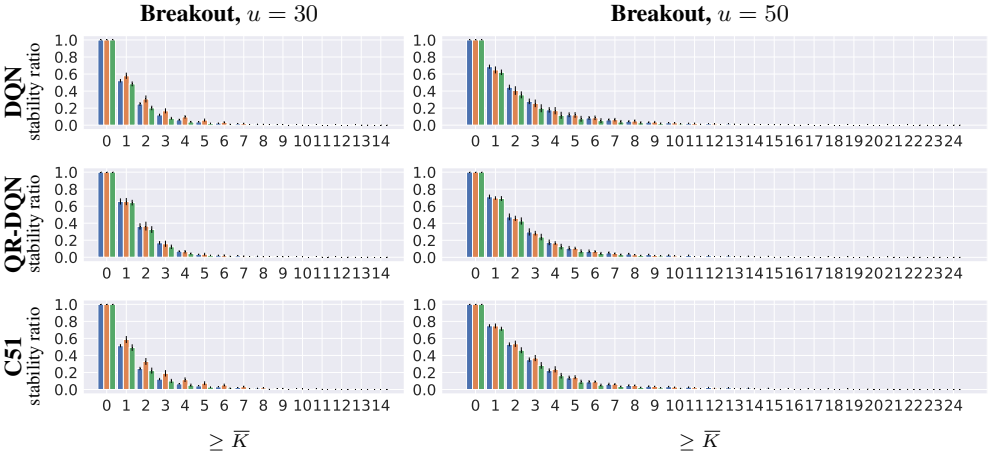
As a complete set of results for per-state action certification apart from Figure 1 in Section 4.1, we present the entire cumulative histogram of *tolerable poisoning thresholds* in Figure 4 and the *average tolerable poisoning thresholds* in Table 8. The definitions of the two metrcis can be found in Section 4.1. Basically, the conclusions in terms of the comparisons on the RL algorithm level, the aggregation protocol level, the partition number level, and the game level are all similar to that derived in Section 4.1.

H.7 Full Results of Robustness Certification for Cumulative Reward Bound

In addition to the evaluation results provided in Figure 2 in section 4.2, we provide a more comprehensive set of evaluation results with more settings of trajectory length H here in Figure 5. We draw similar conclusions as discussed in Section 4.2.



(a) Freeway



(b) Breakout

Figure 4: **Robustness certification for per-state action stability (full results).** We plot the cumulative histogram of the *tolerable poisoning size* $\overline{K}$ for all time steps in one trajectory. We provide results on two games (Freeway and Breakout), two partition numbers ($u = 30$ and $u = 50$), and a comparison of three certification methods (PARL, TPARL, DPARL). The results are averaged over 20 runs considering the randomness in the game environment, and the short vertical bar on top of each bar represents the standard deviation.

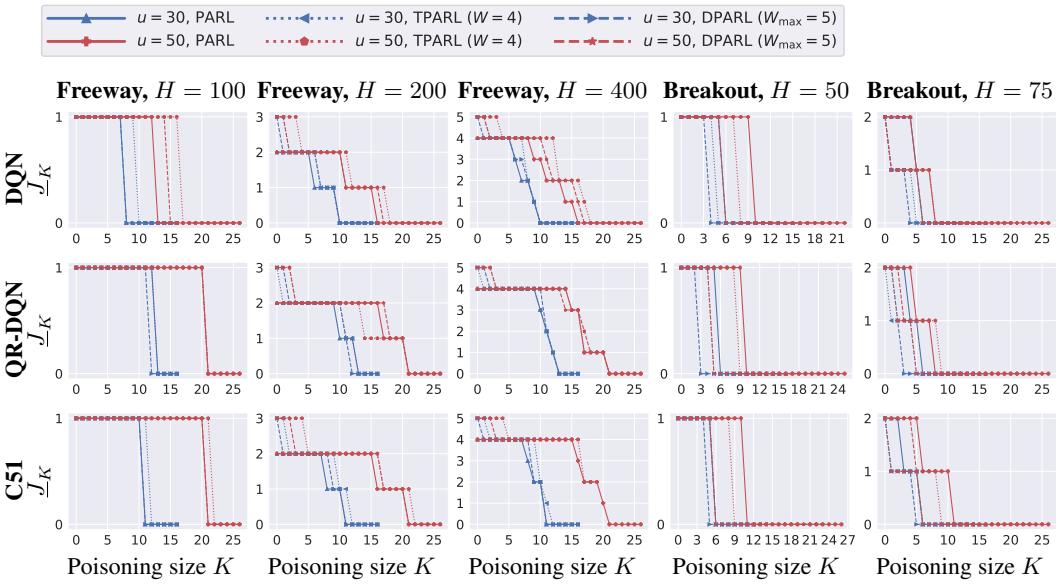


Figure 5: **Robustness certification for cumulative reward (full results).** We plot the *lower bound of cumulative reward* $\underline{J}_K$ w.r.t. *poisoning size* K under three different certification methods (PARL, TPARL ($W = 4$), DPARL ($W_{\max} = 5$)) with two partition numbers ($u \in \{30, 50\}$). Each row corresponds to one RL algorithm, and each column corresponds to one setting of trajectory length H.

I A Broader Discussion on Related Work

I.1 Poisoning Attacks in RL

Below, we provide a brief discussion on the related works on policy poisoning and reward poisoning (Ma et al., 2019; Sun et al., 2021; Huang & Zhu, 2019).

Ma et al. (2019) and Huang & Zhu (2019) study reward poisoning where the attacker can modify the rewards in an offline dataset or the reward signals during the online interaction. The attacker's goal is to force learning a particular target policy, or minimize the agent's reward in the original task. Under this framework, Ma et al. (2019) considers two victim learners with specific assumptions on the learner structure and develops attacks for them, while Huang & Zhu (2019) provides a theoretical analysis on the conditions for successful attacks against a Q-learning agent. In comparison to only reward poisoning in Ma et al. (2019) and Huang et al. (2017), Sun et al. (2021) focuses on general policy poisoning attacks for policy gradient learners in the online RL setting by using the vulnerability-awareness metric to decide when to poison, and the adversarial critic to guide the poisoning. Sun et al. (2021) achieve successful poisoning against policy-based agents in various complex environments in online RL, but it remains an interesting open problem as to how to poison the offline RL dataset which is agnostic to the learning algorithm.

I.2 Empirically Robust RL

Robust RL against Evasion Attacks. We briefly review several categories of RL methods that demonstrate empirical robustness against evasion attacks.

Randomization methods (Tobin et al., 2017; Akkaya et al., 2019) were first proposed to encourage exploration. This type of method was later systematically studied for its potential to improve model robustness. NoisyNet (Fortunato et al., 2017) adds parametric noise to the network's weight during training, providing better resilience to both training-time and test-time attacks (Behzadan & Munir, 2017; 2018), also reducing the transferability of adversarial examples, and enabling quicker recovery with fewer number of transitions during phase transition.

Under the *adversarial training* framework, Kos & Song (2017) and Behzadan & Munir (2017) show that re-training with random noise and FGSM perturbations increases the resilience against adversarial examples. Pattanaik et al. (2018) leverage attacks using an engineered loss function

specifically designed for RL to significant increase the robustness to parameter variations. RS-DQN (Fischer et al., 2019) is an *imitation learning* based approach that trains a robust student-DQN in parallel with a standard DQN in order to incorporate the constrains such as SOTA adversarial defenses (Madry et al., 2017; Mirman et al., 2018).

SA-DQN (Zhang et al., 2020a) is a *regularization* based method that adds regularizers to the training loss function to encourage the top-1 action to stay unchanged under perturbation.

Built on top of the *neural network verification* algorithms (Gowal et al., 2018; Weng et al., 2018), Radial-RL (Oikarinen et al., 2020) proposes to minimize an adversarial loss function that incorporates the upper bound of the perturbed loss, computed using certified bounds from verification algorithms. CARRL (Everett et al., 2021) aims to compute the lower bounds of action-values under potential perturbation and select actions according to the worst-case bound, but it relies on linear bounds (Weng et al., 2018) and is only suitable for low-dimensional environments.

These robust RL methods only provide empirical robustness against perturbed state inputs during test time, but cannot provide theoretical guarantees for the performance of the trained models under any bounded perturbations.

Robust RL against Poisoning Attacks. Banihashem et al. (2021) considers the *threat model* of reward poisoning attacks proposed by Ma et al. (2019); Rakhsha et al. (2020); Zhang et al. (2020b), where the attacker aims to force learning a target policy while minimizing the cost of reward manipulation. As shown in Ma et al. (2019); Rakhsha et al. (2020); Zhang et al. (2020b), this optimization problem is feasible and always has a unique optimal solution. Leveraging this property, Banihashem et al. (2021) formulates the defense task as another optimization problem which aims to optimize the agent's worst case performance among the set of plausible candidates of the true reward function. They specifically consider two settings regarding the agent's knowledge of the attack parameter, and provide lower bounds on the performance of the defense policy. In contrast to Banihashem et al. (2021) which focuses on a specific type of attack, our paper considers the *threat model* of general poisoning attacks, where the attacker has the power to manipulate the training trajectories arbitrarily. Given limited knowledge regarding the attack, our proposed COPA framework is general and applicable to any potential attack. Our robustness is derived from the aggregation over both sub-policy level and temporal level. One similarity between Banihashem et al. (2021) and our work is that we both aim to provide lower bounds of the cumulative reward for our proposed method as the provable guarantee / certification criteria.

I.3 Robustness Certification for RL

Wu et al. (2021) provide the first robustness certification for RL against *test-time evasion attacks* following the line of work on randomized smoothing (Cohen et al., 2019; Salman et al., 2019). Concretely, they apply per-state smoothing to achieve the certification for per-state action stability, as well as trajectory smoothing to obtain the certification for cumulative reward. Notably, Wu et al. (2021) propose an adaptive tree search algorithm to explore all possible trajectories and thus derive the robustness guarantee for cumulative reward. The relationship between our COPA-Search and their Algorithm 3 is discussed in detail in Appendix E.3. In comparison, robustness in their work is derived from *smoothing*, while our robustness comes from *aggregation*. We propose aggregation protocols and certification methods that leverage the temporal information by dynamically aggregating over the past time steps, which is not covered in Wu et al. (2021).