Appendices

A. Proof of Theorem 1

Proof. Consider two MDPs with reward functions defined as $R+\Delta$ and $R-\Delta$, denote the Q table corresponding to them as $Q_{+\Delta}$ and $Q_{-\Delta}$, respectively. Let $\{(s_t, a_t)\}$ be any instantiated trajectory of the learner corresponding to the attack policy ϕ. By assumption, $\{(s_t, a_t)\}$ visits all (s, a) pairs infinitely often and α_t's satisfy $\sum \alpha_t = \infty$ and $\sum \alpha_t^2 < \infty$. Assuming now that we apply Q-learning on this particular trajectory with reward given by $r_t + \Delta$, standard Q-learning convergence applies and we have that $Q_{t,+\Delta} \to Q_{+\Delta}$ and similarly, $Q_{t,-\Delta} \to Q_{-\Delta}$ (Melo).

Next, we want to show that $Q_t(s,a) \le Q_{t,+\Delta}(s,a)$ for all $s \in S, a \in A$ and for all t. We prove by induction. First, we know $Q_0(s,a) = Q_{0,+\Delta}(s,a)$. Now, assume that $Q_k(s,a) \le Q_{k,+\Delta}(s,a)$. We have

$$
\begin{aligned}
& Q_{k+1,+\Delta}(s_{k+1}, a_{k+1}) && (14)\\
= \;& (1-\alpha_{k+1}) Q_{k,+\Delta}(s_{k+1}, a_{k+1}) + \alpha_{k+1}\left(r_{k+1} + \Delta + \gamma \max_{a' \in A} Q_{k,+\Delta}(s'_{k+1}, a')\right) && (15)\\
\ge \;& (1-\alpha_{k+1}) Q_{k}(s_{k+1}, a_{k+1}) + \alpha_{k+1}\left(r_{k+1} + \delta_{k+1} + \gamma \max_{a' \in A} Q_{k}(s'_{k+1}, a')\right) && (16)\\
= \;& Q_{k+1}(s_{k+1}, a_{k+1}), && (17)
\end{aligned}
$$

which established the induction. Similarly, we have $Q_t(s,a) \ge Q_{t,-\Delta}(s,a)$. Since $Q_{t,+\Delta} \to Q_{+\Delta}$, $Q_{t,-\Delta} \to Q_{-\Delta}$, we have that for large enough t,

$$
Q_{-\Delta}(s,a) \le Q_t(s,a) \le Q_{+\Delta}, \forall s \in S, a \in A. \tag{18}
$$

Finally, it's not hard to see that $Q_{+\Delta}(s,a) = Q^*(s,a) + \frac{\Delta}{1-\gamma}$ and $Q_{-\Delta}(s,a) = Q^*(s,a) - \frac{\Delta}{1-\gamma}$. This concludes the proof. ∎

B. Proof of Theorem 4

Proof. We provide a constructive proof. We first design an attack policy ϕ, and then show that ϕ is a *strong attack*. For the purpose of finding a strong attack, it suffices to restrict the constructed ϕ to depend only on (s,a) pairs, which is a special case of our general attack setting. Specifically, for any $\Delta > \Delta_3$, we define the following Q':

$$
Q'(s,a) = \begin{cases} Q^*(s,a) + \dfrac{\Delta}{(1+\gamma)}, & \forall s \in S^\dagger, a \in \pi^\dagger(s), \\ Q^*(s,a) - \dfrac{\Delta}{(1+\gamma)}, & \forall s \in S^\dagger, a \notin \pi^\dagger(s), \\ Q^*(s,a), \forall s \notin S^\dagger, a, \end{cases} \tag{19}
$$

where $Q^*(s,a)$ is the original optimal value function without attack. We will show $Q' \in \mathcal{Q}^\dagger$, i.e., the constructed Q' induces the target policy. For any $s \in S^\dagger$, let $a^\dagger \in \arg\max_{a \in \pi^\dagger(s)} Q^*(s,a)$, a best target action desired by the attacker under the original value function Q^*. We next show that $a^\dagger$ becomes the optimal action under Q'. Specifically, $\forall a' \notin \pi^\dagger(s)$, we have

$$
\begin{aligned}
Q'(s,a^\dagger) &= Q^*(s,a^\dagger) + \frac{\Delta}{(1+\gamma)} && (20)\\
&= Q^*(s,a^\dagger) - Q^*(s,a') + \frac{2\Delta}{(1+\gamma)} + Q^*(s,a') - \frac{\Delta}{(1+\gamma)} && (21)\\
&= Q^*(s,a^\dagger) - Q^*(s,a') + \frac{2\Delta}{(1+\gamma)} + Q'(s,a'), && (22)
\end{aligned}
$$

Next note that

$$\Delta > \Delta_3 \geq \frac{1+\gamma}{2}[\max_{a \notin \pi^\dagger(s)} Q^*(s,a) - \max_{a \in \pi^\dagger(s)} Q^*(s,a)] \tag{23}$$

$$= \frac{1+\gamma}{2}[\max_{a \notin \pi^\dagger(s)} Q^*(s,a) - Q^*(s,a^\dagger)] \tag{24}$$

$$\geq \frac{1+\gamma}{2}[Q^*(s,a') - Q^*(s,a^\dagger)], \tag{25}$$

which is equivalent to

$$Q^*(s,a^\dagger) - Q^*(s,a') > -\frac{2\Delta}{1+\gamma}, \tag{26}$$

thus we have

$$Q'(s,a^\dagger) = Q^*(s,a^\dagger) - Q^*(s,a') + \frac{2\Delta}{(1+\gamma)} + Q'(s,a') \tag{27}$$

$$> 0 + Q'(s,a') = Q'(s,a'). \tag{28}$$

This shows that under Q', the original best target action $a^\dagger$ becomes better than all non-target actions, thus $a^\dagger$ is optimal and $Q' \in \mathcal{Q}^\dagger$. According to Proposition 4 in (Ma et al., 2019), the Bellman optimality equation induces a unique reward function $R'(s,a)$ corresponding to Q':

$$R'(s,a) = Q'(s,a) - \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q'(s',a'). \tag{29}$$

We then construct our attack policy $\phi^{sas}_{\Delta_3}$ as:

$$\phi^{sas}_{\Delta_3}(s,a) = R'(s,a) - R(s,a), \forall s,a. \tag{30}$$

The $\phi^{sas}_{\Delta_3}(s,a)$ results in that the reward function after attack appears to be $R'(s,a)$ from the learner's perspective. This in turn guarantees that the learner will eventually learn Q', which achieves the target policy. Next we show that under $\phi^{sas}_{\Delta_3}(s,a)$, the objective value (5) is finite, thus the attack is feasible. To prove feasibility, we consider adapting Theorem 4 in (Even-Dar & Mansour, 2003), re-stated as below.

Lemma 7 (Even-Dar & Mansour)**.** *Assume the attack is $\phi^{sas}_{\Delta_3}(s,a)$ and let Q_t be the value of the Q-learning algorithm using polynomial learning rate $\alpha_t = (\frac{1}{1+t})^\omega$ where $\omega \in (\frac{1}{2}, 1]$. Then with probability at least $1-\delta$, we have $\|Q_T - Q'\|_\infty \leq \tau$ with*

$$T = \Omega\left(L^{3+\frac{1}{\omega}} \frac{1}{\tau^2} (\ln \frac{1}{\delta\tau})^{\frac{1}{\omega}} + L^{\frac{1}{1-\omega}} \ln \frac{1}{\tau}\right), \tag{31}$$

Note that $Q^\dagger$ is an open set and $Q' \in \mathcal{Q}^\dagger$. This implies that one can pick a small enough $\tau_0 > 0$ such that $\|Q_T - Q'\|_\infty \leq \tau_0$ implies $Q_T \in Q^\dagger$. From now on we fix this τ_0, thus the bound in the above theorem becomes

$$T = \Omega\left(L^{3+\frac{1}{\omega}} (\ln \frac{1}{\delta})^{\frac{1}{\omega}} + L^{\frac{1}{1-\omega}}\right). \tag{32}$$

As the authors pointed out in (Even-Dar & Mansour, 2003), the ω that leads to the tightest lower bound on T is around 0.77. Here for our purpose of proving feasibility, it is simpler to let $\omega \approx \frac{1}{2}$ to obtain a loose lower bound on T as below

$$T = \Omega\left(L^5 (\ln \frac{1}{\delta})^2\right). \tag{33}$$

Now we represent δ as a function of T to obtain that $\forall T > 0$,

$$P[\|Q_T - Q'\|_\infty > \tau_0] \leq C \exp(-L^{-\frac{5}{2}} T^{\frac{1}{2}}). \tag{34}$$

Let $e_t = \mathbb{1}\left[\|Q_t - Q'\|_\infty > \tau_0\right]$, then we have

$$\mathbb{E}_{\phi^{sas}_{\Delta_3}}\left[\sum_{t=1}^{\infty}\mathbf{1}[Q_t \notin \mathcal{Q}^\dagger]\right] \leq \mathbb{E}_{\phi^{sas}_{\Delta_3}}\left[\sum_{t=1}^{\infty} e_t\right] \tag{35}$$

$$= \sum_{t=1}^{\infty} P[\|Q_T - Q'\|_\infty > \tau_0] \leq \sum_{t=1}^{\infty} C\exp(-L^{-\frac{5}{2}}t^{\frac{1}{2}}) \tag{36}$$

$$\leq \int_{t=0}^{\infty} C\exp(-L^{-\frac{5}{2}}t^{\frac{1}{2}})dt = 2CL^5, \tag{37}$$

which is finite. Therefore the attack is feasible.

It remains to validate that $\phi^{sas}_{\Delta_3}$ is a legitimate attack, i.e., $|\delta_t| \leq \Delta$ under attack policy $\phi^{sas}_{\Delta_3}$. By Lemma 7 in (Ma et al., 2019), we have

$$|\delta_t| = |R'(s_t, a_t) - R(s_t, a_t)| \tag{38}$$

$$\leq \max_{s,a}[R'(s,a) - R(s,a)] = \|R' - R\|_\infty \tag{39}$$

$$\leq (1+\gamma)\|Q' - Q^*\| = (1+\gamma)\frac{\Delta}{(1+\gamma)} = \Delta. \tag{40}$$

Therefore the attack policy $\phi^{sas}_{\Delta_3}$ is valid. ■

Discussion on a number of non-adaptive attacks: Here, we discuss and contrast 3 non-adaptive attack polices developed in this and prior work:

1. (Huang & Zhu, 2019) produces the non-adaptive attack that is feasible with the smallest Δ. In particular, it solves for the following optimization problem:

$$\min_{\delta, Q\in\mathbb{R}^{S\times A}} \|\delta\|_\infty \tag{41}$$

$$\text{s.t. } Q(s,a) = \delta(s,a) + \mathbb{E}_{P(s'|s,a)}\left[R(s,a,s) + \gamma\max_{a'\in A} Q(s',a')\right] \tag{42}$$

$$Q \in \mathcal{Q}^\dagger \tag{43}$$

   where the optimal objective value implicitly defines a $\Delta'_3 < \Delta_3$. However, it's a fixed policy independent of the actual Δ . In other word, It's either feasible if $\Delta > \Delta'_3$, or not.
2. $\phi^{sas}_{\Delta_3}$ is a closed-form non-adaptive attack that depends on Δ. $\phi^{sas}_{\Delta_3}$ is guaranteed to be feasible when $\Delta > \Delta_3$. However, this is sufficient but not necessary. Implicitly, there exists a Δ''_3 which is the necessary condition for the feasibility of $\phi^{sas}_{\Delta_3}$. Then, we know $\Delta''_3 > \Delta'_3$, because Δ'_3 is the sufficient and necessary condition for the feasibility of any non-adaptive attacks, whereas Δ''_3 is the condition for the feasibility of non-adaptive attacks of the specific form constructed above.
3. ϕ^{sas}_{TD3} (assume perfect optimization) produces the most efficient non-adaptive attack that depends on Δ.

In terms of efficiency, ϕ^{sas}_{TD3} achieves smaller $J_\infty(\phi)$ than $\phi^{sas}_{\Delta_3}$ and (Huang & Zhu, 2019). It's not clear between $\phi^{sas}_{\Delta_3}$ and (Huang & Zhu, 2019) which one is better. We believe that in most cases, especially when Δ is large and learning rate α_t is small, $\phi^{sas}_{\Delta_3}$ will be faster, because it takes advantage of that large Δ, whereas (Huang & Zhu, 2019) does not. But there probably exist counterexamples on which (Huang & Zhu, 2019) is faster than $\phi^{sas}_{\Delta_3}$.

C. The Covering Time L is $O(\exp(|S|))$ for the chain MDP

Proof. While the ε-greedy exploration policy constantly change according to the agent's current policy π_t, since L is a uniform upper bound over the whole sequence, and we know that π_t will eventually converge to $\pi^\dagger$, it suffice to show that the covering time under $\pi^\dagger_\varepsilon$ is $O(\exp(|S|))$.

Recall that $\pi^\dagger$ prefers going right in all but the left most grid. The covering time in this case is equivalent to the expected number of steps taken for the agent to get from s_0 to the left-most grid, because to get there, the agent necessarily visited all

states along the way. Denote the non-absorbing states from right to left as $s_0, s_1, ..., s_{n-1}$, with $|S| = n$. Denote V_k the expected steps to get from state s_k to s_{n-1}. Then, we have the following recursive relation:

$$V_{n-1} = 0 \tag{44}$$
$$V_k = 1 + (1 - \frac{\varepsilon}{2})V_{k-1} + \frac{\varepsilon}{2}V_{k+1}, \text{for } k = 1, ..., n-2 \tag{45}$$
$$V_0 = 1 + (1 - \frac{\varepsilon}{2})V_0 + \frac{\varepsilon}{2}V_1 \tag{46}$$

Solving the recursive gives

$$V_0 = \frac{p(1 + p(1-2p))}{(1-2p)^2}\left[(\frac{1-p}{p})^{n-1} - 1\right] \tag{47}$$

where $p = \frac{\varepsilon}{2} < \frac{1}{2}$ and thus $V_0 = O(\exp(n))$. ■

D. Proof of Theorem 5

Lemma 8. *For any state $s \in S$ and target actions $A(s) \subset A$, it takes FAA at most $\frac{|A|}{1-\varepsilon}$ visits to s in expectation to enforce the target actions $A(s)$.*

Proof. Denote V_t the expected number of visits s to teach $A(s)$ given that under the current Q_t, $\max_{a \in A(s)}$ is ranked t among all actions, where $t \in 1, ..., |A|$. Then, we can write down the following recursion:

$$V_1 = 0 \tag{48}$$
$$V_t = 1 + (1-\varepsilon)V_{t-1}\varepsilon\left[\frac{t-1}{|A|}V_{t-1} + \frac{1}{A}V_1 + \frac{|A|-t}{|A|}V_t\right] \tag{49}$$

Equation (49) can be simplified to

$$V_t = \frac{1 - \varepsilon + \varepsilon\frac{t-1}{|A|}}{1 - \varepsilon\frac{|A|-t}{|A|}}V_{t-1} + \frac{1}{1 - \varepsilon\frac{|A|-t}{|A|}} \tag{50}$$
$$\leq V_{t-1} + \frac{1}{1-\varepsilon} \tag{51}$$

Thus, we have

$$V_t \leq \frac{t-1}{1-\varepsilon} \leq \frac{|A|}{1-\varepsilon} \tag{52}$$

as needed. ■

Now, we prove Theorem 5.

Proof. Let $i \in [1, n]$ be given. First, consider the number of episodes, on which the agent was found in at least one state s_t and is equipped with a policy π_t, s.t. $\pi_t(s_t) \notin \nu_i(s_t)$. Since each of these episodes contains at least one state s_t on which ν_i has not been successfully taught, and according to Lemma 2, it takes at most $\frac{|A|}{1-\varepsilon}$ visits to each state to successfully teach any actions $A(s)$, there will be at most $\frac{|S||A|}{1-\varepsilon}$ such episodes. These episodes take at most $\frac{|S||A|H}{1-\varepsilon}$ iterations for all target states. Out of these episodes, we can safely assume that the agent has successfully picked up ν_i for all the states visited.

Next, we want to show that the expected number of iterations taken by $\pi_i^\dagger$ to get to s_i is upper bounded by $\left[\frac{|A|}{\varepsilon}\right]^{i-1} D$, where $\pi_i^\dagger$ is defined as

$$\pi_i^\dagger = \underset{\pi \in \Pi, \pi(s_j) \in \pi^\dagger(s_j), \forall j \leq i-1}{\arg\min} \mathbb{E}_{s_0 \sim \mu_0}\left[d_\pi(s_0, s_i)\right]. \tag{53}$$

First, we define another policy

$$\hat{\pi}_i^\dagger(s) = \begin{cases} \pi^\dagger(s) & \text{if } s \in \{s_1, ..., s_{i-1}\} \\ \pi_{s_i}(s) & \text{otherwise} \end{cases} \tag{54}$$

Clearly $\mathbb{E}_{s_0\sim\mu_0}\left[d_{\pi_i^\dagger}(s_0,s_i)\right] \le \mathbb{E}_{s_0\sim\mu_0}\left[d_{\hat{\pi}_i^\dagger}(s_0,s_i)\right]$ for all i.

We now prove by induction that $d_{\hat{\pi}_i^\dagger}(s,s_i) \le \left[\frac{|A|}{\varepsilon}\right]^{i-1} D$ for all i and $s \in S$.

First, let $i=1$, $\hat{\pi}_i^\dagger = \pi_{s_1}$, and thus $d_{\hat{\pi}_i^\dagger}(s,s_i) \le D$.

Next, we assume that when $i=k$, $d_{\hat{\pi}_i^\dagger}(s,s_i) \le D_k$, and would like to show that when $i=k+1$, $d_{\hat{\pi}_i^\dagger}(s,s_i) \le \left[\frac{|A|}{\varepsilon}\right] D_k$. Define another policy

$$\tilde{\pi}_i^\dagger(s) = \begin{cases} \pi^\dagger(s) & \text{if } s \in \{s_2,...,s_{i-1}\} \\ \pi_{s_i}(s) & \text{otherwise} \end{cases} \tag{55}$$

which respect the target policies on $s_2,...,s_{i-1}$, but ignore the target policy on s_1. By the inductive hypothesis, we have that $d_{\tilde{\pi}_i^\dagger}(s,s_i) \le D_k$. Consider the difference between $d_{\hat{\pi}_{i(s)}^\dagger}(s_1,s_k)$ and $d_{\tilde{\pi}_i^\dagger}(s_1,s_k)$. Since $\hat{\pi}_i^\dagger(s)$ and $\tilde{\pi}_i^\dagger$ only differs by their first action at s_1, we can derive Bellman's equation on each policy, which yield

$$\begin{aligned} d_{\hat{\pi}_i^\dagger}(s_1,s_k) &= (1-\varepsilon)Q(s_1,\pi^\dagger(s_1)) + \varepsilon\bar{Q}(s_1,a) && (56)\\ &\le \max_{a\in A} Q(s_1,a) && (57)\\ d_{\tilde{\pi}_i^\dagger}(s_1,s_k) &= (1-\varepsilon)Q(s_1,\pi_{s_1}(s_1)) + \varepsilon\bar{Q}(s_1,a) && (58)\\ &\ge \frac{\varepsilon}{|A|}\max_{a\in A} Q(s_1,a) && (59)\\ & && (60) \end{aligned}$$

where $Q(s_1,a)$ denotes the expected distance to s_k from s_1 by performing action a in the first step, and follow $\hat{\pi}_i^\dagger$ thereafter, and $\bar{Q}(s_1,a)$ denote the expected distance by performing a uniformly random action in the first step. Thus,

$$d_{\hat{\pi}_i^\dagger}(s,s_k) \le \frac{|A|}{\varepsilon} d_{\tilde{\pi}_i^\dagger}(s_1,s_k) \tag{61}$$

With this, we can perform the following decomposition:

$$\begin{aligned} d_{\hat{\pi}_i^\dagger}(s,s_k) &= \mathbb{P}\left[\text{visit } s_1 \text{ before reaching } s_k\right]\left(d_{\hat{\pi}_i^\dagger}(s,s_1) + d_{\hat{\pi}_i^\dagger}(s_1,s_k)\right) + \mathbb{P}\left[\text{not visit } s_1\right]\left(d_{\hat{\pi}_i^\dagger}(s,s_1)|\text{not visit } s_1\right)\\ &\le \mathbb{P}\left[\text{visit } s_1 \text{ before reaching } s_k\right]\left(d_{\tilde{\pi}_i^\dagger}(s,s_1) + \frac{|A|}{\varepsilon} d_{\tilde{\pi}_i^\dagger}(s_1,s_k)\right) + \mathbb{P}\left[\text{not visit } s_1\right]\left(d_{\tilde{\pi}_i^\dagger}(s,s_k)|\text{not visit } s_1\right)\\ &= d_{\tilde{\pi}_i^\dagger}(s,s_k) + \left(\frac{|A|}{\varepsilon} - 1\right) d_{\tilde{\pi}_i^\dagger}(s_1,s_k)\\ &\le D_k + \left(\frac{|A|}{\varepsilon} - 1\right) D_k = \frac{|A|}{\varepsilon} D_k. \end{aligned}$$

This completes the induction. Thus, we have

$$d_{\hat{\pi}_i^\dagger}(s,s_i) \le \left(\frac{|A|}{\varepsilon}\right)^{i-1} D, \tag{62}$$

and the total number of iterations taken to arrive at all target states sequentially sums up to

$$\sum_{i=1}^{n} d_{\hat{\pi}_i^\dagger}(s,s_i) \le \left(\frac{|A|}{\varepsilon}\right)^{n} D. \tag{63}$$

Finally, each target states need to visited for $\frac{|A|}{1-\varepsilon}$ number of times to successfully enforce $\pi^\dagger$. Adding the numbers for enforcing each $\pi_i^\dagger$ gives the correct result. ■

E. Detailed Explanation of Fast Adaptive Attack Algorithm

In this section, we try to give a detailed walk-through of the Fast Adaptive Attack Algorithm (FAA) with the goal of providing intuitive understanding of the design principles behind FAA. For the sake of simplisity, in this section we assume that the Q-learning agent is $\varepsilon = 0$, such that the attacker is able to fully control the agent's behavior. The proof of correctness and sufficiency in the general case when $\varepsilon \in [0, 1]$ is provided in section D.

The Greedy Attack: To begin with, let's talk about *the greedy attack*, a fundamental subroutine that is called in every step of FAA to generate the actual attack. Given a desired (partial) policy ν, the greedy attack aims to teach ν to the agent in a greedy fashion. Specifically, at time step t, when the agent performs action a_t at state s_t, the greedy attack first look at whether a_t is a desired action at $s + t$ according to $s\nu$, i.e. whether $a_t \in \nu(s_t)$. If a_t is a desired action, the greedy attack will produce a large enough δ_t, such that after the Q-learning update, a_t becomes strictly more preferred than all undesired actions, i.e. $Q_{t+1}(s_t, a_t) > \max_{a \notin \nu(s_t)} Q_{t+1}(s_t, a)$. On the other hand, if a_t is not a desired action, the greedy attack will produce a negative enough δ_t, such that after the Q-learning update, a_t becomes strictly less preferred than all desired actions, i.e. $Q_{t+1}(s_t, a_t) < \max_{a \in \nu(s_t)} Q_{t+1}(s_t, a)$. It can be shown that with $\varepsilon = 0$, it takes the agent at most $|A| - 1$ visit to a state s, to force the desired actions $\nu(s)$.

Given the greedy attack procedure, one could directly apply the greedy attack with respect to $\pi^\dagger$ throughout the attack procedure. The problem, however, is efficiency. The attack is not considered success without the attacker achieving the target actions in ALL target states, not just the target states visited by the agent. If a target state is never visited by the agent, the attack never succeed. $\pi^\dagger$ itself may not efficiently lead the agent to all the target states. A good example is the chain MDP used as the running example in the main paper. In section C, we have shown that if an agent follows $\pi^\dagger$, it will take exponentially steps to reach the left-most state. In fact, if $\varepsilon = 0$, the agent will never reach the left-most state following $\pi^\dagger$, which implies that the naive greedy attack w.r.t. $\pi^\dagger$ is in fact infeasible. Therefore, explicit navigation is necessary. This bring us to the second component of FAA, *the navigation polices*.

The navigation polices: Instead of trying to achieve all target actions at once by directly appling the greedy attack w.r.t. $\pi^\dagger$, FAA aims at one target state at a time. Let $s^\dagger_{(1)}, ..., s^\dagger_{(k)}$ be an order of target states. We will discuss the choice of ordering in the next paragraph, but for now, we will assume that an ordering is given. The agent starts off aiming at forcing the target actions in a single target state $s^\dagger_{(1)}$. To do so, the attacer first calculate the corresponding navigation policy ν_1, where $\nu_1(s_t) = \pi_{s^\dagger_{(1)}}(s_t)$ when $s_t \neq s^\dagger_{(1)}$, and $\nu_1(s_t) = \pi^\dagger(s_t)$ when $s_t = s^\dagger_{(1)}$. That is, ν_1 follows the shortest path policy w.r.t. $s^\dagger_{(1)}$ when the agent has not arrived at $s^\dagger_{(1)}$, And when the agent is in $s^\dagger_{(1)}$, ν_1 follows the desired target actions. Using the greedy attack w.r.t. ν_1 allows the attacker to effectively lure the agent into $s^\dagger_{(1)}$ and force the target actions $\pi^\dagger(s^\dagger_{(1)})$. After successfully forcing the target actions in $s^\dagger_{(1)}$, the attacker moves on to $s^\dagger_{(2)}$. This time, the attacker defines the navigation policy ν_2 similiar to ν_1, except that we don't want the already forced $\pi^\dagger(s^\dagger_{(1)})$ to be untaught. As a result, in ν_2, we define $\nu_2(s^\dagger_{(1)}) = \pi^\dagger(s^\dagger_{(1)})$, but otherwise follows the corresponding shortest-path policy $\pi_{s^\dagger_{(2)}}$. Follow the greedy attack w.r.t. ν_2, the attacker is able to achieve $\pi^\dagger(s^\dagger_{(2)})$ efficiently without affecting $\pi^\dagger(s^\dagger_{(1)})$. This process is carried on throughout the whole ordered list of target states, where the target actions for already achieved target states are always respected when defining the next ν_i. If each target states $s^\dagger_{(i)}$ can be reachable with the corresponding ν_i, then the whole process will terminate at which point all target actions are guaranteed to be achieved. However, the reachability is not always guaranteed with any ordering of target states. Take the chain MDP as an example. if the 2nd left target state is ordered before the left-most state, then after teaching the target action for the 2nd left state, which is moving right, it's impossible to arrive at the left-most state when the navigation policy resepct the moving-right action in the 2nd left state. Therefore, the *ordering* of target states matters.

The ordering of target states: FAA orders the target states descendingly by their shortest distance to the starting state s_0. Under such an ordering, the target states achieved first are those that are farther away from the starting state, and they necessarily do not lie on the shortest path of the target states later in the sequence. In the chain MDP example, the target states are ordered from left to right. This way, the agent is always able to get to the currently focused target state from the starting state s_0, without worrying about violating the already achieved target states to the left. However, note that the bound provided in theorem 5 do not utilize this particular ordering choice and applies to any ordering of target states. As a result, the bound diverges when $\varepsilon \to 0$, matching with the pathological case described at the end of the last paragraph.

Parameters	Values	Description
exploration noise	0.5	Std of Gaussian exploration noise.
batch size	100	Batch size for both actor and critic
discount factor	0.99	Discounting factor for the attacker problem.
policy noise	0.2	Noise added to target policy during critic update.
noise clip	$[-0.5, 0.5]$	Range to clip target policy noise.
action L2 weight	50	Weight for L2 regularization added to the actor network optimization objective.
buffer size	10^7	Replay buffer size, larger than total number of iterations.
optimizer	Adam	Use the Adam optimizer.
learning rate critic	10^{-3}	Learning rate for the critic network.
learning rate actor	5^{-4}	Learning rate for the actor network.
τ	0.002	Target network update rate.
policy frequency	2	Frequency of delayed policy update.

Table 1. Hyperparameters for TD3.

F. Experiment Setting and Hyperparameters for TD3

Throughout the experiments, we use the following set of hyperparameters for TD3, described in Table 1. The hyperparameters are selected via grid search on the Chain MDP of length 6. Each experiment is run for 5000 episodes, where each episode is of 1000 iteration long. The learned policy is evaluated for every 10 episodes, and the policy with the best evaluation performance is used for e evaluations in the experiment section.

G. Additional Experiments

G.1. Additional Plot for the rate comparison experiment

See Figure 8.

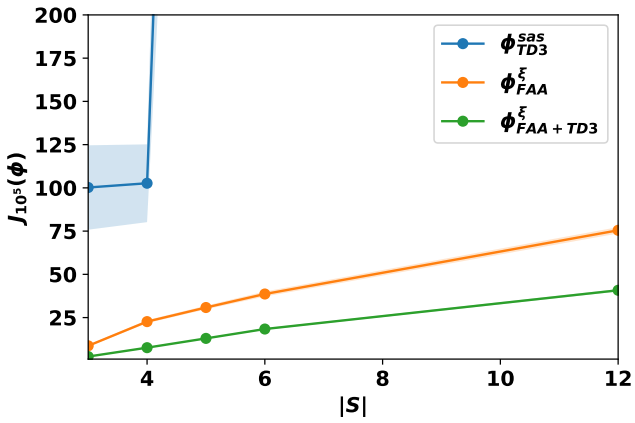

Figure 8. Attack performances on the chain MDP of different length in the normal scale. As can be seen in the plot, both ϕ^{ξ}_{FAA} + $\phi^{\xi}_{TD3+FAA}$ achieve linear rate.

G.2. Additional Experiments: Attacking DQN

Throughout the main paper, we have been focusing on attacking the tabular Q-learning agent. However, the attack MDP also applies to arbitrary RL agents. We describe the general interaction protocol in Alg. 4. Importantly, we assume that the RL agent can be fully characterized by an **internal state**, which determines the agent's current behavior policy as well as the learning update. For example, if the RL agent is a Deep Q-Network (DQN), the internal state will consist of the Q-network parameters as well as the transitions stored in the replay buffer.

Algorithm 4 Reward Poisoning against general RL agent

Parameters: MDP (S, A, R, P, μ_0), RL agent hyperparameters.

for $t = 0, 1, ...$ **do**
  agent at state s_t, has internal state θ_0.
  agent acts according to a behavior policy:
  $a_t \leftarrow \pi_{\theta_t}(s_t)$
  environment transits $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, produces reward $r_t = R(s_t, a_t, s_{t+1})$ and an end-of-episode indicator EOE.
  attacker perturbs the reward to $r_t + \delta_t$
  agent receives $(s_{t+1}, r_t + \delta_t, EOE)$, performs one-step of internal state update:

$$\theta_{t+1} = f(\theta_t, s_t, a_t, s_{t+1}, r_t + \delta_t, EOE) \tag{64}$$

  environment resets if $EOE = 1$: $s_{t+1} \sim \mu_0$.
end for

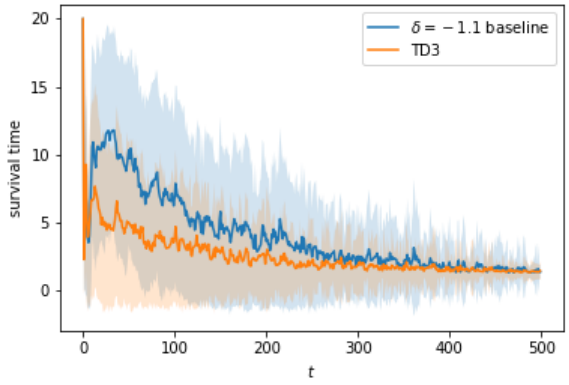

Figure 9. Result for attacking DQN on the Cartpole environment. The left figure plots the cumulative attack cost $J_T(\phi)$ as a function of T. The right figure plot the performance of the DQN agent $J(\theta_t)$ under the two attacks.

In the next example, we demonstrate an attack against DQN in the cartpole environment. In the cartpole environment, the agent can perform 2 actions, moving left and moving right, and the goal is to keep the pole upright without moving the cart out of the left and right boundary. The agent receives a constant $+1$ reward in every iteration, until the pole falls or the cart moves out of the boundary, which terminates the current episode and the cart and pole positions are reset.

In this example, the attacker's goal is to poison a well-trained DQN agent to perform as poorly as possible. The corresponding attack cost $\rho(\xi_t)$ is defined as $J(\theta_t)$, the expected total reward received by the current DQN policy in evaluation. The DQN is first trained in the clean cartpole MDP and obtains the optimal policy that successfully maintains the pole upright for 200 iterations (set maximum length of an episode). The attacker is then introduced while the DQN agent continues to train in the cartpole MDP. We freeze the Q-network except for the last layer to reduce the size of the attack state representation. We compare TD3 with a naive attacker that perform $\delta_t = -1.1$ constantly. The results are shown in Fig. 9.

One can see that under the TD3 found attack policy, the performance of the DQN agent degenerates much faster compared to the naive baseline. While still being a relatively simple example, this experiment demonstrates the potential of applying our adaptive attack framework to general RL agents.