Appendix

A The Anti-Poaching Game Model

In this section, we fully develop the POSG model of the Anti-Poaching Game.

A.1 Agents and game states

The game is played between a team of cooperative rangers $\mathcal{R} = \{1, \ldots, I\}$ and some independent poachers $\mathcal{P} = \{I+1, \ldots, I+J\}$ (note that $|\mathcal{R}| = I$ and $|\mathcal{P}| = J$) on a grid of size $\ell \times \ell$ over a finite horizon $[H] = \{1 \ldots, H\}$. Let $\mathcal{I}$ be the set of agents defined as $\mathcal{I} = \mathcal{R} \cup \mathcal{P}$. Each poacher has a finite number of traps that she can place and remove. A trap τ catches an animal at each time-step with a probability p_{CA}. Lastly, we remark that since the game is a POSG, all agents receive their observations, choose their next actions and obtain rewards simultaneously.

The state at a time step t is defined as a tuple of agent and trap states $s^t = (\sigma^t, \tau^t) \in \mathcal{S}$, where

$$\sigma^t = \left(\sigma_1^t, \ldots, \sigma_{|\mathcal{R}|}^t, \ldots, \sigma_{|\mathcal{R}|+|\mathcal{P}|}^t\right) \tag{3a}$$

$$\tau^t = \left(\tau_{j,m,n}^t\right) \ \forall m, n \in [\ell], j \in \mathcal{P} \tag{3b}$$

Here, σ^t defines the states of the Ranger and Poacher agents as

$$\sigma_i^t = \begin{cases} (m, n), \forall i \in \mathcal{R} \\ (m, n, \eta_{trap}, \eta_{prey}), \forall i \in \mathcal{P} \end{cases} \tag{4}$$

where m, n indicates an agent's position on the grid, η_{trap} is the number of traps currently carried by the poacher, and η_{prey} is the number of currently carried prey. Furthermore, the state of active/placed traps is described as a tuple over each cell of the grid, capturing the number of empty, i.e. $\eta_{E,j}$, and full traps, i.e. $\eta_{F,j}$, for each Poacher $j \in \mathcal{P}$ as

$$\tau_{j,m,n}^t = (\eta_{E,j}, \eta_{F,j}) \in \mathbb{N}^2 \tag{5}$$

A.2 Actions

The action space for each ranger $i \in \mathcal{R}$ is $\mathcal{A}_i = \{\emptyset, \uparrow, \leftarrow, \downarrow, \rightarrow\}$. Similarly, each poacher $j \in \mathcal{P}$ has the action space $\mathcal{A}_j = \{\emptyset, \uparrow, \leftarrow, \downarrow, \rightarrow, place - trap\}$. The joint action for each timestep t is denoted as $a^t = \left(a_1^t, \ldots, a_{|\mathcal{R}|}^t, \ldots, a_{|\mathcal{R}|+|\mathcal{P}|}^t\right)$, $a^t \in \mathcal{A}$.

A.3 Transition Model

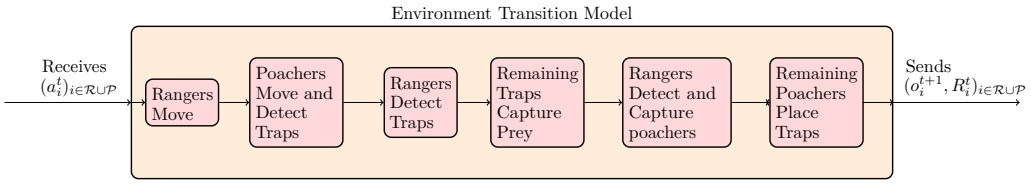


Figure 5: Transition Model for the Anti-Poaching Game. At each step, only the state variables of agents who apply or are affected by an action change value.

The transition probability function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ can be expressed as the probability $T(s^t, a^t, s^{t+1}) = \mathbb{P}(s^{t+1} | s^t, a^t)$.

Order of Transitions Since each action is chosen simultaneously, an order for simulating the actions must be established to avoid ambiguities. For example, the environment will simulate all $capture - poacher$ events before all $place - trap$ events. The complete transition model is defined in multiple steps, as shown in Figure 5. Note that the final state of each step is taken as the input of the next.

1. Each rangers moves deterministically generating a new individual state.
$$\sigma'_i \leftarrow Mov_i(\sigma_i, a_i), \forall i \in \mathcal{R}. \tag{6}$$
where Mov_i is a function that executes the movement action, which will be fully detailed in Section A.3.1.

2. Each poacher moves and reclaims only her traps from the new cell, if any.
$$(\sigma'_j, \tau'_j) \leftarrow PRT_j(Mov_j(\sigma_j, a_j), \tau_j), \forall j \in \mathcal{P}. \tag{7}$$
where PRT_j is a function that executes the Poacher-Remove-Trap action, which will be fully detailed in Section A.3.2.

3. All rangers now detect traps in their new cell and all remaining empty traps then independently capture an animal with probability p_{CA}. A ranger may detect each trap independently with probability p_{DT}.
Suppose that poacher $j \in \mathcal{P}$ has $\tau_{j,m,n} = (\eta_{E,j}, \eta_{F,j})$ empty and full traps in cell (m, n). Then, $P^{RRT}_{j,m,n}$ gives the probability of rangers detecting her traps in this cell, which gives the new trap state $\tau'_{j,m,n} = (\eta'_{E,j}, \eta'_{F,j})$. After this phase, $P^{CA}_{j,m,n}$ measures the probability of the remaining $\eta'_{F,j}$ empty traps to capture an animal and thus reach the trap state $\tau''_{j,m,n} = (\eta''_{E,j}, \eta''_{F,j})$. Both updates to the trap state can be written concisely as
$$P^{RRT,CA}_{j,m,n}(\tau''_{j,m,n}|\tau_{j,m,n}, \sigma) = \sum_{\substack{\eta''_{E,j} \le \eta'_{E,j} \le \eta_{E,j} \\ 0 \le \eta'_{F,j} \le \eta_{F,j}}} P^{CA}_{j,m,n}(\tau''_{j,m,n}|\tau'_{j,m,n}) \times P^{RRT}_{j,m,n}(\tau'_{j,m,n}|\tau_{j,m,n}, \sigma) \tag{8}$$

4. Rangers detect poachers in their new cell. A ranger detects a poacher with probability p_{DP}, and detected poachers are assigned the terminal state $(-1, -1, 0, 0)$. For a poacher j, P^{DP} gives the probability of being captured in this transition, thus transitioning her state from σ_j to σ'_j as $P^{DP}(\sigma'_j|\sigma_j)$. If j chose to place a trap and was not captured, she does so now. This changes the game state as $(\sigma''_j, \tau''_j) = PT_j(\sigma'_j, \tau_j, a_j)$. Since PT_j is deterministic, we condense the two steps as
$$P^{DP,PT}_j(\sigma''_j, \tau''_j|\sigma_j, \tau_j, a_j) = \sum_{\sigma'_j, PT_j(\sigma'_j, \tau_j, a_j) = (\sigma''_j, \tau''_j)} P^{DP}_j(\sigma'_j|\sigma_j), \forall j \in \mathcal{P}. \tag{9}$$

Let us now describe these transition steps in detail.

A.3.1 Movement

For each agent that has chosen a movement action, or the null action $\emptyset$, their state evolves deterministically as expected. We can define this using a Mov function defined on each agent as
$$\sigma' \leftarrow Mov(\sigma, a) = (Mov_i(\sigma_i, a_i))_{i \in \mathcal{I}}, \tag{10}$$
where σ is the current state of all agents, $a \in \mathcal{A}$ is their joint action, and Mov_i only processes action a_i of agent i. Let (m^t_i, n^t_i) be the location of agent i at time t. Then, we can specify the movement function Mov_i explicitly as follows
$$\left(m^{t+1}_i, n^{t+1}_i\right) = \begin{cases} (-1, -1) & \text{if } (m^t_i, n^t_i) = (-1, -1) \\ (\max(0, m^t_i - 1), n^t_i) & \text{if } a_i = \uparrow \\ (m^t_i, \max(0, n^t_i - 1)) & \text{if } a_i = \leftarrow \\ (\min(\ell, m^t_i + 1), n^t_i) & \text{if } a_i = \downarrow \\ (m^t_i, \min(\ell, n^t_i + 1)) & \text{if } a_i = \rightarrow \\ (m^t_i, n^t_i) & \text{if } a_i \in \{\emptyset, place - trap\} \end{cases} \tag{11}$$
where the new state of agent i is $\sigma^{t+1}_i = \left(m^{t+1}_i, n^{t+1}_i, \ldots\right)$. Mov_i does not update the position of the agent that chose an illegal action e.g. a movement beyond the grid border. Lastly, Mov does not modify the number of traps and prey of poachers i.e. $\left(\eta^{t+1}_{trap,j}, \eta^{t+1}_{prey,j}\right) = (\eta^t_{trap,j}, \eta^t_{prey,j}), \forall j \in \mathcal{P}$.

A.3.2 Removing Traps

We define the trap removal step similarly. We assume that each poacher can only detect and thus remove her own traps. This implies that poachers removing their own traps is a deterministic action. However, trap removal by rangers is still stochastic since they detect a trap with probability p_{DT}.

1. **Poachers Remove Traps** Since poachers remove only their traps, their action does not affect other agents or their traps, the transition *Poacher-Remove-Trap* (PRT) can be considered independently for each poacher as

$$(\sigma', \tau') = PRT(\sigma, \tau) = (PRT_j(\sigma_j, \tau_j))_{j \in \mathcal{P}} \tag{12}$$

Let $\sigma_j^t = (m_j^t, n_j^t, \eta_{trap,j}^t, \eta_{prey,j}^t)$ be the state of poacher j at time t, and $\tau_{j,m_j,n_j}^t = (\eta_{E,j}^t, \eta_{F,j}^t)$ the number of her empty and full traps in this cell. The number of traps poacher j has at $t+1$ is the sum of the traps she has at time t, and the number of empty and full traps she found in her new cell. Furthermore, Poachers store the prey taken from the full traps in their new cell.

$$\eta_{trap,j}^{t+1} \leftarrow \eta_{trap,j}^t + \eta_{E,j}^t + \eta_{F,j}^t \tag{13a}$$

$$\eta_{prey,j}^{t+1} \leftarrow \eta_{prey,j}^t + \eta_{F,j}^t \tag{13b}$$

2. **Rangers Remove Traps** Rangers also remove traps only in their new cell after a potential move. Detected traps are directly removed from the game and the reward obtained from removal is equally shared between rangers.

Suppose $k_{m,n} = |\{i \in \mathcal{R}, \sigma_i = (m,n)\}| \geq 1$ rangers are in cell $(m,n) \in [\ell]^2$. Each ranger searches the cell independently and detects a trap with probability p_{DT}. Let poacher j have placed $\eta_{E,j}$ (resp. $\eta_{F,j}$) empty traps (resp. full traps) in this cell i.e. $\tau_{j,m,n} = (\eta_{E,j}, \eta_{F,j})$. Since each ranger detects each trap independently, we can define the probability that a trap is detected by at least one of the $k_{m,n}$ rangers as

$$p_{DT,k_{m,n}} := P(trap \text{ is detected}) = 1 - (1 - p_{DT})^{k_{m,n}} \tag{14}$$

Thus, for cell (m,n) and traps of poacher j, the probability that there remains $\eta'_{E,j} \leq \eta_{E,j}$ undetected empty traps (resp. $\eta'_{F,j} \leq \eta_{F,j}$ undetected full traps) placed by poacher j is

$$P_{E,j}^{RRT}(\eta'_{E,j}|\eta_{E,j}, \sigma) = \binom{\eta_{E,j}}{\eta'_{E,j}} p_{DT,k_{m,n}}^{\eta_{E,j}-\eta'_{E,j}} \left(1 - p_{DT,k_{m,n}}\right)^{\eta'_{E,j}} \tag{15a}$$

$$P_{F,j}^{RRT}(\eta'_{F,j}|\eta_{F,j}, \sigma) = \binom{\eta_{F,j}}{\eta'_{F,j}} p_{DT,k_{m,n}}^{\eta_{F,j}-\eta'_{F,j}} \left(1 - p_{DT,k_{m,n}}\right)^{\eta'_{F,j}} \tag{15b}$$

Thus, the probability of reaching the new trap state $\tau'_{j,m,n} = (\eta'_{E,j}, \eta'_{E,j})$, $\forall (m,n) \in [\ell]^2, \forall j \in \mathcal{P}$ is

$$P_{j,m,n}^{RRT}(\tau'_{j,m,n}|\tau_{j,m,n}, \sigma) = P_{E,j}^{RRT}(\eta'_{E,j}|\eta_{E,j}, \sigma) \times P_{F,j}^{RRT}(\eta'_{F,j}|\eta_{F,j}, \sigma), \tag{16}$$

Taking the product over all cells and all poachers, we can define the probability of reaching trap state τ', given a current state (σ, τ) as

$$P^{RRT}(\tau'|\sigma, \tau) = \prod_{j \in \mathcal{P}, (m,n) \in [\ell]^2} P_{j,m,n}^{RRT}(\tau'_{j,m,n}|\tau_{j,m,n}, \sigma) \tag{17}$$

A.3.3 Rangers Remove Poachers

In this step, all the rangers search their current cell independently, similar to their search for traps in Section A.3.2. A detected poacher is removed from the grid along with the traps and prey they were carrying. Formally, their state evolves to $\sigma_j = (-1, -1, 0, 0)$ i.e. they are placed outside the grid with neither traps nor prey.

Let poacher j be located at cell (m,n), and suppose $k_{m,n}$ rangers are located in the same cell. Then, the probability that poacher $j \in \mathcal{P}$ is not detected (resp. detected) is:

$$P_j^{DP}(\sigma'_j = \sigma_j|\sigma) = (1-p_{DP})^{k_{m,n}} \tag{18a}$$

$$P_j^{DP}(\sigma'_j = (-1,-1,0,0)|\sigma) = 1-(1-p_{DP})^{k_{m,n}} \tag{18b}$$

where p_{DP} is the probability that a ranger detects a poacher located in his own cell. Therefore, the probability that state σ transitions to σ', where some poachers are detected (and therefore captured) is the product

$$P^{DP}(\sigma'|\sigma) = \prod_{j\in\mathcal{P}} P_j^{DP}(\sigma'_j|\sigma) \tag{19}$$

A.3.4 Traps Capture Animals

Each empty trap independently captures an animal with probability p_{CA}. Consider poacher j's traps in cell (m,n), denoted by the trap state $\tau^t_{j,m,n} = (\eta^t_{E,j}, \eta^t_{F,j})$. We first observe that the number of traps must be conserved at this step since no agent places or removes them during this step. Formally, states that do not conserve the number of traps are unreachable i.e.

$$P^{CA}_{j,m,n}(\tau'_{j,m,n}|\tau_{j,m,n}) = 0 \text{ if } (\eta^t_{E,j}+\eta^t_{F,j}) \neq (\eta'_{E,j}+\eta'_{F,j}) \tag{20}$$

Thus, the probability that for each poacher j, $\eta'_{E,j} \leq \eta_{E,j}$ empty traps **do not** capture animals is

$$P^{CA}_{j,m,n}\left(\tau'_{j,m,n} = (\eta'_{E,j}, \eta_{F,j}+\eta_{E,j}-\eta'_{E,j})|\tau_{j,m,n}\right) = \binom{\eta_{E,j}}{\eta'_{E,j}} p_{CA}^{\eta_{E,j}-\eta'_{E,j}} \times (1-p_{CA})^{\eta'_{E,j}}. \tag{21}$$

The transition function over trap states is the product of these probabilites over all poachers and all cells

$$P^{CA}(\tau'|\tau) = \prod_{j\in\mathcal{P},(m,n)\in[\ell]^2} P^{CA}_{j,m,n}(\tau'_{j,m,n}|\tau_{j,m,n}). \tag{22}$$

A.3.5 Poachers Place Trap

Finally, the action to place a trap is deterministic and independent for each poacher. Naturally, this action is considered only if the poacher has not been captured yet. In this case, an empty trap is subtracted from the stock of poacher i (if her stock of empty traps is non-empty, i.e. $\eta_{trap} > 0$) and added to the stock of the cell, whenever valid action *place-trap* is performed. For poacher j with state $\sigma_j = (m,n,\eta_{trap},\eta_{prey})$ and associated trap state $\tau_{j,m,n} = (\eta_{E,j},\eta_{F,j})$ at this cell,

$$\sigma'_i \leftarrow (m,n,\max(\eta_{trap}-1,0),\eta_{prey}) \tag{23a}$$

$$\tau'_{j,m,n} \leftarrow (\eta_{E,j}+\min(1,\eta_{trap}),\eta_{F,j}) \tag{23b}$$

The max and min operators handle the case where $\eta_{trap} = 0$. The new state for all poachers is simply the tuple

$$(\sigma',\tau') \leftarrow PT(\sigma,\tau,a) = \left(PT_j(\sigma_j,\tau_j,a_j)\right)_{j\in\mathcal{P}}, \tag{24}$$

where PT_j is given by Eqns. 23a and 23b when $a_j =$ *place-trap* and is the identity otherwise.

A.4 Agent Observations

We detail the calculation of the observation o_i^{t+1} using the transition $\langle s^t, a^t, s^{t+1}\rangle$.

Suppose that ranger i is located at $\sigma_i^{t+1} = (m,n)$, and observes $o_i^{t+1} = \left(t_{rem}, \sigma_i^{t+1}, s_R, \eta_P, \eta_{capt}, \eta_{cell}\right)$. Here, t_{rem} is the remaining time till the end of the game (i.e. $H-t$) and σ_i^{t+1} is her new state at time $t+1$. $s_R = \{i' \in \mathcal{R}, \sigma_{i'}^{t+1} = \sigma_i^{t+1}\}$ lists all the rangers located in ranger i's new cell and η_P counts the poachers captured in this cell at $t+1$. Let $\mathcal{P}^t_{capt}(m,n) = \{j \in \mathcal{P} : \sigma_j^t = (m,n,\eta^t_{trap,j},\eta^t_{prey,j}) \neq (-1,-1,0,0) = \sigma_j^{t+1}\}$.

	Rangers	Poachers
Ranger detects	1	p_{DP}
Poacher detects	p_{DP}	p_{DP}

Table 3: Probabilities with which an agent detects another type of agent.

Clearly, $\eta_P = |\mathcal{P}^t_{capt}(m,n)|$. The tuple $\eta_{capt} = \left(\eta^E_{capt}, \eta^F_{capt}\right)$ counts the number of Empty and Full traps recovered from these captured poachers and can be specified as

$$\eta_{capt} = \left(\eta^E_{capt}, \eta^F_{capt}\right) = \left(\sum_{j \in \mathcal{P}^t_{capt}(m,n)} \eta^t_{trap,j}, \sum_{j \in \mathcal{P}^t_{capt}(m,n)} \eta^t_{prey,j}\right) \tag{25}$$

The pair $\eta_{cell} = \left(\eta^E_{cell}, \eta^F_{cell}\right)$ counts the number of empty and full traps picked up by rangers in the arrival cell (m,n). Since all rangers in the same cell search collectively, they all receive the total number of traps detected in this cell as an observation. Recall that a ranger picks up traps only after the arriving poachers retrieve all of their traps in their new cell. Therefore, the traps recovered by rangers from the cell can only be owned by poachers not in this cell at time $t+1$.

Let $\mathcal{P}^{t+1}(m,n)$ denote this set of poachers in cell (m,n) at time $t+1$. The number of full and empty traps picked up by rangers in (m,n) is then

$$\eta^F_{cell} = \sum_{j \in \mathcal{P} \setminus \mathcal{P}^{t+1}(m,n)} \left(\eta^t_{F,j,m,n} - \eta^{t+1}_{F,j,m,n}\right) \tag{26a}$$

$$\eta^E_{cell} = \sum_{j \in \mathcal{P} \setminus \mathcal{P}^{t+1}(m,n)} \left(\eta^t_{E,j,m,n} - \eta^{t+1}_{E,j,m,n}\right) \tag{26b}$$

where $\tau^t_{j,m,n} = (\eta_{F,j,m,n}, \eta_{E,j,m,n})$ denotes the number of full and empty traps of poacher j in this cell.

Poacher j observes $o_j = \left(t_{rem}, \sigma^{t+1}_j, \eta_R, \eta_P\right)$, where t_{rem} is the remaining time, $\sigma^{t+1}_j = (m, n, \eta_{trap}, \eta_{prey})$ is her new state, and η_R (respectively η_P) is the number of rangers (resp. poachers) that she detects in her new cell. A poacher independently detects any agent, ranger or poacher, with probability p_{DP}. Suppose there are $k_{m,n}$ rangers (similarly, $q_{m,n}$ poachers) in poacher j's current cell. Therefore, the probability that poacher j detects $\eta_R \leq k_{m,n}$ rangers (similarly, $\eta_P \leq q_{m,n}$ poachers) is simply

$$P^{PDR}_j(\eta_R; k_{m,n}) = \binom{k_{m,n}}{\eta_R} p^{\eta_R}_{DP}(1-p_{DP})^{k_{m,n}-\eta_R} \tag{27a}$$

$$P^{PDP}_j(\eta_R; q_{m,n}) = \binom{q_{m,n}}{\eta_P} p^{\eta_P}_{DP}(1-p_{DP})^{q_{m,n}-\eta_P} \tag{27b}$$

A.5 Rewards

We assume that the stochastic game is zero sum, and that rewards/penalties are obtained through interactions between poachers and rangers. Suppose poacher j has the state $\sigma^t_j = (m, n, \eta_{trap}, \eta_{prey})$, with associated trap state $\tau^t_j = (\eta_{E,j}, \eta_{F,j})$. Her reward $R_j(s^t, a^t, s^{t+1})$ obtained when transition $\langle s^t, a^t, s^{t+1}\rangle$ occurs is

$$R_j(s^t, a^t, s^{t+1}) = r_j(\sigma^t_j, \sigma^{t+1}_j) + c_j(\sigma^t_j, \sigma^{t+1}_j) + \bar{c}_j(\tau^t_{j,m,n}, \tau^{t+1}_{j,m,n}) \tag{28}$$

where she obtains a reward $r_j(\sigma^t_j, \sigma^{t+1}_j)$ whenever she picks up a prey, but incurs the penalty $c_j(\sigma^t_j, \sigma^{t+1}_j)$ if she is captured and the penalty $\bar{c}_j(\tau^t_{j,m,n}, \tau^{t+1}_{j,m,n})$ whenever a full trap (prey)

is picked up by some ranger. Formally,

$$r_j(\sigma_j^t, \sigma_j^{t+1}) = R_{prey} \times \left(\eta_{prey}^{t+1} - \eta_{prey}^t\right) \tag{29a}$$

$$c_j(\sigma_j^t, \sigma_j^{t+1}) = \begin{cases} -C_{rem} - C_{trap} \times \eta_{trap} & \text{if } \sigma_j^t \neq (-1,-1,0,0) = \sigma_j^{t+1} \\ 0 & \text{else} \end{cases} \tag{29b}$$

$$\bar{c}_j(\tau_{j,m,n}^t, \tau_{j,m,n}^{t+1}) = C_{trap} \times \left(Traps^{t+1}(j) - Traps^t(j)\right) \tag{29c}$$

where $Traps^t(j)$ counts the number of active (not captured) traps that j has at time t, whether she is carrying them or if they are placed in the grid. Noting $\tau_{j,m,n}^t = \left(\eta_{E,j,m,n}^t, \eta_{F,j,m,n}^t\right)$, this is defined as

$$Traps^t(j) = \eta_{trap}^t + \sum_{(m,n)\in[\ell]^2} \left(\eta_{E,j,m,n}^t + \eta_{F,j,m,n}^t\right). \tag{30}$$

Since the game is zero-sum and the rangers share rewards equally, we can define ranger i's reward function as

$$R_i(s^t, a^t, s^{t+1}) = -\frac{1}{|\mathcal{R}|} \sum_{j\in\mathcal{P}} R_j(s^t, a^t, s^{t+1}). \tag{31}$$

A.6 Joint policies and values

Consider for agent i playing a POSG, the history of actions and observations upto time $t \leq T$: $h_i^t := h_i^t = (a_i^0, o_i^1, a_i^1, \ldots, a_i^{t-1}, o_i^t)$ (and $h_i^0 = \emptyset$). A policy $\pi = (\pi^0, \ldots, \pi^T)$ for this agent is a function that assigns each history at each time step a probability distribution over his action space, denoted $\pi^t(h_i^t) \in \Delta(\mathcal{A}_i)$. The joint policy is the tuple formed by the policies of all agents i.e. $\pi = (\pi_i)_{i\in\mathcal{I}}$. The set of all policies except agent i's is denoted as $\pi_{-i} = (\pi_j)_{j\in\mathcal{I}\setminus\{i\}}$.

To each policy, agent i can attribute a value to a history h_i^t as

$$v_{\pi,i}(h_i^t) = \mathbb{E}_\rho \left[\sum_{t'=t}^{H} \gamma^{t'-t} R_i(\sigma^{t'}, \tau^{t'}) \middle| \pi, h_i^t \right] \tag{32}$$

For given π_{-i} and history h_i^t, note $v_{i,\pi_{-i}}^*(h_i^t) = \max_{\pi_i} v_{(\pi_i,\pi_{-i}),i}(h_i^t)$. The set of individual policies π_i' which attain this value is called the set of Best Responses to π_{-i} and is denoted as $BR(\pi_{-i})$. A joint policy π^* is called a *Nash Equilibrium* if for each agent $i \in \mathcal{I}$, $\pi_i^* \in BR(\pi_{-i}^*)$. The value of this Nash equilibrium policy to any player i in history h_i^t is[7] $v_{\pi^*,i}^*(h_i^t) \equiv v_{\pi_{-i}^*,i}^*(h_i^t)$.

B Exploitability

As discussed in Section 4, we calculate the $AEXPL$ to obtain lower bounds on the distance of a joint strategy π from the Nash Equilibrium. This quantifies the robustness of a joint strategy since sub-optimal strategies are highly exploitable, and thus incite agents to deviate from their current strategy. The authors are not aware of a generic implementation of this method for RLlib-compatible algorithms. It is, however, implemented for other MARL libraries such as OpenSpiel [9].

Implementation The implementation ExploitabilityCallback is provided at examples/rllib/callbacks.py in the supplied source code. It is implemented as an RLlib callback which runs after each evaluation iteration.

When called, it creates one algorithm instance for each agent i by reusing the environment parameters and the current policies of all agents. In the instance corresponding to agent i, the policies π_{-i} are fixed, and agent i trains against these. An optimal policy for agent i in this restricted environment is the Best Response $BR(\pi_{-i})$.

[7] Both notations are thus equivalent.

It trains each algorithm instance in parallel for 100 iterations by default to obtain this approximate Best Response and then measures the ARR over 100 episodes to estimate $v^*_{\pi_{-i},i}(h_i^0)$. This is then used with the evaluation ARR, which estimates $v_{\pi,i}(h_i^0)$, to calculate $AEXPL(\pi)$.

C Experiments

This section gives further details on the experimental setup used for the results in Section 5. This includes the hyperparameter tuning for each algorithm and the default models used by each algorithm to train individual agent policies. It also includes additional experiments exploring the algorithms' sensitivity to the probability p_{CA} of a game.

Resources and Packages All experiments were conducted on an Intel Xeon processor with 24 cores and no GPU acceleration. Training an algorithm for a particular scenario used as few as 4 CPU cores. While training for a competitive scenario, ExploitabilityCallback replicates all policies (even for non-learning agents) once per agent to train them in parallel instances, which multiplies resource consumption by $|\mathcal{I}|$.

All experiments were run using python>=3.8 and the package ray[rllib]==2.8.0 with compatible versions of pettingzoo and gymnasium. These dependencies have been specified within the supplied source code.

Hyperparameter Tuning Each algorithm was tuned for both training scenarios, Cooperative and Competitive, using the Ray Tune tool. To train an algorithm for a scenario, 100 algorithm instances with randomly chosen hyperparameters were launched in parallel. Each instance was trained for 250,000 SETs and was evaluated at the end. This evaluation iteration measured the ARR over 100 game episodes for the Cooperative scenario, and additionally calculated $AEXPL$ for the Competitive scenario. The best instance in the Cooperative scenario maximised the ARR, and the best one in the Competitive setting minimised $AEXPL$. The parameters' values of the best trials were chosen for the experiments, and these are reported in Table 4 for IL-PG, Table 5 for IL-PPO, and Table 6 for QMix[8].

Training Batch Size	128
Discount Factor (γ)	0.97
Learning Rate	3e-05

(a) Hyperparameters values for Cooperative Scenario

Training Batch Size	512
Discount Factor (γ)	0.99
Learning Rate	3e-05

(b) Hyperparameters values for Competitive Scenarios

Table 4: Hyperparameters values for IL-PG

Default Models A Model refers to the neural network that parametrises the policy of a learning agent. APE agents require models that implement action masking, since some actions may be illegal based on the agent state. For example, an agent cannot cross the borders of the grid, or place a trap when she has already placed all her traps. For IL-PG and IL-PPO, we used the TorchActionMaskModel provided by RLlib with its default parameters. The default model used by QMix already implements action masking and was hence used with its default parameters.

C.1 Sensitivity to the p_{CA} parameter

Here, we discuss the effect of changing the probability p_{CA} (i.e. the probability that an empty trap captures an animal within a timestep) in a Cooperative-Competitive scenario. This is necessary since the Poachers must learn to exploit this change in the environment, which is not possible with heuristics. It is varied in two different ways, named *random* and *kernel*.

[8]All hyperparameters' values are stored in examples/rllib/hyperparams.json, and can be used with both the tuning and main scripts using the −−use−hyper flag.

Training Batch Size	256
Discount Factor (γ)	0.9
Learning Rate	0.004
Generalised Advantage Estimation	Yes
GAE Parameter (λ)	0.95
SGD Iterations	30
SGD Mini-batch Size	128
Value Function Loss Coefficient	0.01
Clipping Parameter	0.2
Gradient Clipping	1.0

(a) Hyperparameters values for Cooperative Scenarios

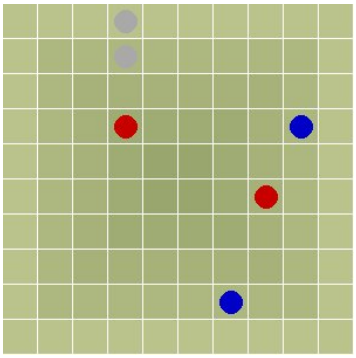
Training Batch Size	256
Discount Factor (γ)	0.999
Learning Rate	0.001
Generalised Advantage Estimation	Yes
GAE Parameter (λ)	0.95
SGD Iterations	30
SGD Mini-batch Size	128
Value Function Loss Coefficient	1.0
Clipping Parameter	0.3
Gradient Clipping	No

(b) Hyperparameters values for Competitive Scenarios

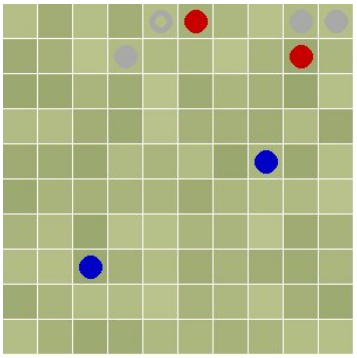
Table 5: Hyperparameters values for IL-PPO

Training Batch Size	2048
Discount Factor (γ)	0.9
Learning Rate	3e-05
Mixer	QMix
Gradient Clipping	10.0
Target Network Update Frequency (Per SETs)	500

Table 6: Hyperparameters values for QMix in Cooperative Scenarios



(a) 2R2P instance with p_{CA} varied in *kernel* mode



(b) 2R2P instance with p_{CA} varied in *random* mode

Figure 6: Illustration of a 2R2P APE instance with varying p_{CA} in *kernel* and *random* modes between $[0.2, 0.3]$. Darker cells correspond to higher p_{CA}, red dots are poachers, blue dots are rangers, hollow grey dots are empty traps and full grey dots are full traps.

In the *random* p_{CA} mode, the probability p_{CA} for each cell is sampled uniformly randomly between a given upper and lower bound. In the *kernel* mode, the p_{CA} changes linearly from an upper bound in the center to a given lower bound at the edges. This gives a hotspot near the center, making it more profitable (and riskier) to place a trap there. In our experiments, p_{CA} was bounded by the range $[0.2, 0.3]$. Since we execute in the cooperative-competitive mode, we only compare Independent Learners.

Changing p_{CA} affects the learned policies, and notably the ARRs - it is on average better for the rangers than the comparable experiments in Figure 4. Interestingly, the best performing policies on ARR are more exploitable and hence sub-optimal for at least one subset of agents.

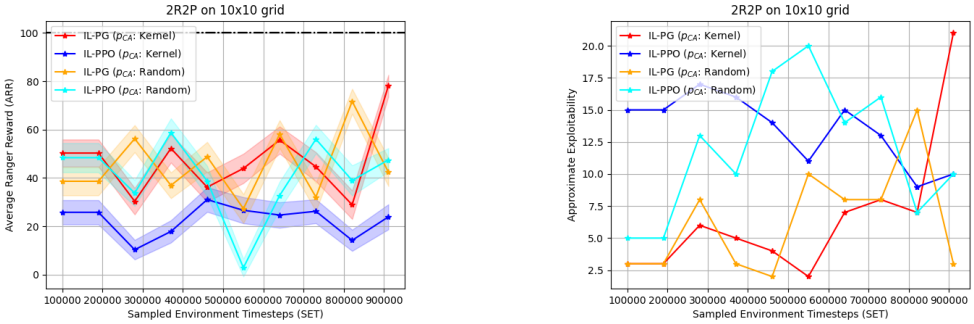


Figure 7: Comparison of Algorithm Performance in the Cooperative-Competitive Training Scenario where p_{CA} is varied in both *random* and *kernel* modes. The black line represents the reward for capturing all poachers.

A fall in ARR and $AEXPL$ suggests that the rangers are more exploitable, while a high ARR and high $AEXPL$ indicates that the poachers may be following a sub-optimal strategy.