8 Environment Details and Training Parameter Details

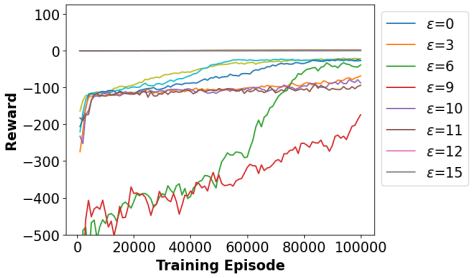
Cooperative navigation (CN): This is a cooperative game. There are 3 agents and 3 landmarks. Agents are rewarded based on how far any agent is from each landmark. Agents are penalized if they collide with other agents. So, agents have to learn to cover all the landmarks while avoiding collisions.

Keep away (KA): This is a competitive task. There is 1 agent, 1 adversary, and 1 landmark. The agent knows the position of the target landmark and wants to reach it. The adversary is rewarded if it is close to the landmark and if the agent is far from the landmark. The adversary should learn to push the agent away from the landmark.

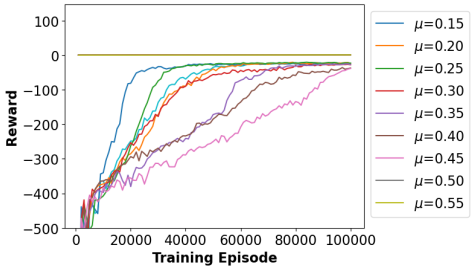
Physical deception (PD): This is a mixed cooperative and competitive task. There are 2 collaborative agents, 2 landmarks, and 1 adversary. Both the collaborative agents and the adversary want to reach the target, but only collaborative agents know the correct target. The collaborative agents should learn a policy to cover all landmarks so that the adversary does not know which one is the true target.

9 Reward Plots for Experiments

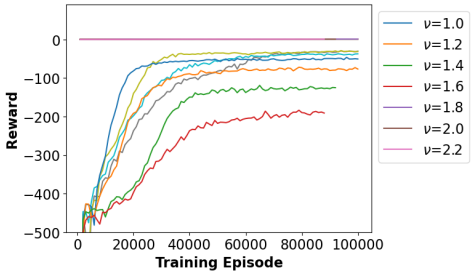
9.1 Cooperative Navigation Environment



(a) Reward plot for baseline for various ϵ in reward uncertainty. Baseline was able to learn only till ϵ = 9.

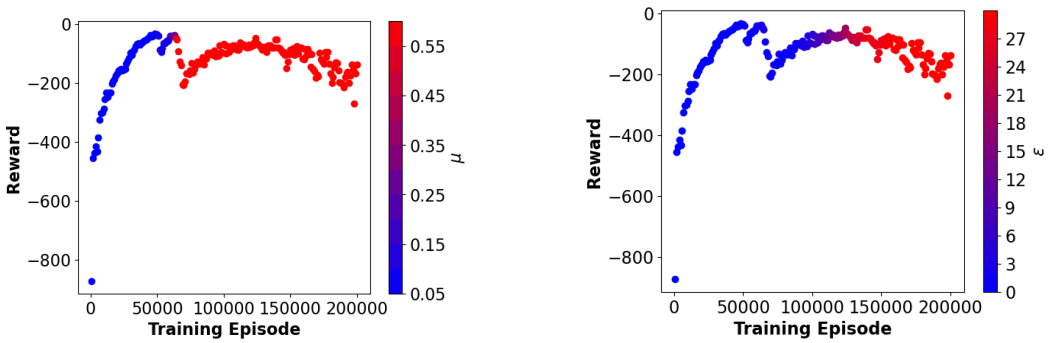


(b) Reward plot for baseline for various μ in state uncertainty. The baseline was able to learn only till μ = 0.5.



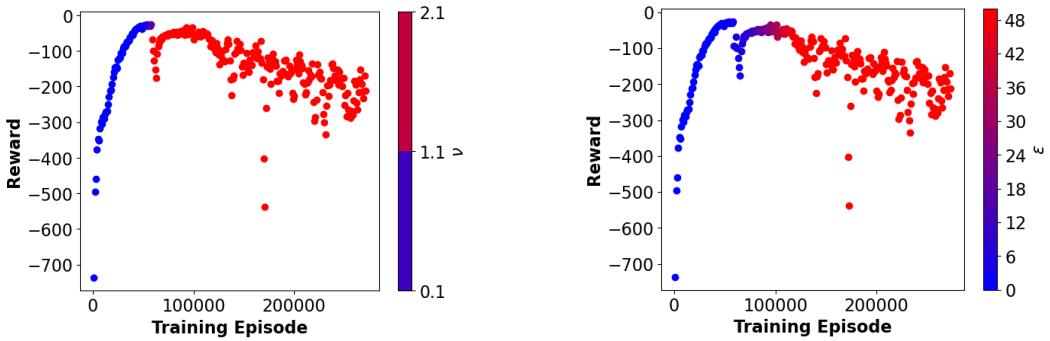
(c) Reward plot for baseline for various ν in action uncertainty. The baseline was able to learn only till μ = 2.0.

Figure 6: Cooperative Navigation: Baseline Performance. Rewards vs training time. Reward uncertainty shows good performance until ϵ=9 (left), state uncertainty shows good performance until μ=0.5 (middle) and action uncertainty shows good performance until ν=2.0 (right).



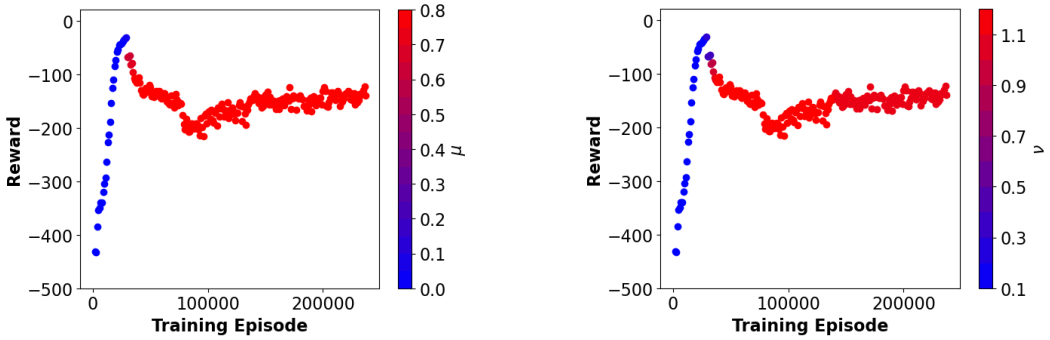
(a) Reward changes with μ **(b)** Reward changes with ϵ

Figure 7: Reward plot for lookahead CL for the case of multiple uncertainties (reward and state) showing the changing reward for various μ (left) and ϵ (right).



(a) Reward changes with ν **(b)** Reward changes with ϵ

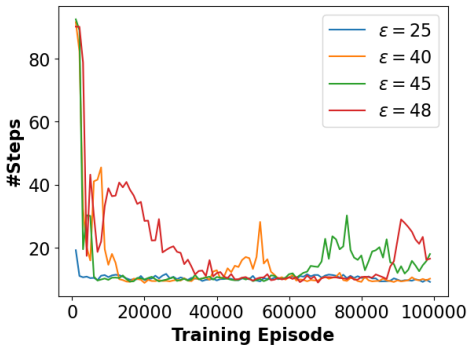
Figure 8: Reward plot for lookahead CL for the case of multiple uncertainties (reward and action) showing the changing reward for various ν (left) and ϵ (right).



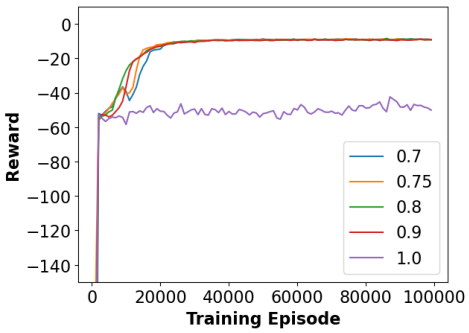
(a) Reward changes with μ **(b)** Reward changes with ν

Figure 9: Reward plot for lookahead CL for the case of multiple uncertainties (state and action) showing the changing reward for various μ (left) and ν (right).

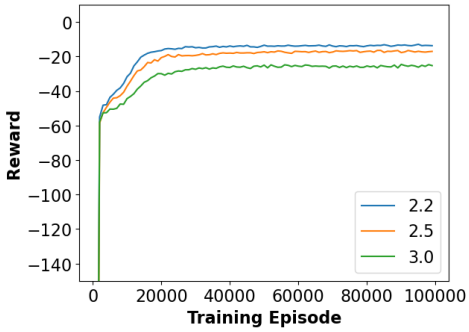
9.2 KEEP AWAY ENVIRONMENT



(a) Time taken by agent to reach the goal for various ϵ in reward uncertainty.

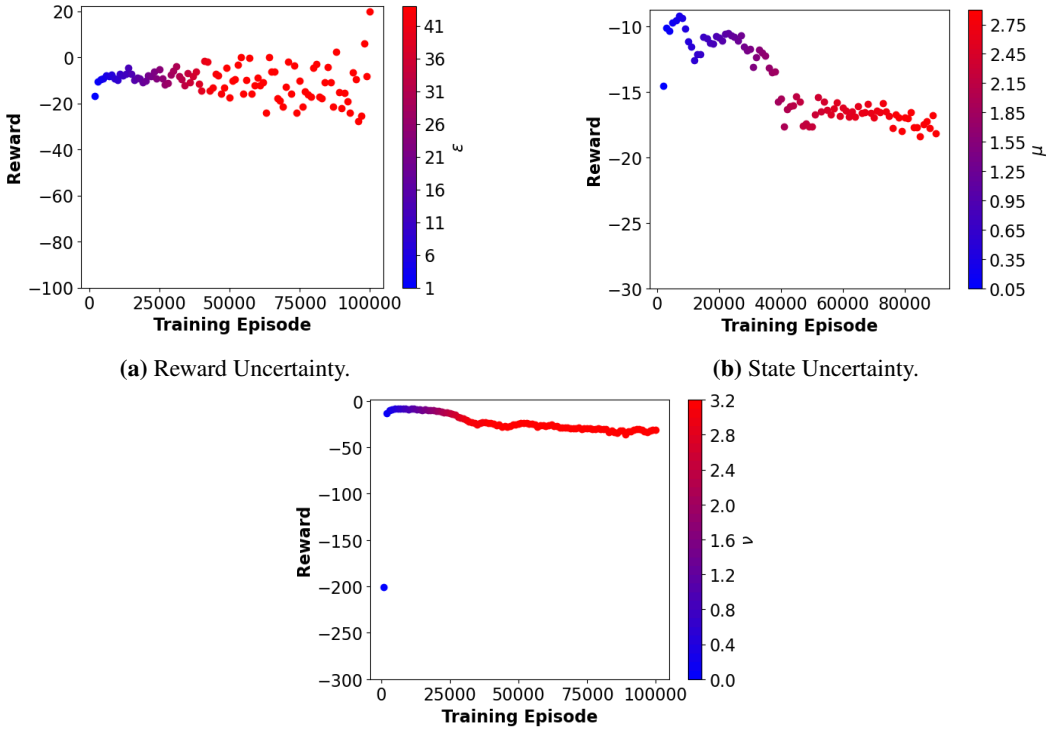


(b) Reward changes for various μ in state uncertainty.



(c) Reward changes for various ν in action uncertainty.

Figure 10: Keep Away: Baseline Performance. For reward uncertainty we show the plot between number of steps taken by an agent to reach the goal vs training time. This is because due to reward uncertainty reward is noisy and hence a plot of noisy reward will not give good conclusions. We observe that this number saturates for $\epsilon = 40$ but for number higher that this, its heavily fluctuating hence concluding that reward uncertainty learns until $\epsilon = 40$. For state and action uncertainty we show reward vs training time. State uncertainty shows good performance until μ=0.9 (middle) and action uncertainty shows good performance until ν=2.0 (last).



(a) Reward Uncertainty.

(b) State Uncertainty.

(c) Action Uncertainty.

Figure 11: Keep Away: CL Method Performance. This plot shows the changing reward as the noise value is incremented in the CL method for the three uncertain parameters separately. Reward uncertainty learns until ϵ=43 (left), state uncertainty until μ=2.5 (middle), and action uncertainty learns until ν=3.1 (last).

10 NASH EQUILIBRIUM FOR STATE UNCERTAINTY IN MARL

A nice proof for the conditional existence of Nash equilibrium is done in He et al. (2023) for the case of state uncertainty. They define the following robust Markov game,

$$\mathcal{G} = \{\mathcal{N}, \mathcal{M}, \{\mathcal{S}^i\}_{i\in\mathcal{N}}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \{\mathcal{B}^i\}_{i\in\mathcal{N}}, \{r^i\}_{i\in\mathcal{N}}, p, \gamma\}$$

$\mathcal{N} = \{1, 2, ..., N\}$ is the set of N agents and $\mathcal{M} = \{\bar{1}, \bar{2}, ..., \bar{N}\}$ is the corresponding set of N adversaries. $\gamma \in [0, 1)$ is the discount factor. $S = S_1 \times S_2 ... \times S_N$ is the joint state space. $A = A_1 \times A_2 ... \times A_N$ is the joint action space. $p : S \times A \to \Delta(S)$ are the state transition probabilities. r^i is the reward function for each agent. Every agent i is associated with an adversary $\bar{i}$. The adversary perturbs the true state of each agent $s^i \in S^i$ by producing an action $b^i \in B^i$. The perturbed state $\bar{s}^i = f(s^i, b^i)$ where f is a unique bijection given the state s^i.

The Markov game $\mathcal{G}$ is shown to be equivalent to a zero-sum two-person extensive-form game with finite strategies and perfect recall in He et al. (2023).

10.1 EXTENSIVE-FORM GAME

An extensive-form game (EFG) is a tree-based representation of a game. An EFG has one root node which indicates the start of the game. Each node branches out into multiple children nodes and each branch represents one possible action. The leaf nodes indicate the end of the game and contain the pay-off/reward for the actions specified by the path from the root node to the leaf node.

The robust optimization equation can be decomposed into a two-player EFG. The first player is the nature/combined adversary who selects the perturbed state and the next player is the combined agent which chooses the best action according to the policy to be learned. The nature player has $|\bar{\mathcal{S}}|$ possible choices for the action and the agent player has $|\mathcal{A}|$ choices where $A = A^1 \times A^2 \times ... \times A^n$ i.e. the space of all possible actions for all agents. The reward for the nature player is the negative of the reward obtained by the action taken by the combined agent.

The Bellman equation for the above game $\mathcal{G}$ is written as below:

$$v^i(s) = \max_{\pi^i} \min_{\rho^i} \mathbb{E}\left[\sum_{s'\in S} p(s'|s, a, b)[r^i(s, a, b) + \gamma v^i(s)]|a \sim \pi(\cdot|\bar{s}), b \sim \rho(\cdot|s)\right]$$

In order for the NE (and the optimal solution to the above equation) to exist, below conditions need to be met:

- $\mathcal{S}^i$, $\mathcal{A}^i$ and $\mathcal{B}^i$ must be finite sets $\forall i \in \mathcal{N}$.
- $|r^i(s, a, b)| < M_i < M < \infty \; \forall \; i \in N, a \in A, b \in B$ and $s \in S$
- Stationary reward and transition probabilities
- f is a bijection for a given s^i
- All agents have the same reward function.

11 NE FOR REWARD AND TRANSITION DYNAMICS UNCERTAINTY IN MARL

In this section, we show how uncertainty in reward and transition dynamics is handled in a multi-agent setting. We follow Kardeş et al. (2011b) and use the following definition of robust Markov game.

$$\bar{\mathcal{G}} = \langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \{\bar{\mathcal{R}}^i_s\}_{(i,s)\in\mathcal{N}\times\mathcal{S}}, \{\bar{\mathcal{P}}_s\}_{s\in\mathcal{S}}, \gamma\rangle$$

Note: In this proof following Kardeş et al. (2011b) s_t denotes the system state and not the individual agent state. The expected return in case of multi-agent RL with no uncertainty for i^{th} agent is -

$$V_{\pi}^{i}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t^i | s_0 = s, a_t^i \sim \pi^i(.|s_t), a_t^{-i} \sim \pi^{-i}(.|s_t)]$$

where $-i$ represents the indices of all agents except agent i, and $\pi^{-i} = \Pi_{i \neq j} \pi_j$ refers to the joint policy of all agents except agent i. In order to find the optimal robust value function for the single agent the other agent policies are considered stationary. Since all policies are evolving continuously and expected return is dependent on all agent policies, one commonly used solution for optimal policy $\pi^* = \{\pi_1^*, \pi_2^*, \dots \pi_N^*\}$ is Nash equilibrium. Non-stationarity is also one of the main reasons for difficulty in MARL convergence as compared to single agent RL which also reflects when uncertainty is added.

We now introduce uncertainty in rewards and transition dynamics. Thus, the desired policy should now not only be able to play against other agents' policies but also robust to the possible uncertainty of the MARL model. Each player considers a distribution-free Markov game to be played using robust optimization. To find the optimal value function we focus on the following idea from Kardeş et al. (2011b). If the player knows how to play in the robust Markov game optimally starting from the next stage on, then it would play to maximize not only the worst-case (minimal) expected immediate reward, due to the model uncertainty set at the current stage, but also the worst-case expected reward incurred in the future stages. Formally, such a recursion property leads to the following Bellman-type equation:

$$\bar{V}_*^i(s) = \max_{\pi^i(.|s)} \min_{\substack{\bar{P}(.|s,.) \in \bar{\mathcal{P}}_s \\ \bar{R}_s^i \in \bar{\mathcal{R}}_s^i}} \sum_{a \in A} \prod_{j=1}^{N} \pi^j(a^j|s)(\bar{R}^i(s,a) + \gamma \sum_{s' \in S} \bar{P}(s'|s,a) \bar{V}_*^i(s'))$$

The corresponding joint policy $\pi^* = \{\pi^1, \pi^2 \dots \pi^N\}$ is robust Markov perfect Nash equilibrium.

12 Proof for Theorem 1

Lets define the non-linear operator on $\mathcal{L}$ such that,

$$\mathcal{L}^i v^i(s) = \max_{\pi^i(.|s^i)} \min_{\rho} \left[\sum_{a \in \mathcal{A}} \bar{R}^i(s, \bar{a}) + \gamma \sum_{s' \in S} \bar{P}(s'|s, \bar{a}) v^i(s') \right]$$

, where $\rho = \{\bar{P}, \bar{R}, \bar{s}, \bar{a}\}$

We can think of ρ as adversarial strategy that is playing against the good policy π by selecting the values $\{\bar{P}, \bar{R}, \bar{s}, \bar{a}\}$ from their respective uncertainty sets such that it minimises the expected return.

Let u and v be two value functions in $\mathbb{V}$. Let $\{\pi_*^u, \rho_*^u\}$ and $\{\pi_*^v, \rho_*^v\}$ be two different Nash Equilibrium with respect to $\bar{\mathcal{G}}_{general}$.

$$\mathcal{L}^i v^i(s) = \sum_{a \in \mathcal{A}} \bar{R}^i(s, \bar{a})_{(\pi_*^v, \rho_*^v)} + \gamma \sum_{s' \in S} \bar{P}(s'|s, \bar{a})_{(\pi_*^v, \rho_*^v)} v^i(s')$$

, where $\rho_*^v = \{\bar{P}, \bar{R}, \bar{s}, \bar{a}\}$ is the optimal value that minimises the value function equation.

$$\mathcal{L}^i u^i(s) = \sum_{a \in \mathcal{A}} \bar{R}^i(s, \bar{a})_{(\pi_*^u, \rho_*^u)} + \gamma \sum_{s' \in S} \bar{P}(s'|s, \bar{a})_{(\pi_*^u, \rho_*^u)} u^i(s')$$

, where $\rho_*^u = \{\bar{P}, \bar{R}, \bar{s}, \bar{a}\}$ is the optimal value that minimises the value function equation.

Its intuitive that optimal π_* maximizes the above equation, whereas optimal ρ_* minimises the above equation. Therefore we can write the following equation,

$$\sum_{a \in \mathcal{A}} \bar{R}^i(s, \bar{a})_{(\pi_*^u, \rho_*^v)} + \gamma \sum_{s' \in S} \bar{P}(s'|s, \bar{a})_{(\pi_*^u, \rho_*^v)} v^i(s') \leq \mathcal{L}^i v^i(s) \leq \sum_{a \in \mathcal{A}} \bar{R}^i(s, \bar{a})_{(\pi_*^v, \rho_*^u)} + \gamma \sum_{s' \in S} \bar{P}(s'|s, \bar{a})_{(\pi_*^v, \rho_*^u)} v^i(s')$$

$$\sum_{a\in\mathcal{A}} \bar{R}^i(s,\bar{a})_{(\pi_*^v,\rho_*^u)}+\gamma \sum_{s'\in S} \bar{P}(s'|s,\bar{a})_{(\pi_*^v,\rho_*^u)}u^i(s') \leq \mathcal{L}^i u^i(s) \leq \sum_{a\in\mathcal{A}} \bar{R}^i(s,\bar{a})_{(\pi_*^u,\rho_*^v)}+\gamma \sum_{s'\in S} \bar{P}(s'|s,\bar{a})_{(\pi_*^u,\rho_*^v)}u^i(s')$$

(a) Now lets assume, $\mathcal{L}^i v^i(s) \leq \mathcal{L}^i u^i(s)$

$$\begin{aligned}
0 &\leq \mathcal{L}^i u^i(s) - \mathcal{L}^i v^i(s) \\
&\leq \left[\sum_{a\in\mathcal{A}} \bar{R}^i(s,\bar{a})_{(\pi_*^u,\rho_*^v)} + \gamma \sum_{s'\in S} \bar{P}(s'|s,\bar{a})_{(\pi_*^u,\rho_*^v)}u^i(s')\right] - \left[\sum_{a\in\mathcal{A}} \bar{R}^i(s,\bar{a})_{(\pi_*^u,\rho_*^v)} + \gamma \sum_{s'\in S} \bar{P}(s'|s,\bar{a})_{(\pi_*^u,\rho_*^v)}v^i(s')\right] \\
&\leq \gamma \sum_{s'\in S} \bar{P}(s'|s,\bar{a})_{(\pi_*^u,\rho_*^v)}(u^i(s') - v^i(s')) \\
&\leq \gamma||u^i(s') - v^i(s')||
\end{aligned}$$

(b) Assuming , $\mathcal{L}^i u^i(s) \leq \mathcal{L}^i v^i(s)$ and following the same argument as before we get,

$$\mathcal{L}^i v^i(s) - \mathcal{L}^i u^i(s) \leq \gamma||v^i(s') - u^i(s')||$$

Thus, combining (a) and (b), we get,

$$||\mathcal{L}^i v^i(s) - \mathcal{L}^i u^i(s)|| \leq \gamma||v^i(s') - u^i(s')||$$

Thus, $\mathcal{L}^i$ is a contraction mapping on V

Now since $||v|| = \sup_i ||v^i||$, we can write the following -

$$||\mathcal{L}v - \mathcal{L}u|| = \sup_i ||\mathcal{L}^i v^i - \mathcal{L}^i u^i|| \leq \gamma \sup_i ||v^i - u^i|| = \gamma||v - u||$$

Thus, $\mathcal{L}$ is a contraction mapping on $\mathbb{V}$