

# Analyzing Probabilistic Logic Shields for Multi-Agent Reinforcement Learning

Satchit Chatterji<sup>a,\*</sup> and Erman Acar<sup>a</sup>

<sup>a</sup>IvI & ILLC, University of Amsterdam

## Abstract.

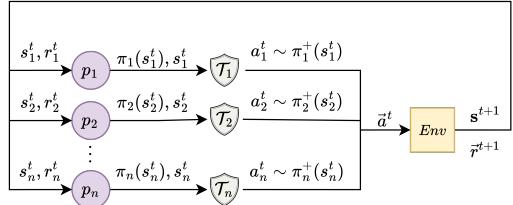
Safe reinforcement learning (RL) is crucial for real-world applications, and multi-agent interactions introduce additional safety challenges. While Probabilistic Logic Shields (PLS) has been a powerful proposal to enforce safety in single-agent RL, their generalizability to multi-agent settings remains unexplored. In this paper, we address this gap by conducting extensive analyses of PLS within decentralized, multi-agent environments, and in doing so, propose **Shielded Multi-Agent Reinforcement Learning (SMARL)** as a general framework for steering MARL towards norm-compliant outcomes. Our key contributions are: (1) a novel Probabilistic Logic Temporal Difference (PLTD) update for shielded, independent Q-learning, which incorporates probabilistic constraints directly into the value update process; (2) a probabilistic logic policy gradient method for shielded PPO with formal safety guarantees for MARL; and (3) comprehensive evaluation across symmetric and asymmetrically shielded  $n$ -player game-theoretic benchmarks, demonstrating fewer constraint violations and significantly better cooperation under normative constraints. These results position SMARL as an effective mechanism for equilibrium selection, paving the way toward safer, socially aligned multi-agent systems.

## 1 Introduction

Recent years have witnessed significant progress in multi-agent reinforcement learning (MARL), with sophisticated algorithms tackling increasingly complex problems in various domains including autonomous vehicles [34], distributed robotics [6], algorithmic trading [15], energy grid management [40] and healthcare [32]. The ultimate success of RL in this diverse collection of domains and the deployment in the real-world, however, demands overcoming a difficult key challenge: *safety*.

This has naturally resulted in the research direction of *Safe RL* which aims to learn optimal policies that are, by some measure, ‘safe’. Gu et al. [17] present an up-to-date overview of the field. A number of proposals focus on the application of formal methods [20, 12], within which the notion of *shielding* is used, a technique that is inspired by formal verification through temporal logic specifications to avoid unsafe actions during the agent’s learning process [5, 2, 21, 8].

One recent proposal that uses shielding to represent safety constraints is *probabilistic logic shields* [PLS, 46] whose semantics are based on probabilistic logic (PL) programming [10]. PLS constrains an agent’s policy to comply with formal specifications *probabilistically*. The specification of constraints, (the *shield*), is defined within



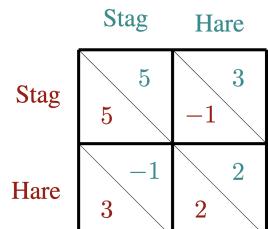
**Figure 1.** Shielded MARL (SMARL) interaction schematic. At time  $t$ , each agent  $p_i$  passes their respective policy  $\pi_i^t(s_i^t)$  and a safety-specific state  $s_i^t$  to their shield  $T_i$ , resulting in a safe policy  $\pi_i^+(s_i^t)$  from which an action  $a_i^t$  is sampled and returned to the environment.

the exploration and learning pipeline. PLS offers policy-level evaluation of safety in contrast to action-level hard-rejection shields [e.g., 20, 21, 8], differentiability, modest requirements on knowing the MDP, and safety guarantees within single-agent RL.

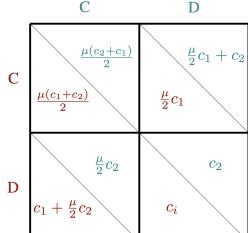
However, safety is inherently a *multi-agent* concept, as real-world environments often involve multiple agents interacting simultaneously, leading to hard-to-control complex systems. While there exist a few approaches tackling safety in multi-agent settings [16, 13, 26, 35], to the best of our knowledge, PLS has not been adopted, extended or analyzed in the context of MARL. In this paper, we address this gap with the following contributions:

1. We present a framework that extends shielding in RL to multi-agent settings, named *Shielded MARL* (SMARL). We provide a theoretical guarantee that SMARL with PLS produces safer joint policies than unshielded counterparts. Using decentralized techniques, we introduce (a) *Probabilistic Logic Temporal Difference Learning* (PLTD), with convergence guarantees, allowing for shielded independent Q-learning (SIQL), and (b) shielded independent PPO (SIPPO) using probabilistic logic policy gradients;
2. We show that PLS can be used as an equilibrium selection mechanism, a key challenge in MARL, under various game-theoretic settings, such as coordination, spatio-temporal coordination, social dilemmas, cooperation under uncertainty, and mixed-motive problems. We provide strong empirical evidence across various  $n$ -player environments including an extensive-form game (Centipede), a stochastic game (Extended Public Goods Game), a simultaneous game (Stag-Hunt), and its grid-world extension (Markov Stag-Hunt). Moreover, we investigate the impact of smaller (weak, less specific) and larger (strong, more specific) shields;
3. We investigate asymmetric shielding, i.e. the ability of shielded agents to influence unshielded peers. Results show that partial shielding can significantly enhance safety, highlighting SMARL’s effectiveness in both cooperative and non-cooperative settings.

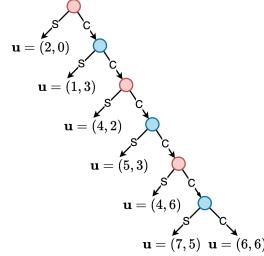
\* Corresponding Author. Email: s.chatterji@uva.nl



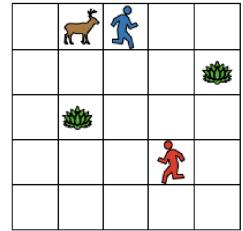
(a) Payoff matrix for *Stag-Hunt*



(b) Expected EPGG utilities



(c) *Centipede* instance



(d) *Markov Stag-Hunt* [30]

**Figure 2.** Representations of the games that are experimented within this paper. For details please refer to Section 2.3.

## 2 Preliminaries

### 2.1 Safety Definition

We expand the interpretation of ‘safety’ in RL beyond that of works like ElSayed-Aly et al. [13] and Yang et al. [46] (i.e. informally, *the agent should not perform actions that may lead to something ‘harmful’*), and more towards an alternative conception provided in Gu et al. [17, pg. 3], namely, *they act, reason, and generalize obeying human desire*. We formalize this in the vein of Alshiekh et al. [2]:

**Definition 2.1.** **Formally-Constrained Safe RL** is the process of learning an optimal policy  $\pi^*$  while maximally satisfying a set of formal specifications  $\mathcal{T}$  during learning and deployment.

$\mathcal{T}$  may represent any set of constraints, (e.g. arbitrary temporal logic formulae). These may not necessarily be ‘harmful’, but may instead ascribe normative or desirable behavior, such as social norms or equilibrium conditions. In our case,  $\mathcal{T}$  is a ProbLog program.

### 2.2 Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) generalizes RL to environments with multiple interacting agents, making the learning problem non-stationary since each agent’s dynamics depend on the evolving policies of others ([1] provides a detailed introduction). Formally, MARL is often modeled as a *stochastic game* in which each agent  $i$  selects actions according to a local policy  $\pi_i$ , and inducing a joint policy  $\vec{\pi} = \langle \pi_1, \dots, \pi_n \rangle$  determining the next state and rewards. A common baseline is *Independent Q-Learning* [IQL, 37], where each agent maintains its own Q-function and updates it independently using local observations and rewards. Similarly, *Independent Proximal Policy Optimization* [IPPO, 11] adapts PPO to the multi-agent case by training each one with a separate policy-gradient update. In practice, MARL algorithms may employ *parameter sharing*, where agents use a common neural architecture for policies and/or critics, differing only through their inputs (e.g., observations). This has been shown to improve sample efficiency and facilitates coordination, but may reduce policy diversity and lead to homogenized behaviors [1, 18].

### 2.3 Game-Theoretic Environments

A normal-form game (NFG) is a tuple  $\langle N, \mathbf{A}, \mathbf{u} \rangle$  where  $N = \{1, \dots, n\}$  is a finite set of agents (or players),  $\mathbf{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_n$  is a finite set of *action profiles*  $\mathbf{a} = (a_1, \dots, a_n)$  with  $\mathcal{A}_i$  being the set of player  $i$ ’s actions, and  $\mathbf{u} = (u_1, \dots, u_n)$  is a profile of utility functions  $u_i : \mathbf{A} \rightarrow \mathbb{R}$ . A strategy profile  $\mathbf{s} = (s_1, \dots, s_n)$  is called a *Nash equilibrium* if for each player  $k \in N$ , the strategy  $s_k$  is a best response to the strategies of all the other players  $s_{i \in N \setminus \{k\}}$  [24, 25].

► **Stag-Hunt** is a two-player NFG in which each player has the actions  $\mathcal{A}_1 = \mathcal{A}_2 = \{Stag, Hare\}$ . If they coordinate and both play *Stag*, each will get a large reward. However, if only one plays *Stag*, it gets a penalty (low/negative utility). Alternatively, either agent may unilaterally play *Hare* to receive a small positive utility regardless of the actions of the other agent. An example Stag-Hunt *payoff matrix* (a representation of the agents’ utility functions) is seen in Figure 1a.

► **Extended Public Goods Game (EPGG)** is a mixed-motive simultaneous game that represents the *cooperation vs. competition* social dilemma [28]. An EPGG is a tuple  $\langle N, c, \mathbf{A}, f, u \rangle$ , where  $N = \{1, \dots, n\}$  is the set of players,  $c_i \in \mathbb{R}_{\geq 0}$  is the amount of coin each player  $i$  is endowed with and collected in  $c = (c_1, \dots, c_n)$ , and  $f \in \mathbb{R}_{\geq 0}$  is the multiplication factor for the lump sum endowment (hence the name ‘extended’, as opposed to the case  $f \leq n$ ). Each player  $i \in N$  decides whether to invest in the public good (*cooperate*) or not (*defect*), i.e.,  $\mathcal{A}_i = \{C, D\}$ . The resulting quantity is then evenly distributed among all agents. The utility function for  $i$  is defined as  $u_i : \mathbf{A} \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}$ , with:  $r_i(\mathbf{a}, f, c) = \frac{1}{n} \sum_{j=1}^n c_j I(a_j) \cdot f + c_i(1 - I(a_i))$  where  $\mathbf{a}$  is the action profile,  $a_j$  is the  $j$ -th entry of  $\mathbf{a}$ ,  $I(a_j)$  is the indicator function returning 1 if the action of the agent  $j$  is cooperative and 0 otherwise, and  $c_j$  denotes the  $j$ -th entry of  $c$ . Here, EPGG is formulated as a partially observable stochastic game in which  $f := f_t \sim \mathcal{N}(\mu, \sigma)$ , sampled every time step  $t$ , where  $\mathcal{N}(\mu, \sigma)$  is a normal distribution with mean  $\mu$  and variance  $\sigma$ . Depending on the value of  $\mu$ , the game is expected to be non-cooperative ( $\mu < 1$ ), cooperative ( $\mu > n$ ) or a mix of both (i.e., mixed-motive for  $1 < \mu < n$ ). The *expected Nash equilibrium* is determined by  $\mu = \mathbb{E}_{f_t \sim \mathcal{N}(\mu, \sigma)}[f_t]$ , and can be estimated empirically ( $\hat{\mu}$ ) by taking the mean of several observations of  $f_t$ . A generic two-player EPGG payoff matrix is shown in Figure 1b.

► **Centipede** is a two-player extensive-form game (i.e., the game has several states) in which two agents take turns deciding whether to continue the game (increasing the potential rewards for both) or defect (ending the game and collecting a short-term reward). The game structure incentivizes short-term defection, as each agent risks being defected upon by their partner. Formally, the game begins with a ‘pot’  $p_0$  (a set amount of utility) where each player has two actions,  $\mathcal{A}_i = \{Continue, Stop\}$ . After each round  $t$ , the pot increases, e.g., linearly (such as  $p_{t+1} \leftarrow p_t + 1$ ) or exponentially (such as  $p_{t+1} \leftarrow p_t^2$ ). If a player plays *Stop*, then it receives  $p_t/2 + 1$  while the other player receives  $p_t/2 - 1$ . If both players play *Continue*, they split the pot equally after a certain amount of rounds  $t_{max}$ , each receiving  $p_{t_{max}}/2$ . An instance of the game ( $t_{max} = 6$ ,  $n = 2$ ,  $p_0 \leftarrow 2$ ,  $p_{t+1} = p_t + 2$ ) is given in Figure 1c.

► **Markov Stag-Hunt** is a grid-world environment inspired by Peysakhovich and Lerer [30], where agents move through a grid and must decide whether to hunt a stag (with risk of penalty if hunting

alone) or harvest a plant, with an underlying reward structure similar to the *Stag-Hunt* game. Figure 1d visualizes this environment.

## 2.4 Probabilistic Logic Shielding

We summarize Yang et al. [46]’s approach known as *Probabilistic Logic Shielding* (PLS), a method for incorporating probabilistic safety in RL, and refer the reader to the original work for a detailed treatment. Unlike earlier rejection-based shields that fully block unsafe actions and limit exploration [2], PLS reduces the probability of unsafe actions proportional to their risk, allowing them to occasionally occur. Under soft constraints, this lets agents learn from unsafe actions.

Assume a model  $P$  such that  $P(\text{safe}|s, a)$  represents the likelihood that taking action  $a$  in state  $s$  is safe. The safety of  $\pi$  is the sum of the disjoint probabilities of the safety of taking each action:

$$P_\pi(\text{safe} | s) = \sum_{a \in \mathcal{A}} P(\text{safe} | s, a) \cdot \pi(a | s) \quad (1)$$

By marginalizing the base policy over the safety model, we obtain a reweighted policy that favors safer actions – this is called *probabilistic shielding*. Formally, given a base policy  $\pi$  and a probabilistic safety model  $P$ , the shielded policy  $\pi^+$  is constructed as:

$$\pi^+(a | s) = P_\pi(a | s, \text{safe}) = \frac{P(\text{safe} | s, a)}{P(\text{safe} | s)} \pi(a | s) \quad (2)$$

Succinctly, PLS re-normalizes the action distribution to make unsafe actions less likely. Yang et al. [46] chose to implement the model  $P$  in ProbLog [10] for a number of reasons, including that it is easily differentiable and allows easy modeling and planning. This means our safety constraints are to be specified *symbolically* through PL predicates and relations. Within this paper, a *shield* is thus synonymous with a ProbLog program  $\mathcal{T}$  (further details in Section 3.3), which induces a probabilistic measure  $\mathbf{P}_\mathcal{T}$ . We can (conditionally) query this program to give us various information including the action safety (for each action  $a$ ) as  $P(\text{safe}|s, a) = \mathbf{P}_\mathcal{T}(\text{safe}|a)$ , the safety of the whole policy under  $s$  as  $P_\pi(\text{safe}|s) = \mathbf{P}_\mathcal{T}(\text{safe})$  and the safe/shielded policy  $\pi^+$  as  $P_\pi(a|s, \text{safe}) = \mathbf{P}_\mathcal{T}(a|\text{safe})$ .

## 3 Probabilistic Logic Shields for Multi-Agent Reinforcement Learning

We present a general SMARL framework in Section 3.1, applying (probabilistic) shielding to multi-agent algorithms. Next, in Section 3.2, we introduce PLTD, which incorporates logical constraints directly into the TD-learning process. Finally, Section 3.3 outlines the encoding of safety constraints as ProbLog programs.

### 3.1 General Framework Description

A schematic diagram of a parallel SMARL setup is in Figure 1. Each agent  $p_i$  (indexed with  $i \in \{1, \dots, n\}$ ) observes a possibly subjective state  $s_i^t$  at time  $t$ , and sends both the computed policy  $\pi_i(s_i^t)$  and relevant safety-related information about  $s_i^t$  to its shield  $\mathcal{T}_i$ . Each  $\mathcal{T}_i$  is a ProbLog program that is used to compute a safe policy  $\pi_i^+(s_i^t)$  according to Eq. 2 from which an action  $a_i^t$  is sampled. Finally, the joint action profile  $\vec{a}^t$  is sent to the environment  $Env$ . After updating its state,  $Env$  sends new observations  $s^{t+1}$  and rewards  $\bar{r}^{t+1}$  to the agents, and the cycle continues with  $t \leftarrow t + 1$  until a terminal state is reached. The *agent-environment-cycle interaction* scheme, used often in turn-based games [38] can likewise be extended naturally to the SMARL framework by having each agent interact with their own shield before sampling an action.

Though SMARL is agnostic to the underlying MARL algorithm, we experiment with two: IQL and IPPO (introduced in Section 2.2). IQL with parameter sharing will be called *Parameter Sharing IQL (PSQL)*, and IPPO will be called either *Critic Sharing IPPO (CSPPO)* or *Actor-Critic Sharing IPPO (ACSPPO)*, depending on whether the actors and/or critics are shared. In **Probabilistic Logic SMARL (PL-SMARL)**, at least one of the RL agents uses PLS, optionally sharing policies and parameter updates. PL-SMARL algorithms will be denoted ‘*Shielded X*’ ( $SX$ ); e.g., ‘*Shielded IQL*’, ( $SIQL$ ) when  $X = \text{IQL}$ , or ‘*Shielded CSPPO*’, ( $SCSPPO$ ) when  $X = \text{CSPPO}$ .

### 3.1.1 Safety Guarantees

Following Definition 2.1, we define relative safety for SMARL:

**Definition 3.1** (Relative SMARL Safety). A joint policy  $\vec{\pi}^+ = \langle \pi_1^+, \pi_2^+, \dots, \pi_n^+ \rangle$  is defined to be **at least as safe** as another joint policy  $\vec{\pi} = \langle \pi_1, \pi_2, \dots, \pi_n \rangle$  if and only if for all agents  $i \in N$  with respective shields  $\mathcal{T}_i$ , it is the case that  $P_{\pi_i^+}(\text{safe}|s) \geq P_{\pi_i}(\text{safe}|s)$  for all reachable states  $s \in S$ ; and is **strictly safer** if additionally for at least one  $j \in N$  and  $s \in S$ ,  $P_{\pi_j^+}(\text{safe}|s) > P_{\pi_j}(\text{safe}|s)$ .

Based on Definition 3.1, we attain the following proposition:

**Proposition 3.2.** A PL-SMARL algorithm with agents  $i \in N$  with respective policies  $\pi_i^+$  and shields  $\mathcal{T}_i$  has a joint shielded policy  $\vec{\pi}^+$  at least as safe as their base joint policy  $\vec{\pi}$ , and **strictly safer** whenever any individual base policy  $\pi_j$  violates  $\mathcal{T}_j$ .

A proof is provided in the supplementary materials. This is similar to the first case of ElSayed-Aly et al. [13]’s correctness proof of the safety of factored shields. However, unlike factored shields, PLS does not guarantee *per-step* safety; rather, it ensures that the shielded algorithm’s safety is higher than that of the base algorithm in expectation.

## 3.2 PL-Shielded Temporal Difference Learning (PLTD)

Yang et al. [46] define their shielded algorithm, probabilistic logic policy gradients (PLPG), as an extension of policy gradient methods, specifically, PPO [33]. Thus, to use it with DQN [27], SARSA [48] (or other TD-learning methods), formalize the use of a ProbLog shield in conjunction with TD-learning. An important consideration for extending PLS to TD-learning is whether the algorithm is *on-policy* or *off-policy*. Yang et al. [46] note that for off-policy algorithms to converge, the exploration and learned policies must cover the same state-action space – this assumption must be handled with care in the design of exploration strategies. We propose both on- and off-policy algorithms that incorporate safety constraints into Q-learning.

### 3.2.1 Objective Function

PLPG includes a *shielded policy gradient*  $[\nabla_\theta \log P_{\pi^+}(\text{safe}|s)]$  and a *safety gradient penalty*  $[-\nabla_\theta \log P_{\pi^+}(\text{safe}|s)]$ . Since TD-methods do not use policy gradients, we must rely on a modified version of the latter to introduce safety constraints into the objective function.

Let  $\mathcal{D}$  be the distribution of  $d = \langle s^t, a^t, r^t, s^{t+1}, a^{t+1} \rangle$  tuples extracted from a history buffer of the agent interacting with the MDP and  $s^t, a^t, r^t$  be the state, taken action, and reward at time  $t$ . Let the Q-value approximator  $Q_\theta(s, a)$  be parameterized by some parameters  $\theta$ . The off-policy loss (Q-learning based) and on-policy loss (SARSA-based) are augmented using the aforementioned safety penalty term. We refer to these methods as *Probabilistic Logic TD Learning* (PLTD).

**Definition 3.3** (Probabilistic Logic TD Learning). The PLTD minimization objective is:

$$\mathcal{L}^{Q^+}(\theta) = \mathbb{E}_{d \sim \mathcal{D}} \left[ \left( r^t + \gamma \mathcal{X} - Q_\theta(s^t, a^t) \right)^2 - \alpha S_P \right] \quad (3)$$

where  $\mathcal{X} = \max_{a'} Q_\theta(s^{t+1}, a')$  for off-policy, and  $\mathcal{X} = Q_\theta(s^{t+1}, a^{t+1})$  for on-policy DQN;  $S_P = \log P_{\pi^+}(\text{safe}|s)$  is the safety penalty; and  $\alpha \in \mathbb{R}_{\geq 0}$  is the safety coefficient, or the weight of the safety penalty.

The quantity  $[-\log P_{\pi^+}(\text{safe}|s)]$  is interpreted as *the probability that  $\pi^+$  satisfies the safety constraints* [46]. Under perfect sensor information, PLTD reduces to vanilla TD-learning within the reachable set of safe states. We thus get the following convergence property (following Tsitsiklis and Van Roy [39]):

**Proposition 3.4.** *PLTD, i.e. Eq. 3 converges to an optimal safe policy given perfect safety information in all states in tabular and linear function approximation settings.*

We prove this in the supplementary. For neural Q-networks, global convergence is currently unsolved, though we can adopt common DQN stabilization tricks (summarized in [19]). While we observe empirical stability of PLTD, a complete theoretical analysis is left for future work.

### 3.3 Shield Construction using ProbLog

A shield in PLS is defined using a ProbLog program [10], denoted  $\mathcal{T}$ . Each consists of three components: (i) an annotated disjunction  $\Pi_s$  of predicates whose values are set to a base action distribution  $\pi$ , (ii) a set of constraint-related inputs describing the current state  $\mathbf{H}_s$ , and (iii) a set of sentences  $\mathcal{KB}$  (for *knowledge base*) that specify the agent’s constraints, including a definition for a predicate ‘safe\\_next’ (the predicted safety of the next state under  $\pi$ ). Again, ‘safety’ broadly refers to any probabilistic constraint satisfaction goal (ref. Sec 2.1).

An exemplary shield (Shield 1) for constraining agent behavior towards the mixed Nash equilibrium in Stag-Hunt games is described as such: The probabilistic valuations of the ‘action( $\cdot$ )’ predicates are inferred from  $\pi$ . The ‘sensor( $\cdot$ )’ predicates are set to the normalized absolute difference between the precomputed mixed Nash strategy and the mean of the agent’s historical actions. The predicate ‘safe\\_next’ is inferred via ProbLog semantics [9] and determines the safety of the next state under  $\pi$  – safer states are those where the stated normalized absolute differences are low. For brevity, we describe subsequent shields in-line rather than displaying its full ProbLog source. For details on how we designed the shields for each experiment, we kindly direct the reader to the supplementary.

```

1 % actions
2 action(0)::action(stag);
3 action(1)::action(hare).
4 % sensors
5 sensor_value(0)::sensor(stag_diff).
6 sensor_value(1)::sensor(hare_diff).
7 % safety constraints
8 unsafe_next :- action(stag),
9         sensor(stag_diff).
10 unsafe_next :- action(hare),
11         sensor(hare_diff).
12 safe_next :- \+unsafe_next.

```

**Shield 1.** Shield ( $\mathcal{T}_{mixed}$ ) for mixed *Stag-Hunt* equilibrium.

#### 3.3.1 Deterministic Shields

Using PLS, agents may be made to be deterministic which can be useful when training agents in environments requiring hard constraints. This can be formulated as the following (proof in the supplementary):

**Proposition 3.5.** *For any game there exists a shield  $\mathcal{T}$  such that the resulting shielded agent deterministically selects a single action  $a \in \mathcal{A}$  for each state  $s \in S$  if the shielded policy  $\pi^+$  computes all other actions  $a' \in \mathcal{A} \setminus \{a\}$  as perfectly unsafe.*

## 4 Results

We organize our experiments around key multi-agent phenomena relevant to several open MARL challenges. Rather than focusing on games individually, each section centers on a particular strategic challenge (e.g., equilibrium selection, social dilemmas) and uses one or more games as illustrative testbeds. For the sake of clarity, details are moved to the supplementary material, such as implementation details, shield programs for each experiment, a validation of PLTD in a single-agent experiment, all hyperparameters and training curves. All displayed results are averaged over 5 random seeds to account for variability in training. All reported errors correspond to standard deviations across runs and respective episodes/steps. Our code is open-sourced at <https://github.com/satchitchatterji/ShieldedMARL>.

### 4.1 Coordination and Equilibrium Selection (Stag-Hunt)

We use *Stag-Hunt* to test the ability of PLS to guide agents towards normative behaviors in games with multiple Nash equilibria. IPPO serves as the baseline, while SIPPO agents are equipped with a shield constraining the agents towards either the pure cooperative ( $\mathcal{T}_{pure}$ ) or mixed Nash equilibrium ( $\mathcal{T}_{mixed}$ ) – play *Stag/Hare* 60%/40% of the time with an expected utility of 2.6.  $\mathcal{T}_{pure}$  simply tells the agent to not play *Hare*, making it a deterministic shield according to Proposition 3.5.  $\mathcal{T}_{mixed}$ , used as an example in Section 3.3, defines safety as the absolute difference between the precomputed mixed Nash strategy and the mean of the agent’s historical actions.

**Table 1.** Results for *Stag-Hunt* with PPO-based agents over the last 50 training episodes.  $\bar{r}$  is the mean return and *cooperation* is defined as  $t_{max}^{-1} \sum_{t=0}^{t_{max}} \mathbf{P}_{\mathcal{T}_{pure}}(\text{safe}|s^t)$ .

Algorithm	$\bar{r}$ (train)	$\bar{r}$ (eval)	cooperation
IPPO	$1.99 \pm 0.03$	$1.99 \pm 0.02$	$0.01 \pm 0.01$
SIPPO ( $\mathcal{T}_{pure}$ )	$5.00 \pm 0.00$	$5.00 \pm 0.00$	$1.00 \pm 0.00$
SIPPO ( $\mathcal{T}_{mixed}$ )	$2.57 \pm 0.48$	$2.63 \pm 0.43$	$0.58 \pm 0.08$

Table 1 shows the results this setting. The IPPO agents converge to the non-cooperative Nash equilibrium, consistently playing *Hare* with a reward of 2 per step. In contrast, agents using  $\mathcal{T}_{pure}$  quickly and reliably play *Stag*, earning a higher reward of 5. The unshielded agents never converge to the mixed Nash equilibrium, even though PPO is capable of learning stochastic policies – small deviations push these agents toward playing pure strategies, often resulting in the non-cooperative equilibrium. However, shielded PPO agents using  $\mathcal{T}_{mixed}$  successfully adopt the mixed strategy, though with high variability due to this attracting nature of the pure equilibria.

### 4.2 Temporal Coordination (*Centipede*)

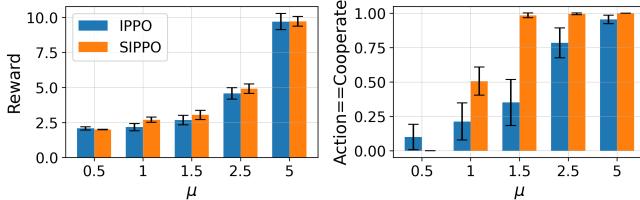
We use *Centipede* to test how well SMARL guides agents to cooperate to achieve large collective long-term rewards, even in scenarios where the dominant strategy suggests defection. A shield was constructed ( $\mathcal{T}_{continue}$ ) to encourage agents to play the game by constraining against early defection. Table 2 shows results for pairs of MARL

agents learning to play the *Centipede* game. IPPO agents occasionally learn to play *Continue* through all  $t_{max} = 50$  steps within 500 episodes. In contrast, we see the SIPPO agents playing the full game from the start of training.

**Table 2.** Results for *Centipede* with PPO- and DQN-based agents over the last 50 training episodes. Safety is defined as  $\frac{1}{T} \sum_{t=0}^T \mathbf{P}_{\mathcal{T}_{cent}}(\text{safe}|s^t)$  and  $R_{ep} = \sum_{t=0}^{t_{max}} r^t$ .

Algorithm	$R_{ep}$ (train)	$R_{ep}$ (eval)	Safety
IPPO	42.35±36.62	42.83±35.81	0.83±0.23
SIPPO	100.50±0.00	100.50±0.00	1.00±0.00
IQL ( $\epsilon$ -greedy)	34.62±46.59	34.70±46.53	0.68±0.23
SIQL ( $\epsilon$ -greedy)	100.50±0.00	100.50±0.00	1.00±0.00
IQL (softmax)	1.73±1.01	30.10±38.61	0.73±0.21
SIQL (softmax)	100.50±0.00	100.50±0.00	1.00±0.00

For DQN-based agents,  $\epsilon$ -greedy and softmax exploration policies were tested. Unshielded  $\epsilon$ -greedy agents fail to progress far, as early on in exploration,  $\epsilon \approx 1$  ( $\pi$  is uniform), and thus a 25% probability of the agents going to the second stage of the game. The probability of reaching some early stage  $\ell$  is  $\approx 0.25^{\ell-1}$ . This discourages exploration, as stopping early provides a small but consistent reward, aligning with the Nash/subgame-perfect equilibrium. Unshielded softmax agents perform slightly better, but with high variance – some agent pairs cooperate while most exit early. This highlights the influence of the choice of exploration strategy, and how using shield can guide desirable behavior – all SIQL variations learn to play *Continue* consistently.



**Figure 4.** Results for the final 10 episodes for the two-player *Extended Public Goods Game* for PLPG-based agents.

### 4.3 Social Dilemmas in Stochastic, Mixed-Motive Environments (Extended Public Goods Game)

We use EPGG to investigate how PLS can promote cooperation in environments with stochastic social incentives. Our experiments address two central questions: (i) *can agents be made to learn strategies that reflect long-term expectations rather than short-term payoffs?*, and (ii) *how does the effectiveness of shielding scale when only a subset of agents is shielded, and under different degrees of parameter sharing?* In the first experiment, we assess how shielding can steer agents toward the *expected* Nash equilibrium of the EPGG. In the second, we evaluate how the degree of parameter sharing affects emergent cooperation.

#### 4.3.1 Learning Long-Term Normative Behavior

We begin with a 2-player setting to isolate the effect of shielding on equilibrium selection under payoff uncertainty. This controlled setup enables a clear comparison between shielded agents learning long-term behavior versus unshielded agents reacting to stochastic payoffs. Limiting the population size removes confounding factors

such as social influence and allows for direct game-theoretic analysis, providing a clean baseline before scaling to larger populations.

To test whether PLS can guide agents toward long-term cooperation, we designed a shield ( $\mathcal{T}_{EPGG}$ ) that directs agents to follow the expected Nash equilibrium, based on an empirical estimate  $\hat{\mu}$  of the stochastic payoff multiplier  $f_t \sim \mathcal{N}(\mu, 1)$ . This contrasts with the unshielded baseline, where agents maximize the instantaneous payoff.

Figure 4 reports results for  $\mu \in \{0.5, 1, 1.5, 2.5, 5.0\}$ . Unshielded IPPO agents reliably converge to the instantaneous equilibrium – defecting when  $f_t < 1$  and cooperating when  $f_t > 2$ . In contrast, SIPPO agents quickly converge to the desired behavior, leading to more prosocial behavior under uncertainty. We observe that SIPPO agents exhibit lower reward variance and more stable cooperation compared to the baseline, especially in mixed-motive cases ( $1 < \mu < 2$ ). For  $\mu = 0.5$ , the shielded agents correctly learn to defect, but earn slightly less due to foregoing occasional cooperative gains from high  $f_t$ . Overall, shielded agents match or exceed the baseline in reward and demonstrate more consistent cooperation.

#### 4.3.2 Partial Shielding and Parameter Sharing

To focus on how shield coverage and parameter sharing influence group behavior, we use a simplified shield that always selects the cooperative action. Unlike the previous experiment, this avoids computing expected equilibria and isolates the effect of shielding structure. Importantly, even though the shield is deterministic, shielded agents still receive gradient-based learning signals. This allows safety information to influence shared parameters, enabling unshielded agents to adopt cooperative behavior indirectly.

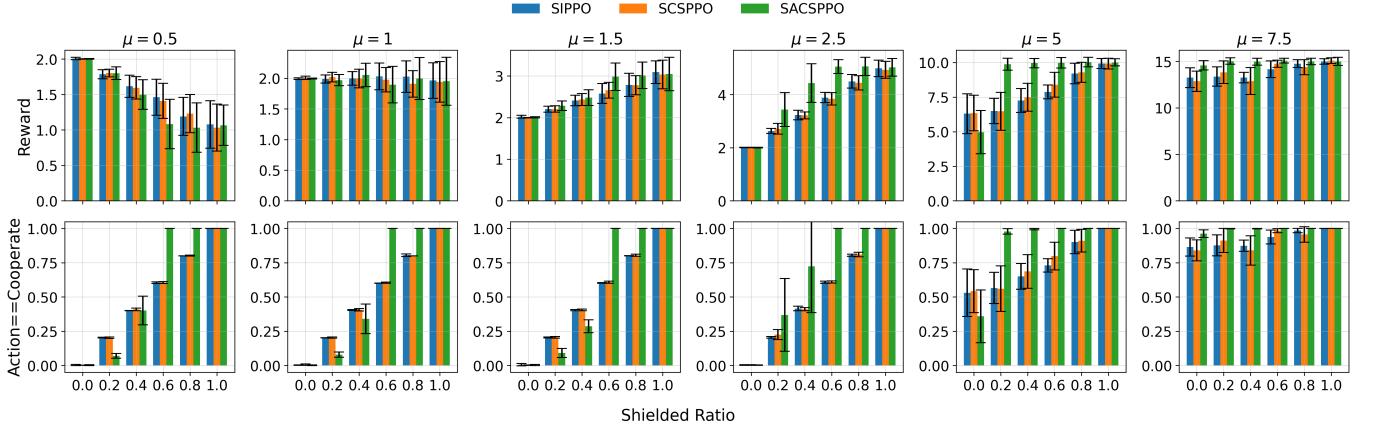
We run EPGG with  $n = 5$  agents, where a subset  $k \in \{0, \dots, 5\}$  of agents are shielded with a deterministic policy that always cooperates (reported as a *shielded ratio*= $k/n$ ). We compare SIPPO, SCSPPPO, and SACSPPO to test how different levels of parameter sharing modulate the influence of the shielded agents on the unshielded population.

Figure 3 shows grouped bar charts for  $\mu \in \{0.5, 1, 1.5, 2.5, 5, 7.5\}$ . Results show a clear trend: increasing the number of shielded agents leads to higher overall cooperation and greater episodic reward in the mixed-motive range ( $1 < \mu < 5$ ). Furthermore, this effect is amplified by parameter sharing: SACSPPO consistently achieves the same or higher cooperation than SCSPPPO and SIPPO at the same shielded ratio. In fully cooperative settings ( $\mu = 7.5$ ), all configurations converge to high cooperation, but SACSPPO still exhibits the lowest variance. In competitive settings ( $\mu = 0.5$ ), cooperation remains low for the baseline algorithms, as expected, increasing with the number of shielded agents – this results in an increasingly lower mean reward with the benefit of the agents instead cooperating. Notably, in the competitive regime, SACSPPO exhibits the highest level of cooperation among the methods, indicating that shared learning with shielded agents can drive prosocial behavior in unshielded agents – even when it reduces their individual rewards.

These results demonstrate that probabilistic shields can influence unshielded agents via shared parameters, serving as an effective conduit for prosocial behavior in multi-agent social dilemmas.

### 4.4 Spatio-Temporal Coordination with Imperfect Information (Markov Stag-Hunt)

We use the *Markov Stag-Hunt* environment [30] to investigate how PLS-SMARL enables coordination in sequential, stochastic, and partially observable multi-agent settings. This environment extends the normal-form *Stag-Hunt* to a spatial domain where agents must learn



**Figure 3.** Results for the final 10 episodes of the 5-player EPGG for SIPPO, SCSPPPO, and SACSPPO agents with varying  $\mu$  and ratios of shielded agents.

when and how to cooperate. Three experiments are conducted: (i) we study the effect of shield constraint strength on agent behavior, (ii) we evaluate how cooperation scales with population size, (iii) finally we explore how cooperation emerges when only a subset of agents is shielded, highlighting the benefits of shielded parameter sharing.

#### 4.4.1 Variation in Shield Strength

To isolate the effect of constraint strength on emergent cooperation, we compare two shields:  $\mathcal{T}_{weak}$  and  $\mathcal{T}_{strong}$ . These were constructed such that  $\mathcal{T}_{weak} \subset \mathcal{T}_{strong}$  in terms of their safety constraints. The former constrains agents to cooperate only when both are near a stag, while the latter more strictly promotes cooperation regardless of global position. Both shields are applied in a 2-agent setting using the SIPPO algorithm, with unshielded IPPO as a baseline.

In this experiment, analyzing just reward or safety does not fully capture cooperative behavior, so the key metrics displayed in Table 3 are: total plant harvests per episode, successful cooperative stag hunts, and unsuccessful attempts to hunt a stag alone. Additionally, we show the total return for each episode, and mean safety for shielded agents.

As expected, IPPO agents avoid cooperation (thereby avoiding risk) and steadily harvest plants for a small, reliable reward. SIPPO agents using  $\mathcal{T}_{weak}$  show moderate improvement, achieving occasional stag hunts while maintaining similar plant collection. Their increased cooperation also results in more penalties, as agents are willing to risk solo hunts. Under  $\mathcal{T}_{strong}$ , cooperation improves dramatically: agents prioritize stag hunting and harvest fewer plants, yet earn more than five times the episodic reward compared to SIPPO with  $\mathcal{T}_{weak}$ . This suggests that well-designed stronger shields not only induce safer behavior, but may lead to better coordination and long-term returns.

#### 4.4.2 Scaling Cooperation with Population Size

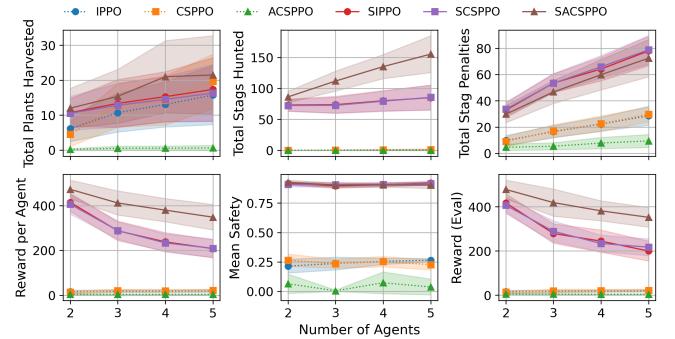
We next examine how cooperation and safety scale with group size when all agents are shielded using  $\mathcal{T}_{strong}$ . We vary the number of agents  $n \in \{2, 3, 4, 5\}$  and compare three unshielded baselines (IPPO, CSPPO, ACSPPPO) and three shielded variants (SIPPO, SCSPPPO, SACSPPO). Metrics include total plants harvested, stags hunted, solo hunt penalties, mean episodic safety, and per-agent reward.

Figure 5 shows that unshielded agents (IPPO, CSPPO, ACSPPPO) consistently avoid cooperation, harvesting plants for low-risk, low-reward outcomes. Notably, ACSPPPO – despite its fully shared architecture – performs worst overall, with near-zero safety and minimal

rewards, underscoring that parameter sharing alone is insufficient to induce prosocial behavior.

In contrast, all shielded variants show increasing cooperation as population size grows. SACSPPO agents, in particular, demonstrate the strongest cooperative intent: they achieve the highest number of stag hunts, while also incurring more penalties, reflecting increased willingness to take coordination risks. Despite these penalties, SACSPPO yields the highest rewards and very high adherence to the shield’s definition of safety constraints. While per-agent rewards tend to decrease with larger populations, this is expected given that key environment parameters – such as the number of stags and plants, grid size, and episode length – remain fixed across all settings.

These results demonstrate that PLS scales robustly with population size. Notably, with full parameter sharing, SACSPPO leverages shared gradients and consistent safety signals to generalize cooperation more effectively across agents, even in complex multi-agent settings.



**Figure 5.** Results for the final 50 episodes of Markov Stag-Hunt with all agents shielded with  $\mathcal{T}_{strong}$ , for  $n \in \{2, 3, 4, 5\}$ .

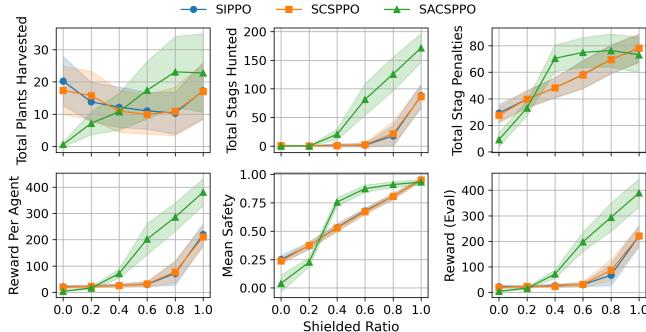
#### 4.4.3 Partial Shielding and Parameter Sharing

Finally, we investigate cooperation in partially shielded populations. Fixing the population size at  $n = 5$ , we vary the number of shielded agents  $k \in \{0, 1, \dots, 5\}$  and compare SIPPO, SCSPPPO, and SACSPPO. All shielded agents use the strong shield  $\mathcal{T}_{strong}$ , and unshielded agents remain unconstrained. This experiment tests how shielded behavior propagates through shared parameters in populations where only a subset of agents receive explicit safety constraints.

Figure 6 shows that cooperation, as measured by total stags hunted, increases monotonically with the number of shielded agents across

**Table 3.** Results for 2-player *Markov Stag-Hunt* with shielded and unshielded PPO-based agents. Columns show: total plants harvested, stags killed, penalties from solo hunts, episodic reward ( $R_{\text{ep}} = \sum_{t=0}^{t_{\max}} r^t$ ), and mean episodic safety ( $t_{\max}^{-1} \sum_{t=0}^{t_{\max}} \mathbf{P}_{T_{(\cdot)}}(\text{safe}|s^t)$ ).

Algorithm	$\sum_{\text{plants}}$	$\sum_{\text{stags}}$	$\sum_{\text{penalties}}$	$R_{\text{ep}} (\text{train})$	$R_{\text{ep}} (\text{eval})$	Mean safety
IPPO	18.74±4.98	0.09±0.11	5.50±2.04	13.71±4.97	16.07±5.80	–
SIPPO ( $T_{\text{weak}}$ )	18.73±4.26	12.80±7.41	14.25±3.57	68.48±35.19	79.80±38.33	0.96±0.01
SACSPPO ( $T_{\text{strong}}$ )	11.33±4.08	77.32±8.82	11.07±3.25	386.86±46.50	393.07±65.27	0.95±0.01



**Figure 6.** Results for the final 50 episodes of the *Markov Stag-Hunt* with 5 agents, varying the number of shielded agents using the strong shield  $T_{\text{strong}}$ .

all methods. SACSPPO shows the strongest effect: even with just two shielded agents, it significantly outperforms the other methods in cooperative behavior and safety adherence. As more agents are shielded, SACSPPO reaches a high level of cooperation, but also accumulates the most stag penalties – suggesting increased risk-taking and cooperative intent, even when coordination occasionally fails.

Reward and evaluation performance trend similarly. While all methods improve with greater shield coverage, SACSPPO consistently yields the highest returns, particularly at higher shielded ratios. This indicates that the safety-driven cooperative strategies learned via PLS are not only prosocial but also effective (reward-maximizing). In contrast, SIPPO and SCSPPO agents remain more conservative: they hunt fewer stags, incur fewer penalties, and achieve lower overall rewards.

Plant harvesting does not uniformly decrease with increasing shielded ratio across all methods. While SIPPO and SCSPPO show a modest decline – suggesting a shift from self-interested strategies to coordinated risk-taking – SACSPPO exhibits an unexpected increase in plant harvesting as more agents are shielded. Taken together with SACSPPO’s high shield adherence, this may indicate that stronger parameter sharing drives homogenization that improves reward maximization, encouraging agents to opportunistically exploit lower-risk strategies. Alternatively, the behavior could reflect an emergent division of labor, where agents already near the stag focus on hunting while others stabilize returns via plant harvesting. Distinguishing between these hypotheses would require further analysis of agent roles and policy diversity. Importantly, mean safety steadily increases with shield coverage in all methods, but SACSPPO achieves the highest safety levels – highlighting its superior ability to align unshielded agents with shielded behavior through shared learning.

Together, these results demonstrate that PLS can induce prosociality even under partial shielding, and that shared parameters are a key conduit for propagating safety-aligned behavior to unshielded agents.

## 5 Related Work

In safe (MA)RL, several recent frameworks address safety during learning and execution [17]. Gu et al. [16], propose **constrained Markov games** and **MACPO**, but without formal language semantics. Subramanian et al. [35] integrate **PL neural networks** as the

function approximator within agents to affect interpretable top-down control; however, in our method, the agents may use any function approximator, maximizing potential learning flexibility. Waga et al. [41] use **Mealy machines** to approximate a world model to ensure safety, leading to some inherent symbolic interpretability. Other related concepts include **dynamic shielding**, for instance, ElSayed-Aly et al. [13] who introduce two frameworks that dynamically enforce safety constraints – our own method is similar to their **factored shields**. Banerjee et al. [4], Zhang et al. [47] and Xiao et al. [45] are recent works that use the notion of (dynamic) **model predictive shielding**. Though these show promising results, these do not benefit from the interpretability of formal language semantics. Melcer et al. [26] explore **decentralized shielding** (similar to factored shields), i.e. allowing each agent to have its own shield, reducing computational overhead and enhancing scalability. [26] and [13] use **LTL as formal language**, though with a high knowledge requirement of the underlying MDP – this is in contrast to our use of PLS which requires only safety-related parts of the states. Finally, some methods, such as Wang et al. [42], enforce safety through **natural language constraints**; in contrast, our approach builds on PLS to provide stronger formal guarantees.

## 6 Discussion and Conclusion

In this article, we introduced Probabilistic Logic Shielded MARL, a novel set of DQN- and PPO-based methods that extend PLS to MARL. Several classes of games were analyzed and shown to benefit from SMARL in terms of safety and cooperation, and theoretical guarantees of these methods were presented.

Despite the benefits implied by our results, a few limitations and extensions must be reflected upon: ▶ First, current methods for solving PL programs become computationally expensive as the state and action spaces grow [14]. However, as with factored shields [13], our method scales only linearly with the number of agents. ▶ The use of *a-priori* constraints in PLS provides *predefined* safety knowledge, which is straightforward for simple environments but becomes complex in dynamic, real-world systems. This approach limits scalability, relying on expert-defined safety measures. Most existing dynamic shielding methods do not use formal language semantics, and thus a balance must be struck between autonomous shield updating and formal verification. ▶ The complexity of the ProbLog program in PLS impacts performance and behavior, as seen in the Markov Stag-Hunt experiments where the stronger shield promoted more cooperation but at the cost of increased complexity w.r.t. hand-designed rules. It may be interesting to examine how varying shield complexity affects computational load and safety outcomes. ▶ Partial shielding was explored where shielded agents update their parameters with safety gradients, while unshielded agents focus on maximizing rewards, optionally sharing parameters. A weighted shared parameter update may balance safety and reward optimization dynamically. ▶ Finally, while PL-SMARL aims to enhance safety, its implementation must carefully consider ethical and societal impacts, as hand-designed constraints may introduce biases, under-specified definitions, or system vulnerabilities that current research has yet to fully address.

## Acknowledgements

This publication is part of the project ‘Hybrid Intelligence: augmenting human intellect’ (<https://hybrid-intelligence-centre.nl/>) with project number 024.004.022 of the research programme ‘Gravitation’ which is (partly) financed by the Dutch Research Council (NWO).

## References

- [1] S. V. Albrecht, F. Christianos, and L. Schäfer. Multi-agent Reinforcement Learning: Foundations and Modern Approaches. *Massachusetts Institute of Technology: Cambridge, MA, USA*, 2023.
- [2] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. Safe Reinforcement Learning via Shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [3] R. Axelrod. Effective Choice in the Prisoner’s Dilemma. *Journal of Conflict Resolution*, 24(1):3–25, 1980.
- [4] A. Banerjee, K. Rahmani, J. Biswas, and I. Dillig. Dynamic Model Predictive Shielding for Provably Safe Reinforcement Learning. *Advances in Neural Information Processing Systems*, 37:100131–100159, 2024.
- [5] R. Bloem, B. Könighofer, R. Könighofer, and C. Wang. Shield Synthesis: Runtime Enforcement for Reactive Systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 533–548. Springer, 2015.
- [6] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm Robotics: A Review from the Swarm Engineering Perspective. *Swarm Intelligence*, 7:1–41, 2013.
- [7] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym, 2016.
- [8] S. Carr, N. Jansen, S. Junges, and U. Topcu. Safe Reinforcement Learning via Shielding Under Partial Observability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14748–14756, 2023.
- [9] F. G. Cozman and D. D. Mauá. On the Semantics and Complexity of Probabilistic Logic Programs. *Journal of Artificial Intelligence Research*, 60:221–262, 2017.
- [10] L. De Raedt, A. Kimmig, and H. Toivonen. Problog: A Probabilistic Prolog and Its Application in Link Discovery. In *IJCAI*, volume 7, pages 2462–2467, 2007.
- [11] C. S. De Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. Torr, M. Sun, and S. Whiteson. Is Independent Learning All You Need in the Starcraft Multi-agent Challenge? *arXiv preprint arXiv:2011.09533*, 2020.
- [12] F. Den Hengst, V. François-Lavet, M. Hoogendoorn, and F. van Harmelen. Planning for Potential: Efficient Safe Reinforcement Learning. *Machine Learning*, 111(6):2255–2274, 2022.
- [13] I. ElSayed-Aly, S. Bharadwaj, C. Amato, R. Ehlers, U. Topcu, and L. Feng. Safe Multi-Agent Reinforcement Learning via Shielding. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 483–491, 2021.
- [14] D. Fierens, G. Van den Broeck, J. Renkens, D. Shterionov, B. Gutmann, I. Thon, G. Janssens, and L. De Raedt. Inference and Learning in Probabilistic Logic Programs using Weighted Boolean Formulas. *Theory and Practice of Logic Programming*, 15(3):358–401, 2015.
- [15] S. Ganesh, N. Vadori, M. Xu, H. Zheng, P. Reddy, and M. Veloso. Reinforcement Learning for Market Making in a Multi-Agent Dealer Market. *arXiv preprint arXiv:1911.05892*, 2019.
- [16] S. Gu, J. G. Kuba, Y. Chen, Y. Du, L. Yang, A. Knoll, and Y. Yang. Safe Multi-Agent Reinforcement Learning for Multi-Robot Control. *Artificial Intelligence*, 319:103905, 2023.
- [17] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, and A. Knoll. A Review of Safe Reinforcement Learning: Methods, Theories and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [18] J. K. Gupta, M. Egorov, and M. Kochenderfer. Cooperative Multi-Agent Control using Deep Reinforcement Learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017.
- [19] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [20] N. Hunt, N. Fulton, S. Magliacane, T. N. Hoang, S. Das, and A. Solar-Lezama. Verifiably Safe Exploration for End-to-End Reinforcement Learning. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.
- [21] N. Jansen, B. Könighofer, S. Junges, A. Serban, and R. Bloem. Safe Reinforcement Learning using Probabilistic Shields. In *31st International Conference on Concurrency Theory (CONCUR 2020)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2020.
- [22] jemaw. gym-safety, July 2019. URL <https://github.com/jemaw/gym-safety>. commit hash: 468831a2bae112454a8954ab702198d0c69ff50f.
- [23] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [24] K. Leyton-Brown and Y. Shoham. *Essentials of Game Theory: A Concise Multidisciplinary Introduction*. Springer Nature, 2022.
- [25] M. Maschler, S. Zamir, and E. Solan. *Game Theory*. Cambridge University Press, 2020.
- [26] D. Melcer, C. Amato, and S. Tripakis. Shield Decentralization for Safe Multi-Agent Reinforcement Learning. *Advances in Neural Information Processing Systems*, 35:13367–13379, 2022.
- [27] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602*, 2013.
- [28] N. Orzan, E. Acar, D. Grossi, and R. Rădulescu. Emergent Cooperation under Uncertain Incentive Alignment. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’24, page 1521–1530, Richland, SC, 2024. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9798400704864.
- [29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32, 2019.
- [30] A. Peysakhovich and A. Lerer. Prosocial Learning Agents Solve Generalized Stag Hunts Better than Selfish Ones. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS ’18, page 2043–2044. International Foundation for Autonomous Agents and Multiagent Systems, 2018.
- [31] A. Prakash. pz\_dilemma, November 2021. URL [https://github.com/arjun-prakash/pz\\_dilemma](https://github.com/arjun-prakash/pz_dilemma). commit hash: 7df435f83cf0917359ecc8847dd8b474df18fa5c.
- [32] L. D. Riek. Healthcare Robotics. *Communications of the ACM*, 60(11):68–78, 2017.
- [33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [34] S. Shalev-Shwartz, S. Shamir, and A. Shashua. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving. *arXiv preprint arXiv:1610.03295*, 2016.
- [35] C. Subramanian, M. Liu, N. Khan, J. Lenchner, A. Amarnath, S. Swaminathan, R. Riegel, and A. Gray. A Neuro-Symbolic Approach to Multi-Agent RL for Interpretability and Probabilistic Decision Making. *arXiv preprint arXiv:2402.13440*, 2024.
- [36] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.
- [37] M. Tan. Multi-agent Reinforcement Learning: Independent vs. Cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, 1993.
- [38] J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente, et al. PettingZoo: Gym for Multi-Agent Reinforcement Learning. *Advances in Neural Information Processing Systems*, 34:15032–15043, 2021.
- [39] J. Tsitsiklis and B. Van Roy. Analysis of Temporal-Difference Learning with Function Approximation. *Advances in Neural Information Processing Systems*, 9, 1996.
- [40] E. van der Sar, A. Zocca, and S. Bhulai. Multi-Agent Reinforcement Learning for Power Grid Topology Optimization. *arXiv preprint arXiv:2310.02605*, 2023.
- [41] M. Waga, E. Castellano, S. Pruekprasert, S. Klikovits, T. Takisaka, and I. Hasuo. Dynamic Shielding for Reinforcement Learning in Black-box Environments. In *International Symposium on Automated Technology for Verification and Analysis*, pages 25–41. Springer, 2022.
- [42] Z. Wang, M. Fang, T. Tomilin, F. Fang, and Y. Du. Safe Multi-Agent Reinforcement Learning with Natural Language Constraints. *arXiv preprint arXiv:2405.20018*, 2024.
- [43] C. J. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.
- [44] B. P. Welford. Note on a Method for Calculating Corrected Sums of Squares and Products. *Technometrics*, 4(3):419–420, 1962.
- [45] W. Xiao, Y. Lyu, and J. Dolan. Model-based Dynamic Shielding for Safe and Efficient Multi-agent Reinforcement Learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’23, page 1521–1530, Richland, SC, 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9798400704864.

- Systems*, pages 1587–1596, 2023.
- [46] W. Yang, G. Marra, and L. De Raedt. Safe Reinforcement Learning via Probabilistic Logic Shields. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5739–5749, 2023.
- [47] W. Zhang, O. Bastani, and V. Kumar. MAMPS: Safe Multi-agent Reinforcement Learning via Model Predictive Shielding. *arXiv preprint arXiv:1910.12639*, 2019.
- [48] D. Zhao, H. Wang, K. Shao, and Y. Zhu. Deep Reinforcement Learning with Experience Replay Based on SARSA. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6. IEEE, 2016.

## Supplementary Material for Analyzing Probabilistic Logic Shields for Multi-Agent Reinforcement Learning

The supplementary material is organized in the following way:

- Appendix A: Theoretical proofs of propositions.
- Appendix B: In-depth discussion of environments used, including implementation details, reward functions and settings.
- Appendix C: Details on shields including their ProbLog source codes and process of construction
- Appendix D: Single-agent experiment used to as an initial indication of PLTD’s safety and efficacy.
- Appendix E: Table and explanation of all hyperparameters used.
- Appendix F: Partial training/reward curves.

## A Proofs

### A.1 Proof of Proposition 3.2

**Proposition A.1.** *A PL-SMARL algorithm with agents  $i \in N$  with respective policies  $\pi_i^+$  and shields  $\mathcal{T}_i$  has a joint shielded policy  $\bar{\pi}^+$  at least as safe as their base joint policy  $\bar{\pi}$ , and strictly safer whenever any individual base policy  $\pi_j$  violates its shield  $\mathcal{T}_j$ .*

*Proof.* Let  $\Pi_i$  be the space of all policies for each agent  $i \in N$ . We then recall from Yang et al. [46]’s Proposition 1 that  $P_{\pi_i^+}(\text{safe}|s) \geq P_{\pi_i}(\text{safe}|s)$ ,  $\forall \pi_i \in \Pi_i$  and  $\forall s \in S$ . Thus, according to the safety specifications in  $\mathcal{T}_i$ , every  $\pi_i^+$  is at least as safe as  $\pi_i$ , and thus  $\bar{\pi}^+$  is at least as safe as  $\bar{\pi}$ .

If all base policies  $\pi_i \in \bar{\pi}$  already satisfy their respective shields  $\mathcal{T}_i$  in all reachable states (i.e.,  $P_{\pi_i}(\text{safe}|s) = 1$  for all  $s$ ), then shielding leaves the policy unchanged:  $\pi_i^+ = \pi_i$ . Otherwise, if any  $\pi_j \in \bar{\pi}$  does not abide by  $\mathcal{T}_j$ , by construction, the shielded policy will assign higher probability to safe actions in expectation, i.e.  $P_{\pi_j^+}(\text{safe}|s) > P_{\pi_j}(\text{safe}|s)$ , and the resulting joint policy  $\bar{\pi}^+$  will be strictly safer than  $\bar{\pi}$ .  $\square$

### A.2 Proof of Proposition 3.4

First, for completion, we prove a lemma that states that any shield with perfect information (binary information about the valuations of the `sensor(.)` predicates) always yield  $P_{\mathcal{T}}(\text{safe} | s, a) \in \{0, 1\}$ . Then we define a more specific form of Proposition 3.4 and show that PLTD converges under perfect sensor information and regular TD-learning assumptions.

**Lemma A.2** (Perfect-information shield yields binary safety). *Let  $\mathcal{T}$  be a ProbLog shield that, at a given state  $s$ , receives action facts `action(a)` whose probabilities are the policy  $\pi(a | s)$ , and sensor facts `sensor(l)` that are deterministic in  $s$  (i.e. we have perfect sensor information), i.e.,  $P_{\mathcal{T}}(\text{sensor}(l) = \text{true}) \in \{0, 1\}$  and  $\text{sensor}(l) = \text{true} \iff l \text{ actually holds in } s$ .*

Suppose  $\mathcal{T}$ ’s knowledge base contains, for each ground action  $a$ , a clause of the schematic form

$$\text{unsafe\_next} :- \text{action}(a), \text{sensor}(\ell_a). \quad (4)$$

where the literal  $\ell_a$  may be one of:

- i. a state-dependent ‘hazard’ literal, or
- ii. an always-true dummy literal (i.e.  $\text{unsafe\_next} :- \text{action}(a)$ ).

Together with the complementary rule

$$\text{safe\_next} :- \neg \text{unsafe\_next},$$

the shield satisfies

$$P_{\mathcal{T}}(\text{safe\_next} | \text{action}(a)) \in \{0, 1\} \quad \text{for every } a. \quad (5)$$

Define the state-wise safe-action set

$$A_{\text{safe}}(s) = \{a \mid P_{\mathcal{T}}(\text{safe\_next} | \text{action}(a)) = 1\}. \quad (6)$$

Then

$$P_{\mathcal{T}}(\text{safe} | s, a) = \begin{cases} 1, & a \in A_{\text{safe}}(s); \\ 0, & a \notin A_{\text{safe}}(s). \end{cases} \quad (7)$$

*Proof.* Condition the ground program on `action(a)`. All remaining random facts are the deterministic `sensor(l)` atoms, so the program is now fully deterministic. Hence the queried probability is 1 if and only if the rule for `unsafe_next` fails.

*Case (a).* If  $\ell_a$  is false in  $s$  the rule body fails, so `safe_next` holds with certainty.

*Case (b).* If  $\ell_a$  is always true, the rule body succeeds, so `safe_next` is false with certainty.

Thus the conditional probability is always 0 or 1, and partitioning actions by the value 1 yields the stated set  $A_{\text{safe}}(s)$ .  $\square$

**Proposition A.3** (PLTD Convergence under Perfect Safety Information). *Let  $\mathcal{M} = \langle S, A, P, r, \gamma \rangle$  be a finite MDP with  $\gamma < 1$  and let  $\mathcal{T}$  be a PL shield with perfect sensor information. Assume the agent*

- i. [GLIE] follows a time-varying policy  $\pi_t$  such that (a)  $\pi_t(a | s) = 0$  for all  $a \notin A_{\text{safe}}(s)$ , and (b)  $\pi_t \xrightarrow{t \rightarrow \infty} \bar{\pi}^+$  while every safe pair  $(s, a)$  is selected infinitely often (e.g.  $\epsilon_t$ -greedy on  $A_{\text{safe}}(s)$  with  $\epsilon_t \downarrow 0, \sum_t \epsilon_t = \infty$ ).
- ii. stores a tabular action-value function  $Q_\theta$ , and
- iii. updates  $\theta$  by minimising the PLTD loss (Eq. 3.3) with a stepsize sequence  $\{\eta_t\}$  satisfying  $\sum_t \eta_t = \infty$  and  $\sum_t \eta_t^2 < \infty$  (regular TD-learning assumption, Sutton and Barto [36]).

Then, with probability 1, the parameter sequence  $\{\theta_t\}$  converges to a fixed point  $Q_{\text{safe}}^*$  that is optimal over the set of safe policies, i.e.

$$Q_{\text{safe}}^*(s, a) = \max_{\pi: \pi(a' | s) = 0 \vee a' \notin A_{\text{safe}}(s)} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a \right]. \quad (8)$$

*Proof.* Fix a finite MDP  $\mathcal{M} = \langle S, A, P, r, \gamma \rangle$  with  $\gamma < 1$  and a ProbLog shield  $\mathcal{T}$  whose sensor atoms are deterministic in the current state  $s$ ; by Lemma A.2 the shield therefore induces binary state-action safety probabilities. Throughout, let  $A_{\text{safe}}(s) \subseteq A$  denote the set of  $s$ -safe actions singled out in the lemma.

**1. The PLTD loss collapses to the ordinary TD loss.** For any state-action pair  $(s, a)$ ,

$$P_{\mathcal{T}}(\text{safe} \mid s, a) = \mathbf{1}\{a \in A_{\text{safe}}(s)\}. \quad (9)$$

Consequently  $P_{\pi^+}(\text{safe} \mid s) = 1$  for every  $s$  that can be encountered under the behaviour policy  $\pi^+$  (assumption i). Thus the safety penalty  $S_P = \log P_{\pi^+}(\text{safe} \mid s)$  in Eq. 3.3 vanishes identically and the PLTD objective reduces to

$$\mathcal{L}^{Q^+}(\theta) = \mathbb{E}_{d \sim \mathcal{D}}[(r^t + \gamma \mathcal{X} - Q_\theta(s^t, a^t))^2], \quad (10)$$

namely the usual squared one-step TD error (with  $\mathcal{X}$  equal to  $\max_{a'} Q_\theta(s^{t+1}, a')$  in the off-policy variant or  $Q_\theta(s^{t+1}, a^{t+1})$  in the on-policy variant).

**2. Behaviour-target consistency.** The agent acts with  $\pi^+$  and evaluates TD targets under  $\pi^+$ , so there is no off-policy mismatch; the TD recursion is on-policy in the SARSA setting and uses the same transition-probability matrix in the Q-learning setting.

**3. Almost-sure convergence of the tabular TD iterates.** Because  $S, A$  and therefore the post-shield state-action space  $\{(s, a) \mid a \in A_{\text{safe}}(s)\}$  are finite, rewards are bounded, and the stepsizes  $\{\eta_t\}$  satisfy  $\sum_t \eta_t = \infty$  and  $\sum_t \eta_t^2 < \infty$  (assumption iii), the classical stochastic-approximation proofs of Watkins and Dayan [43] for Q-learning or Sutton and Barto [36] for SARSA apply *verbatim*. A minor notational change is that the maximisation (in the off-policy case) is taken only over safe actions, but the contraction argument is identical. Under the usual further condition that every safe state-action pair is visited infinitely often (assumption i) the parameter sequence  $\{\theta_t\}$  converges with probability 1 to the unique fixed point  $Q^{\pi^+}$  of the Bellman operator corresponding to  $\pi^+$ .

**4. Optimality of the limit among safe policies.** By construction  $\pi^+$  places zero probability on unsafe actions and is *greedy* with respect to  $Q^{\pi^+}$  over each safe-action set  $A_{\text{safe}}(s)$ :

$$\pi^+(a \mid s) > 0 \implies a \in \operatorname{argmax}_{a' \in A_{\text{safe}}(s)} Q^{\pi^+}(s, a'). \quad (11)$$

Standard policy-evaluation/-improvement arguments (e.g. Sutton and Barto [36]) now imply that  $Q^{\pi^+}$  coincides with the *optimal* action-value function over the restricted policy class  $\Pi_{\text{safe}} = \{\pi \mid \pi(a' \mid s) = 0 \text{ whenever } a' \notin A_{\text{safe}}(s)\}$ . We denote this limit by  $Q_{\text{safe}}^*$  and  $\forall (s, a)$  obtain

$$Q_{\text{safe}}^*(s, a) = \max_{\pi \in \Pi_{\text{safe}}} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a \right], \quad (12)$$

**5. Putting it together.** Steps 1–4 establish that  $Q_\theta$  converges to  $Q_{\text{safe}}^*$  as claimed.  $\square$

**Remark A.4.** As stated in the main paper, for tabular and linear function-approximation settings, Lemma A.2 together with the projected TD results of Tsitsiklis and Van Roy [39] yields convergence of PLTD. For deep neural Q-networks, global convergence is currently not known; in that regime we can adopt the usual stabilisation tricks (target network, replay buffer, Double-DQN). Empirically, we can show that PLTD remains stable across tasks, but a complete theoretical treatment is left for future work.

### A.3 Proof of Proposition 3.5

**Proposition A.5.** For any game there exists a shield  $\mathcal{T}$  such that the resulting shielded agent deterministically selects a single action  $a \in \mathcal{A}$  for each state  $s \in S$  if the shielded policy  $\pi^+$  computes all other actions  $a' \in \mathcal{A} \setminus \{a\}$  as perfectly unsafe.

*Proof.* Assume a game  $\mathcal{E}$  that has actions  $\mathcal{A}$  for a given agent and  $|\mathcal{A}| = n_{\mathcal{A}}$ . This agent has a policy  $\pi(s) = \langle a_0, \vec{a}_{-a_0} \rangle \in \mathbb{R}_{\geq 0}^{n_{\mathcal{A}}}$ , where  $\vec{a}_{-a_0}$  is the tuple of action values except  $a_0$ , such that  $\sum \pi(s) = 1$ . We recall from Definition ... that for any agent with a policy  $\pi$ , a safe policy  $\pi^+$  is computed as:

$$\pi^+(a \mid s) = P_\pi(a \mid s, \text{safe}) = \frac{P(\text{safe} \mid s, a)}{P(\text{safe} \mid s)} \pi(a \mid s)$$

We now construct a shield  $\mathcal{T}$  in ProbLog. Let `action(a0)` be a probabilistic fact in  $\mathcal{T}$  that represents  $\pi(a_0 \mid s)$  and with constraints  $\mathcal{KB}$  as follows:

```

1  action(0)::action(a0);
2  ... % optional: other actions
3
4  % optional: sensor inputs
5
6  % safety specification
7  unsafe_next :- action(X), X\=a0.
8  safe_next :- \+unsafe_next.
```

Note that the shield is state independent. Using the semantics of PL, we see that  $P(\text{safe} \mid s, a') = 0$  for  $a' \neq a_0$ , i.e. there is no state in which the safety of any other action apart from  $a_0$  at all safe. Now, we can compute  $\pi^+(a_0 \mid s)$  and determine that  $P(\text{safe} \mid s) = 1 - \sum \vec{a}_{-a_0} = a_0 = \pi(a_0 \mid s)$ , leaving us with  $\pi^+(a_0 \mid s) = P_\pi(\text{safe} \mid s, a_0) = 1$ . Thus, we get a safe deterministic policy (pure strategy)  $\pi^+(s) = \langle 1, 0, \dots \rangle$ , sampling from which, the agent will always play  $a_0$ .  $\square$

## B Environment Specifics

$a_1$	$a_2$	$a_3$	$a_4$
$b$	$c$	$d$	$e$
$a$	$f$	$g$	$h$

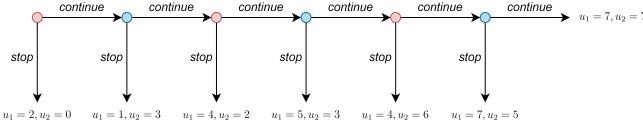
**Figure 7.** Generic payoff matrix

### B.1 Repeated NFGs (Stag-Hunt)

In these experiments both agents choose an action simultaneously and receive a reward determined by the payoff matrix. With reference to the generic payoff matrix in 7, any game with a payoff matrix with the following structure of payoffs is considered an instance of the *Stag-Hunt*:  $a = b > d = e \geq g = h > c = f$ . Each episode is a series of 25 repeated games. Thus the agents must learn to maximize the discounted reward attained over the whole episode [3]. The three Nash equilibria for the *Stag-Hunt* game instance as in Figure 2 (main paper) are  $(\text{Stag}, \text{Stag})$ ,  $(\text{Hare}, \text{Hare})$  and a mixed Nash equilibrium where each agent plays *Stag* 60% of the time and *Hare* 40% of the time.

## B.2 Centipede Game

The centipede game is a turn-based game that has the following description: the game begins with a ‘pot’  $p_0$  (a set amount of payoff). Each player has two actions, either to *continue* or to *stop*. After each round  $t$ , the pot increases, for example linearly ( $p_{t+1} \leftarrow p_t + 1$ ) or exponentially ( $p_{t+1} \leftarrow p_t^2$ ). If any player decides to *stop* at their turn, they receive a utility of  $p_t/2 + 1$  and the other player gets a utility of  $p_t/2 - 1$ . If both players continue after a set number of rounds  $t_{max}$ , both players split the pot equally, receiving  $p_{t_{max}}/2$ . An instance of a centipede game with  $t_{max} = 6$  turns and  $n = 2$  players is shown in Figure 8. The optimal strategy here is for the very first player to stop. This is called the *subgame perfect Nash equilibrium*, one analogue of the Nash equilibrium for extensive-form games.

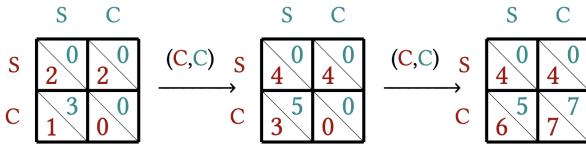


**Figure 8.** Centipede game for 2 players with  $p_0 = 2$  and growth rule  $p_{t+1} \leftarrow p_t + 2$ . Red nodes represent player 1’s decision node and blue ones are player 2’s. The leaves represent the utilities received by each player. The first move is at the root of the tree.

### B.2.1 Environment description

Prakash [31] provides a good starting point with respect to implementation. However, in order to make the structure of the game homogeneous with the other environments experimented with<sup>1</sup>, a simple modification was made that turns the centipede game into a repeated simultaneous game with changing utilities each round.

This is done by combining the two agents’ turns into a 2-player 2-action game, where the player who goes *first* is the *row* player and the *second* player is the *column* player. Let these actions be *S* (for *Stop*) and *C* (for *Continue*). If and only if both agents choose *C*, a new game is created with utilities derived from the next pair of turns, with no reward being given to either player. This transformation is general and can be applied to any *Centipede* game instance. Figure 9 shows an example of this – a repeated NFG version of Figure 8. It follows that we can also verify that the Nash solution (more specifically the subgame-perfect Nash equilibrium) is for the *first (row)* player to play *S* right away regardless of what the *second (column)* player plays (this is their dominant strategy).



**Figure 9.** Repeated simultaneous game form of Centipede shown in Figure 8.

### B.2.2 Experimental conditions

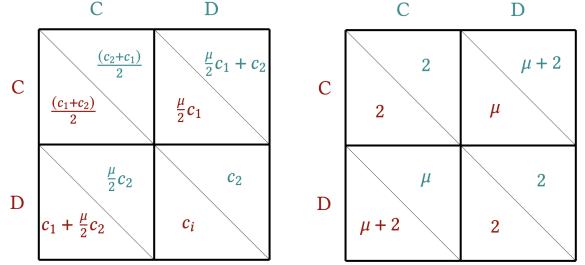
Two agents play this game. The one that has the first move is randomised at the start of each episode. The initial pot is set to  $p_0 = 1$

<sup>1</sup> This is not strictly required but makes the implementation of the Python environment, the interaction with agents, and the shield more straightforward.

and is updated for each successful *Continue* action as  $p_{t+1} \leftarrow p_t + 2$ . If an agent plays *Stop*, they receive a reward of  $p_t/2 + 1$  and the other receives  $p_t/2 - 1$ . The maximum number of steps are set to  $t_{max} = 50$ , with the final rewards being equal for both agents at  $p_{t_{max}}/2$ .

## B.3 Two-Player Extended Public Goods Game

The *Extended Public Goods Game* is an environment where each player  $i$  has an initial amount of money  $c_i$  and can either choose to *Cooperate* (put their money into the pot) or *Defect* (keep the money). The money in the pot is multiplied by a value  $f$ , and the new total is distributed evenly to all players. The Nash equilibrium for a single game depends on the value of  $f$ , and thus the Nash strategy for each agent for a full game is not necessarily dependent on the expected value of  $f$ .



**Figure 10.** Expected payoff matrix for a two-player EPGG with general endowments  $c_1$  and  $c_2$  (left) and the payoff when  $c_1 = c_2 = 2$  (right). For both,  $\mu = \mathbb{E}_{f \sim \mathcal{N}(\mu, \sigma)}[f]$ .

Assume a game with  $n = 2$  agents. At each  $t$ , the expected payoffs can be computed a-priori with respect to taking an action with the knowledge of  $f$  being passed to the agents. Thus, perfectly rational agents maximize their expected reward by learning to cooperate conditioned on the instantaneous  $f$  they receive. It might be interesting to try and coerce the agents to take actions based on the *expected* value of  $f$  instead of instantaneous ones – indeed, if we know the distribution from which  $f$  is sampled (e.g.  $f \sim \mathcal{N}(\mu, \sigma)$  for some  $\mu$  and  $\sigma$ ) and we want the agents to converge to a singular action throughout the episode (i.e. not the time step-wise Nash equilibrium), then the expected payoff matrix can be derived as Figure 10. Here we can see that the *expected* Nash equilibrium over the population of  $f$  is determined by  $\mu = \mathbb{E}_{f \sim \mathcal{N}(\mu, \sigma)}[f]$  and can be estimated empirically by taking the mean of several observations of  $f$ .

### B.3.1 Experimental conditions

Two agents play this game. The agents observe the action of the other agent taken at the previous time step and the current sample of the multiplication factor  $f_t \sim \mathcal{N}(\mu, \sigma)$ . It is expected for rational agents to defect when  $f_t < 1$  and cooperate when  $f_t > 2$  and have mixed incentives otherwise. The game is played for 500 episodes consisting of 25 games each. At each time step, a new  $f_t$  is sampled. In separate experiments  $\mu \in \{0.5, 1.5, 2.5, 5.0\}$ . The standard deviation of the distribution for  $f_t$  is kept at  $\sigma = 1$  always. The initial endowment for each agent is  $c_i = c_j = 2$ .

## B.4 Markov Stag-Hunt

Peysakhovich and Lerer [30] describe a few grid environments that have the property that the optimal policies for agents reflect the struc-

ture of the *Stag-Hunt* game – principally, there are two strategies that the agents can converge to, either take on more risk by cooperating but attain higher rewards, or take on lower risk but get guaranteed small rewards. An optimal strategy may even be some mix of the two, depending on the details of the environment in question.

One of these environments, which they term the *Markov Stag-Hunt* environment has been adapted here to show that adding local probabilistic logic shields to MARL agents can help them converge to a global behavior that corresponds to either of the *underlying* pure Nash equilibria of choice – either the cooperative or the non-cooperative one. It should be stated that the original code for *Markov Stag-Hunt* from [30] is not publicly available, and thus the environment created here is a custom approximation based on the sparse description in their paper. Thus, their results should not be directly compared ours.

#### B.4.1 Environment description

The adapted *Markov Stag-Hunt* environment used within this paper can be described in general as such:  $n$  agents are placed in a square grid world of size  $g \times g$ . The agents have the following action set at each time step to move through the grid with respect to a global direction system: {UP, DOWN, LEFT, RIGHT, NONE}. There are  $n_{stag}$  stags placed in the environment, as well as  $n_{plant}$  plants, each occupying a single position each on the lattice. The agents can have either local or global observations, with or without a ‘self’ token, either as a 3D tensor or flattened one-hot encoded vector depending on the needs and configuration of the experiment.

An agent attempts to ‘hunt’ a stag if it is on the same grid position as the stag, and ‘harvests’ a plant if it is on the same position of a plant. A stag is successfully hunted if there are  $n_{hunt} \geq n_{hunt\_min}$  agents that hunt the same stag at the same time step and each receive a large positive reward  $r_{stag}$ , otherwise, the agents are unsuccessful and receive a penalty (negative reward)  $r_{pen}$ . An agent can harvest a plant alone, and will receive a small reward of  $r_{plant} < r_{stag}$ . Once a plant is harvested or a stag is attempted to be hunted, they reappear in another part of the grid. The stag also has an optional behaviour of moving towards the nearest agent at each time step with probability  $p_{move}$ . The grid exists for a certain amount of time steps  $t_{max}$ .

It is intuitive that if the agents learn to cooperate and attack the stag, they end up with a higher expected reward, with the risk of getting a penalty if the cooperation is inadequate. They may also learn an anti-cooperative strategy of only harvesting plants, which has a low reward but no penalty. This is a similar outcome to the NFG version of the Stag-Hunt, but in a far more complex environment. The first strategy is *high risk - high reward*, and the second is *low risk - low reward*.

#### B.4.2 Experimental conditions

In the experiments conducted, the grid was set to a size of  $5 \times 5$  with  $n = 2$  agents. The number of stags and plants were respectively set to  $n_{stag} = 1$  and  $n_{plant} = 2$ . In order to have a successful hunt, both agents need to hunt the stag at the same time ( $n_{hunt\_min} = 2$ ). The rewards were respectively set to  $r_{stag} = 10$ ,  $r_{pen} = -2$  and  $r_{plant} = 2$ . The stag was set not to move ( $p_{move} = 0$ ) as there is ample stochasticity in the environment already for the sake of initial experimentation (stemming from the random positions and reappearances of the plants and the stag and from the behavior of the agents themselves). Finally, the environment lasted  $t_{max} = 200$  steps.

The agents observe the grid as a concatenated (flattened) vector consisting of the set of one-hot encoded vectors for each position on the grid to describe what is in contained in them: either (i) nothing, (ii) a stag, (iii) a plant, (iv) the observing agent, (v) the other agent, or (vi) both agents. Thus, the full input is a vector of size is  $5 \times 5 \times 6 = 150$ .

## C Details on Shield Construction

### C.1 Stag-Hunt

For the pure equilibria, cooperation would refer to collaborating with the other agent to hunt the stag and `defect` refers to an agent individually going for the hare instead.

```

1 % actions
2 action(0)::action(stag);
3 action(1)::action(hare).
4
5 % safety constraints
6 unsafe_next :- action(hare).
7 safe_next :- \+unsafe_next.
```

**Shield 2.** Shield ( $\mathcal{T}_{pure}$ ) for pure Nash equilibrium for *Stag-Hunt*

For the mixed equilibria, we would need a more sophisticated shield, given the current limitations of the existing implementation of Yang et al. [46]. One way of doing this is to add soft constraints to the actions based on how far they differ from the mixed equilibrium. As a proxy for the policy of the agent, a set of the latest  $h$  historical actions of the agents can be collected – this is termed a *buffer*. A **mean policy**  $\hat{\pi}_h$  at the current time  $t$  can then be derived from these actions, iterating over all  $a' \in \mathcal{A}$ :

$$\hat{\pi}_h(a' | s) = \frac{1}{h} \sum_{i \in \{1, \dots, h\}} \mathbb{I}(a^{t-i} = a') \quad (13)$$

Let an *a-priori* normative policy be  $\pi^*$ , say, the mixed Nash strategy. If  $\hat{\pi}_h$  is too distant from the expected mixed strategy with respect to some divergence measure, then we can re-normalize the input policy using Definition 2 to be more similar to  $\pi^*$ . The divergence measure should be bounded between  $[0, 1]$  in order to feed it into the ProbLog shield program. For *Stag-Hunt*, let such an input be the absolute difference between  $\hat{\pi}_h$  and  $\pi^*$ :

$$\text{sensor(stag\_diff)} := |\pi^*(\text{Stag} | s) - \hat{\pi}_h(\text{Stag} | s)| \quad (14)$$

$$\text{sensor(hare\_diff)} := |\pi^*(\text{Hare} | s) - \hat{\pi}_h(\text{Hare} | s)| \quad (15)$$

We can then construct a shield as follows:

```

1 % actions
2 action(0)::action(stag);
3 action(1)::action(hare).
4
5 % sensors
6 sensor_value(0)::sensor(stag_diff).
7 sensor_value(1)::sensor(hare_diff).
8
9 % safety constraints
10 unsafe_next :- action(stag), sensor(stag_diff).
11 unsafe_next :- action(hare), sensor(hare_diff).
12 safe_next :- \+unsafe_next.
```

**Shield 3.** Shield ( $\mathcal{T}_{mixed}$ ) towards mixed Nash equilibrium for *Stag-Hunt*

The constraints that define the value of the `unsafe_next` predicate can be interpreted as *it is unsafe to take action stag proportional to the value of sensor(stag\_diff)* (and analogously for playing *Hare*). Thus, the higher the difference between the Nash strategy and the historical policy for each agent, the more the policy is normalized.

The policy is not normalized at all when `sensor(stag_diff)` and `sensor(hare_diff)` are both zero. In this case, the overall policy is perfectly safe.

For the experiments discussed below, the buffer length is set to  $h = 50$ . Additionally, under the payoff matrix in Figure 7, the *a priori* mixed Nash equilibrium can be computed such that  $\pi^*(\text{Stag} | s) = 0.6$  and  $\pi^*(\text{Hare} | s) = 0.4$  for both agents. The evaluation safety is computed based on  $\mathcal{T}_{\text{pure}}$  as it directly reflects the actions of the agents (with  $\mathbf{P}_{\mathcal{T}_{\text{pure}}}(\text{Stag} | s) = 1$  and  $\mathbf{P}_{\mathcal{T}_{\text{pure}}}(\text{Hare} | s) = 0$ ).

## C.2 Centipede Game

Since the agents have two actions, we can use a shield to force the agents to always play  $C$  at every time step (regardless of previous actions) until the game terminates and a reward is attained using the following shield.

```

1 % actions
2 action(0)::action(continue);
3 action(1)::action(stop).
4
5 % safety constraints
6 unsafe_next :- action(stop).
7 safe_next :- \+unsafe_next.

```

**Shield 4.** Simple shield  $\mathcal{T}_{\text{continue}}$  for *Centipede*

As a result, it is expected that shielded agents will always continue, as this is the only safe action in any round.

## C.3 Extended Public Goods Game

Since, at each time step  $t$ , there is effectively a new simultaneous game with 2 actions, we can dictate the actions of the agents with a simple shield such as the one used for the *Centipede Game*. Thus, it is trivial to create a shield that dictates the actions of the agents to have a certain policy such as *always play cooperate*, regardless of inputs.

Let us instead create a shield that uses a model  $\hat{\mu}$  of the true  $\mu$  estimated over training so far. Additionally, we can use the standard deviation of the multiplication factors so far,  $\hat{\sigma}$  to model the uncertainty the agent has with respect to the current  $f_t$  compared to  $\hat{\mu}$ . This can be done online without the need of a buffer using Welford's method [44]. We then compare  $\hat{\mu}$  to the predetermined Nash outcomes and decide if it is high enough to warrant defecting – if  $\hat{\mu} \geq 1$ , *cooperate*, else *defect*. In Shield 5, this is set as the (Boolean) value of the fact `sensor(mu_high)` – its value is set to 1 if  $\hat{\mu} \geq 1$  else 0. The agents will err on the side of a cooperative strategy when  $1 < \hat{\mu} < 2$ , where the rational strategy would be mixed.

We then might want to include uncertainty to increase exploitation if it has high certainty about the environment (align the agent more strongly with the inductive bias induced via the shield), and contrariwise, increase exploration if there is high uncertainty (guide the agent more weakly). We must thus construct a smooth distance measure  $d$  between  $f_t$  and  $\hat{\mu}$  bounded between  $[0, 1]$  such that  $d(f_t, \hat{\mu}) = 1$  when  $f_t = \hat{\mu}$  and  $d(f_t, \hat{\mu}) = 0$  when  $|f_t - \hat{\mu}| \rightarrow \infty$ . One of many possibilities is described here. First, we compute the  $z$ -score of the current  $f_t$  with respect to the estimated parameters so far:

$$z(f_t | \hat{\mu}, \hat{\sigma}) = \frac{f_t - \hat{\mu}}{\hat{\sigma}} \quad (16)$$

We then use the CDF of a standard normal distribution to translate this into a probability value  $\Phi(z(f_t | \hat{\mu}, \hat{\sigma}))$ . Subtracting 0.5 centers  $\Phi(z(f_t | \hat{\mu}, \hat{\sigma}))$  around 0 for  $z(f_t | \hat{\mu}, \hat{\sigma}) = 0$ . Doubling the absolute value and subtracting from 1 inverts the scale to make it 1 at the mean

and symmetrically 0 at extreme tails. Thus the final distance function is:

$$d(f_t, \hat{\mu}) = 1 - 2 \cdot |\Phi(z(f_t | \hat{\mu}, \hat{\sigma})) - 0.5| \quad (17)$$

In Shield 5 below, the value of the fact `sensor(f_certainty)` is thus set to  $d(f_t, \hat{\mu})$ . Thus using the inputs defined above, the shield is constructed as follows:

```

1 % actions
2 action(0)::action(cooperate);
3 action(1)::action(defect).
4
5 % sensors
6 sensor_value(0)::sensor(mu_high).
7 sensor_value(1)::sensor(f_certainty).
8
9 % safety constraints
10 unsafe_next :- \+action(cooperate), sensor(mu_high),
11   , sensor(f_certainty).
11 unsafe_next :- \+action(defect), \+sensor(mu_high),
12   , sensor(f_certainty).
12 safe_next :- \+unsafe_next.

```

**Shield 5.** Shield ( $\mathcal{T}_{\text{EPGG}}$ ) for EPGG capturing uncertainty.

The two constraints on lines 10 and 11 essentially say, *it is unsafe if cooperate is not played and  $\hat{\mu} > 1$  with some certainty about  $f_t$  (and analogously for defect)*.

## C.4 Markov Stag-Hunt

Let us focus on constructing a shield that might help the agents learn how to cooperate and hunt a stag together. Similar shields can be constructed for non-cooperative behavior too, however, it is much harder for the agents to converge to the cooperative behavior in the first place.

Since the environment is rather complex, there are a myriad of ways this goal may be achieved, and two are described here. The inputs to the shield are the policy and some sensors that describe relative positions of the stag and the other agent:

```

1 % actions
2 action(0)::action(left);
3 action(1)::action(right);
4 action(2)::action(up);
5 action(3)::action(down);
6 action(4)::action(stay).
7
8 % sensors
9 sensor_value(0)::sensor(left).
10 sensor_value(1)::sensor(right).
11 sensor_value(2)::sensor(up).
12 sensor_value(3)::sensor(down).
13 sensor_value(4)::sensor(stag_near_self).
14 sensor_value(5)::sensor(stag_near_other).

```

**Shield 6.** Partial shield for *Markov Stag-Hunt*.

The value of the sensor predicates `sensor(stag_near_self)` and `sensor(stag_near_other)` is 1 when the stag is adjacent to the acting agent and the other agent respectively, and 0 otherwise. The sensors with literals named after the cardinal directions are binary and represent the relative position of the stag with respect to the agent – for example, `sensor(left)` is 1 when the stag is strictly to the left of the acting agent and 0 otherwise.

A first shield can be constructed that influences the actions of the agents such that they strongly tend to go towards the stag, wait for the other agent, and then hunt the stag together. Such a shield can be implemented as such:

```

1 % actions & sensors go here
2 ...

```

```

3 % define movement towards the stag
4 go_towards_stag :- action(Dir), sensor(Dir).
5 % define a state where both agents are near the
6 % stag
7 stag_surrounded :- sensor(stag_near_self),
8 %           sensor(stag_near_other).
9
10 % 1) it is unsafe to not go towards the stag if it
11 %     not near
12 unsafe_next :- \+go_towards_stag,
13 %           \+sensor(stag_near_self).
14 % 2) it is unsafe to not wait, the stag is near
15 %     without another agent
16 unsafe_next :- \+action(stay), sensor(
17 %           stag_near_self),
18 %           \+sensor(stag_near_other).
19 % 3) it is unsafe to not hunt, the stag is near and
20 %     there is another agent
21 unsafe_next :- \+go_towards_stag, stag_surrounded.
22
23 % combine all unsafe conditions to get safety
24 safe_next :- \+unsafe_next.

```

#### Shield 7. Strong shield ( $\mathcal{T}_{strong}$ ) for *Markov Stag-Hunt*.

This shield will be referred to as the *strong shield* in this environment, since it significantly affects the policy of the agent regardless of its position on the grid.

However, in general, depending on the environment, we may not know how to construct a strongly shielded agent or we may wish to constrain the agent less in order for it to learn optimal policies in a more end-to-end fashion. Thus, in these cases, a shield can be constructed that only significantly effects the agent's policy in a small number of cases. For *Markov Stag-Hunt* another shield created is the following:

```

1 % actions & sensors go here
2 ...
3
4 % define movement towards the stag
5 go_towards_stag :- action(Dir), sensor(Dir).
6 % define a state where both agents are near the
7 % stag
8 stag_surrounded :- sensor(stag_near_self),
9 %           sensor(stag_near_other).
10
11 % it is unsafe not to hunt stag and it is
12 % surrounded
13 unsafe_next :- \+go_towards_stag, stag_surrounded.
14 safe_next :- \+unsafe_next.
15
16 % if the stag is not nearby, any action is fine (
17 %   safety=1)
18 safe_next :- \+sensor(stag_near_self).

```

#### Shield 8. Weak shield ( $\mathcal{T}_{weak}$ ) for *Markov Stag-Hunt*.

This shield will be called the *weak shield* as it affects the policy of an agent that is in the neighborhood of the stag. It only guides the agent to hunt the stag when another agent is nearby, and otherwise allows the policy to explore freely.

The set of safety-related sentences  $\mathcal{BK}$  for the weak shield can be argued to be a subset of that of the strong shield ( $\mathcal{BK}_{weak} \subset \mathcal{BK}_{strong}$ ). An additional line (line 14 in the weak shield) has to be added to ensure that the behaviour of the agent over all the sensor space is well-defined, and to allow any policy to be safe in the complement of  $\mathcal{BK}_{weak}$  with respect to  $\mathcal{BK}_{strong}$  (i.e. the sentences  $\mathcal{BK}_{strong} \setminus \mathcal{BK}_{weak}$ ).

## D Single-Agent Experiment: CartSafe

Before testing the two shielded multi-agent DQN-based algorithms (SIQL and SPSQL), it is imperative to first test whether or not PLTD (ProbLog-shielded DQN) works as well as vanilla DQN for environments with constraints, and how it compares to PLPG (ProbLog-shielded PPO). Additionally, it would be useful to verify that the custom PLPG code works for single agent settings before attempting multi-agent experiments.

*CartSafe* is modification of OpenAI Gym's classical control problem *CartPole* [7] with simple constraints inspired by jemaw [22]. The problem has the following elements: the environment consists of a rod attached to a box or cart at one end and is allowed to freely swing around the attachment point. This setup exists in a bounded 2-dimensional real space with a set width and an active physics simulation. At each time step, the agent has two actions: accelerate the cart to the left or to the right. The agent observes the cart's position, velocity, the angular velocity of the rod and the angle of the rod in radians. The goal is to actively balance the rod upright without leaving the bounds of the environment. The agent receives a reward at each time step based on the angle of the pole ( $\theta_{pole}$ ) with respect to vertical:  $r^t = 1$  if  $-15^\circ \leq \theta_{pole} \leq 15^\circ$  and  $r^t = -1$  otherwise. The environment terminates when a maximum number of steps ( $t_{max}$ ) are reached or the cart moves off screen. The additional constraints that are added to this setup are such that the cart must not enter the region bounded by the leftmost and rightmost quarters of the environment – thus the agent must learn to balance the rod near the center of the screen.

The purpose of this experiment is trifold:

1. To test whether the PLPG and ProbLog shield implementations adequately work on a common RL task with a continuous state space;
2. To verify the ideas of PLTD introduced in Section 3.2 and its implementation to see whether it would help (or at least does not hurt) the ability of a DQN agent converging to a good and safe policy;
3. To explore the possibilities in terms of exploration and on/off-policy strategies to see how they affect performance and safety of PLTD agents.

### D.1 Experimental conditions

The maximum length of each episode was set to  $t_{max} = 200$  with 500 training episodes. The agents are evaluated every 10 episodes. Each algorithm was re-run with 5 different seeds each and their aggregated results are discussed below. The PPO and DQN parameters were minimally hand-tuned and can be found in Table 4 in Appendix E. The PPO agents were shielded with safety penalty coefficient  $\alpha \in \{0.5, 1.0, 2.0\}$ . The DQN-based algorithms were tested by varying the following aspects: on-policy vs. off-policy,  $\epsilon$ -greedy vs. softmax exploration (paired with greedy and softmax exploitation policies respectively) and whether they are shielded (SIQL with safety coefficient  $\alpha \in \{1.0, 5.0\}$ ) or not (IQL).

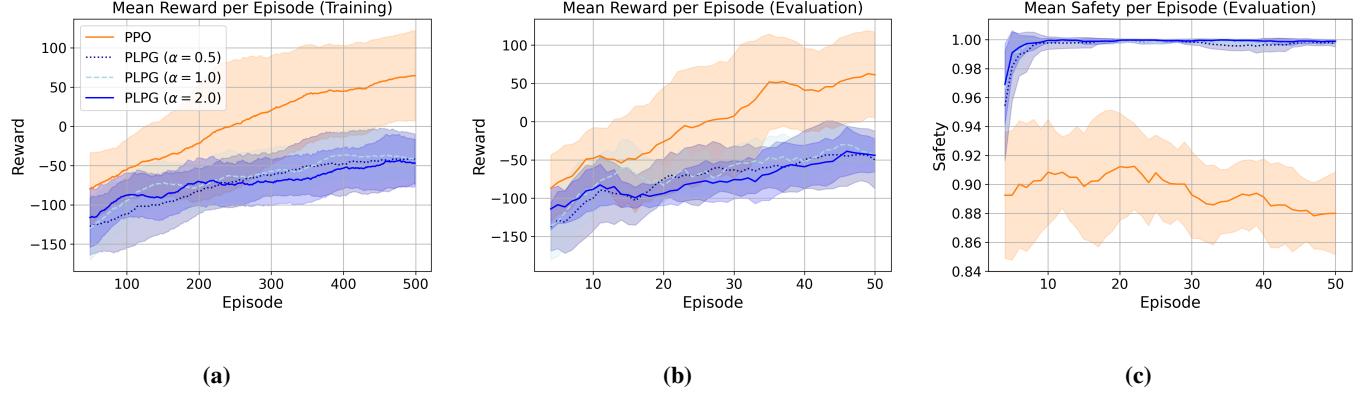
### D.2 Shield construction

A simple shield that aims to satisfy the constraints outlined in the task description can be constructed as follows:

```

1 % actions
2 action(0)::action(left);
3 action(1)::action(right).

```



**Figure 11.** Training, evaluation and safety results for *CartSafe* for PPO-based agents. The lines represent the mean and the shadow the standard deviation over 5 seeds. Results are smoothed using a rolling average with window size 50.

```

4 %
5 % sensors
6 sensor_value(0) ::sensor(cost).
7 sensor_value(1) ::sensor(xpos).
8 sensor_value(2) ::sensor(left).
9 sensor_value(3) ::sensor(right).
10
11 % constraints
12 unsafe_next :- action(X), sensor(X),
13                 sensor(cost), sensor(xpos).
14 safe_next :- \+unsafe_next.

```

#### Shield 9. Shield for soft constraint satisfaction for *CartSafe*.

The actions correspond to moving the cart to the left or right. The sensors here correspond to (in order) whether or not the current state is within a constrained region (Boolean value), the distance from the center on the  $x$ -axis normalized between 0 and 1 (such that it is 0 when the cart is in the middle, and 1 when it is at the right or left edges of the screen), and whether the cart is on the left side or right side of the screen (Boolean value). The constraint can be interpreted as *it is unsafe if an action is taken towards some side X, the cart is on side X, the current state is in the constrained region, and the x-position is sufficiently large*.

This means that if the cart is within the permissible central region of the environment, the safety probability of the next action is computed as  $(\text{sensor}(\text{cost}) = 0) \implies (\text{safe\_next} = 1)$ , and the policy remains unchanged. If the cart is too far right, outside the permissible region, it is unsafe to move the cart towards the right. The *right* action gets progressively more unsafe as the cart gets further to the right. The safe policy will proportionally favor the left action as a result. Analogous behavior occurs if the cart is too far left instead of right.

Note that this also means that the agent is *already* within the constrained region (specifically, at the edge of the region) when the shield takes effect – this can be easily solved by instead having a predicate that looks at whether the *next* step will be impermissible given an action, rather than whether it is *currently* in such a state. However, this should not significantly alter the behavior of the agent in this environment.

Note that *sensor (xpos)* is the *only* non-Boolean input to this shield. If it not placed in the body of the constraint in line 12, since *sensor (X)* and *sensor (cost)* are purely Boolean, the safe policy computation will result in a deterministic behavior in the constrained regions – *go left if the cart is in the right-hand constrained region with probability 1*, and vice-versa for the left-hand constrained

region. Thus, the behavior of the agent act safely immediately and deterministically. The inclusion of *sensor (xpos)* allows us to experiment with soft constraints – here, it enforces the idea that it gets linearly worse when travel deeper into the constrained regions.

### D.3 Results: PLTD (Shielded DQN)

Figure 12 displays training, evaluation and safety results of a variation of PLTD agents. As a reminder, these agents vary in the following aspects: on-policy vs. off-policy,  $\epsilon$ -greedy vs. softmax exploration (paired with greedy and softmax exploitation policies respectively) and whether they are shielded (SIQL with safety coefficient  $\alpha \in \{1, 5\}$ ) or not (IQL).

**Shielding:** First, we may look at whether adding a shield significantly alters the behavior of the agent in this setting. The rightmost column shows the mean safety per episode computed with the *CartSafe* shield (Shield 9). We see that in general, adding a shield to an agent with either of the tested safety penalty values does not significantly affect its safety, *ceteris paribus*. However, all safety values tend to be quite high, generally above 0.9 with not much improvement after the first  $\approx 100$  episodes. We also note that the shielded agents do not do *worse* than the unshielded ones. This may hint at the safety constraints being relatively simple to accomplish given the environment goals and algorithms. Thus there is little to no trade-off between reward and safety in this task.

**$\epsilon$ -Greedy vs. Softmax:** When looking at the difference between the exploration and exploitation strategies, we see quite a large difference in the variation of results when trained multiple times – the  $\epsilon$ -greedy-based algorithms tend to more reliably converge to a good policy (converging around reward  $\approx 100$ ), whereas the softmax policies tend to have a much larger standard deviation with less smoothly increasing curves, to policies that generally attain less reward per episode.

**On vs. Off-Policy:** For the  $\epsilon$ -greedy runs, it is noticeable that the on-policy versions of the algorithms all tends to converge faster than the off-policy ones, reaching around reward  $\approx 50$  in 100 episodes during evaluation versus 200 episodes for off-policy DQN. We do not see the same trend for the softmax policies, it seems that that it does reduce the variance across seeds significantly. There does not seem to be a significant difference in safety between these two conditions. One outlier in this regard is that the off-policy agents with softmax policies are slightly but significantly more safe than all the others.

**PLTD vs. PLPG:** Based on the experiments conducted and discussed, we see that PLPG and PLTD agents act somewhat differently in this setting. None of the PPO-based agents attain the same reward as the DQN-based ones. Adding shields to these agents has a stronger influence the PLPG agents than the PLTD ones – this is not unexpected, as the PLTD agents attain safety information both from the safety penalty term and the safe policy gradient, whereas safety information for PLTD agents is only input through the safety penalty. The PLTD agents tend to do better in terms of rewards, but worse in terms of safety, which is on par with the baseline of the unshielded PLTD agents or the vanilla PPO agent.

#### D.4 Results: PLPG (Shielded PPO)

Figure 11 shows training and evaluation results for four agents that have a PPO-based learning scheme. The first agent is vanilla (unshielded) PPO (the orange line) and the other three are PLPG agents (blue) with varying safety penalties,  $\alpha \in \{0.5, 1, 2\}$ .

We see that for the training reward graphs (showing the total rewards attained per agent per episode), there is a relatively high variation in rewards, but the evaluation graphs show a clear upward trend of the agents learning to balance the pole. The variation is likely due to (at least in part) the choice of hyperparameters. The PPO agent shows better performance based on reward, but with much higher variation, and it does not seem like the algorithms have yet converged after 500 episodes. This trend holds for the evaluations as well (occurring after every 10 episodes) – the PPO agent have better evaluation rewards but with more variation.

When it comes to evaluating safety, we do see a very significant difference – the unshielded PPO agent has a final mean evaluation safety of around 0.88 (learning to balance the pole generally but not always in the center), but all other PLPG agents converge quickly to a nearly perfect safe policy within 10 training episodes across seeds. Increasing the safety penalty  $\alpha$  does not seem to make a significant difference for any of the shielded algorithm results.

## E List of Hyperparameters

Table 4 describes the hyperparameters used in the underlying RL algorithms for all independent MARL experiments. The hyperparameters are:

- **Epochs:** The number of epochs that the neural networks are trained per training call.
- $\gamma$ : The discount factor.
- **Buffer size:** The number of latest steps (states and actions taken) contained in the history buffer.
- **Batch size:** (DQN) The batch size sampled from the buffer per training epoch.
- **lr/lr (actor)/lr (critic):** Learning rate for Q network (DQN), actor/critic networks (PPO). All networks were optimised via Adam [23], with all other hyperparameters left as in PyTorch’s default implementation [29].
- **$\epsilon$ -decay:** (DQN)  $\epsilon$ -decay factor  $\delta$  for  $\epsilon$ -greedy exploration according to the following formula, where  $t$  is the number of exploration steps taken so far:

$$\epsilon \leftarrow \delta^t \quad (18)$$

- $\epsilon_{min}$ : (DQN) Minimum value of  $\epsilon$  for  $\epsilon$ -greedy exploration.
- $\tau$ : (DQN) Exploration factor for softmax exploration.
- **VF coef:** (PPO) Value-function loss coefficient.
- **S coef:** (PPO) Entropy loss coefficient.

- **$\epsilon$ -clip:** (PPO) Clipped policy update ratio range.
- $\alpha$ : Safety penalty coefficient (cf. Definition 3.3 and Yang et al. [46]’s Definition 5.1).

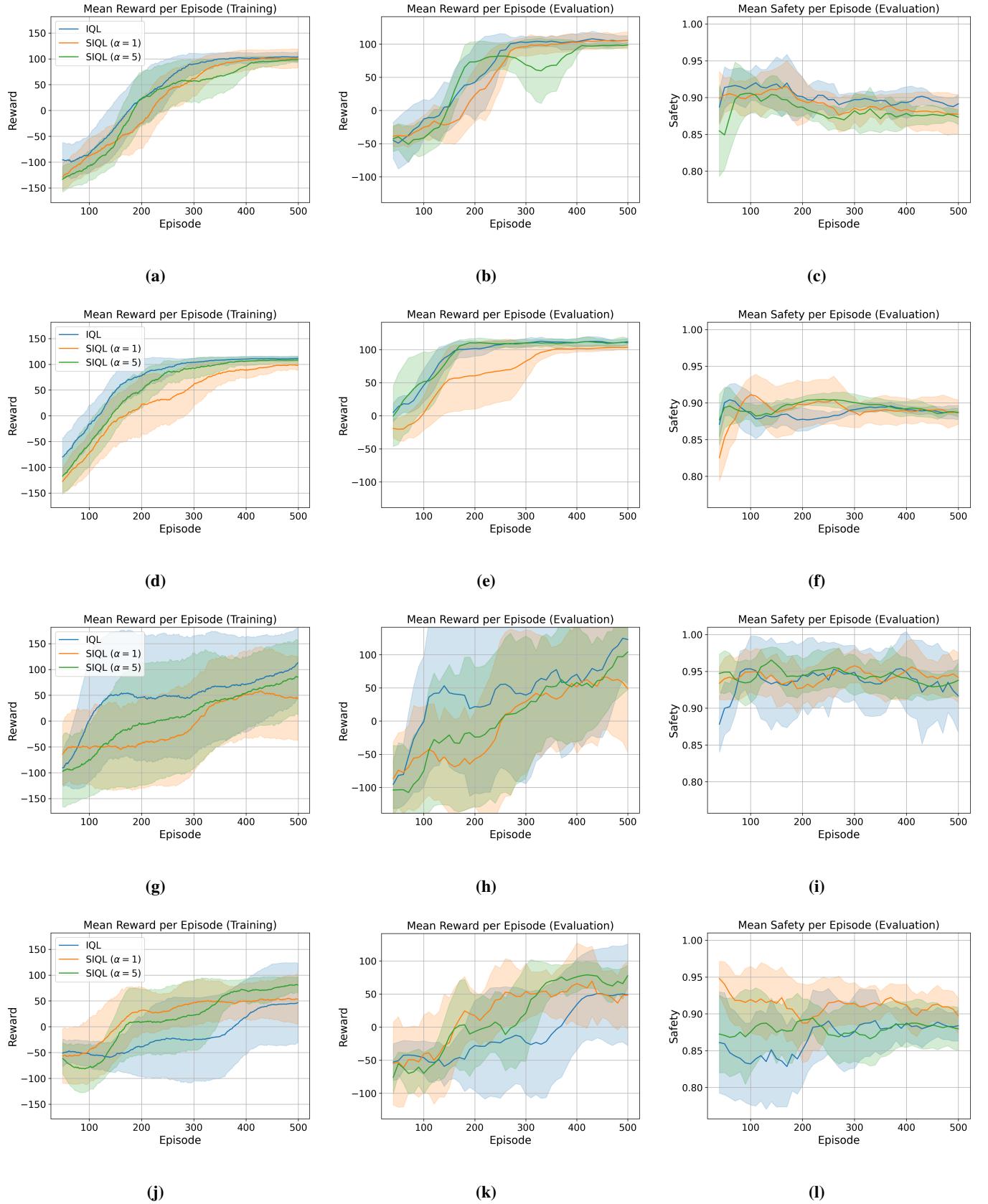
Additionally, all DQN agents used neural networks with 2 hidden layers of 64 neurons each with a ReLU activation in between each layer, and all PPO agents used the same for actor and critic networks, except using the hyperbolic tangent instead of ReLU between layers.

## F Training Curves

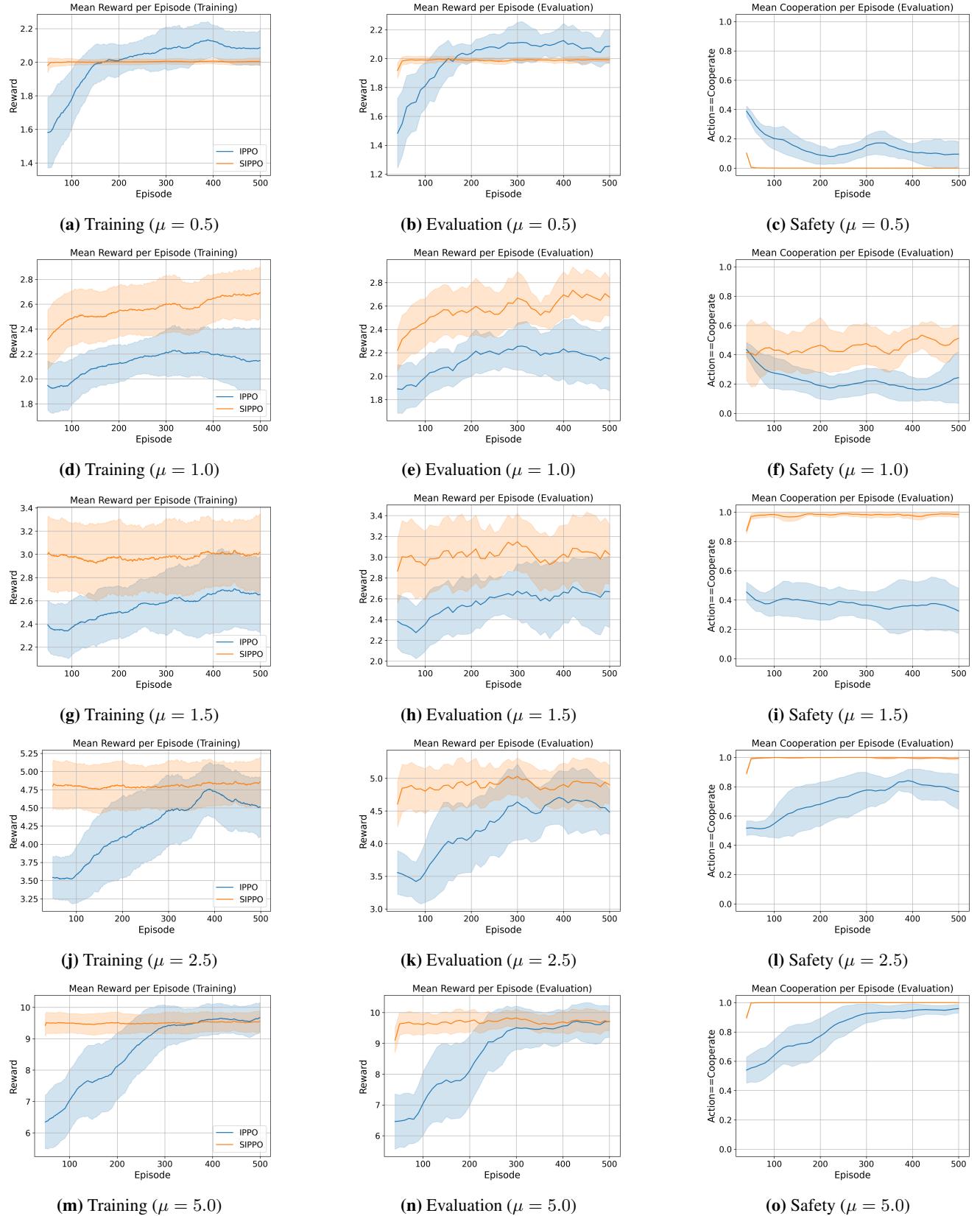
The training curves for 2-player *Extended Public Goods Game* (Figure 13) and 2-player *Markov Stag-Hunt* (Figures 14 and 15) are given below. All remaining experiment training curves will be provided at the request of the reader.

Algorithm	Parameter	CartSafe	Stag-Hunt	Centipede	EPGG	Markov Stag-Hunt
DQN	Epochs	1	x	1	x	x
	$\gamma$	0.9	x	0.99	x	x
	Buffer size	10000	x	512	x	x
	Batch size	128	x	128	x	x
	lr	0.001	x	0.001	x	x
	$\epsilon$ -decay	0.99996	x	0.9972	x	x
	$\epsilon_{min}$	0.01	x	0.01	x	x
	$\alpha$	{1.0, 5.0}	x	1.0	x	x
	$\tau$	1.0	x	1.0	x	x
PPO	Epochs	10	10	10	10	10
	$\gamma$	0.9	0.99	0.99	0.99	0.99
	Buffer size	400	50	100	50	100
	$\epsilon$ -clip	0.1	0.1	0.15	0.1	0.1
	lr (actor)	0.001	0.001	0.001	0.001	0.001
	lr (critic)	0.001	0.001	0.001	0.001	0.001
	VF coef 0.5	0.5	0.5	0.5	0.5	0.5
	S coef 0.01	0.01	0.01	0.01	0.01	0.01
	$\alpha$	{0.5, 1.0, 2.0}	1.0	1.0	1.0	1.0

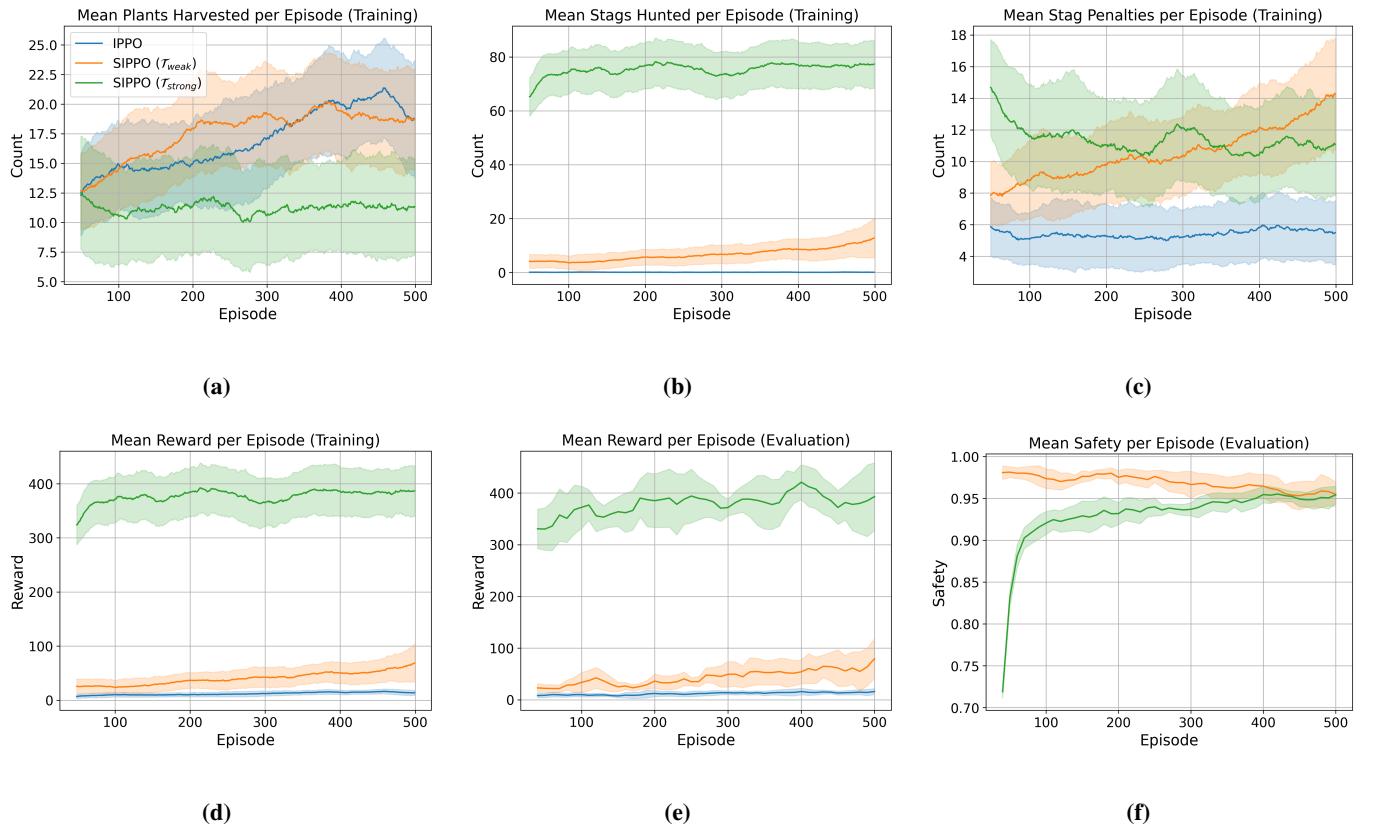
**Table 4.** Hyperparameters for experiments.



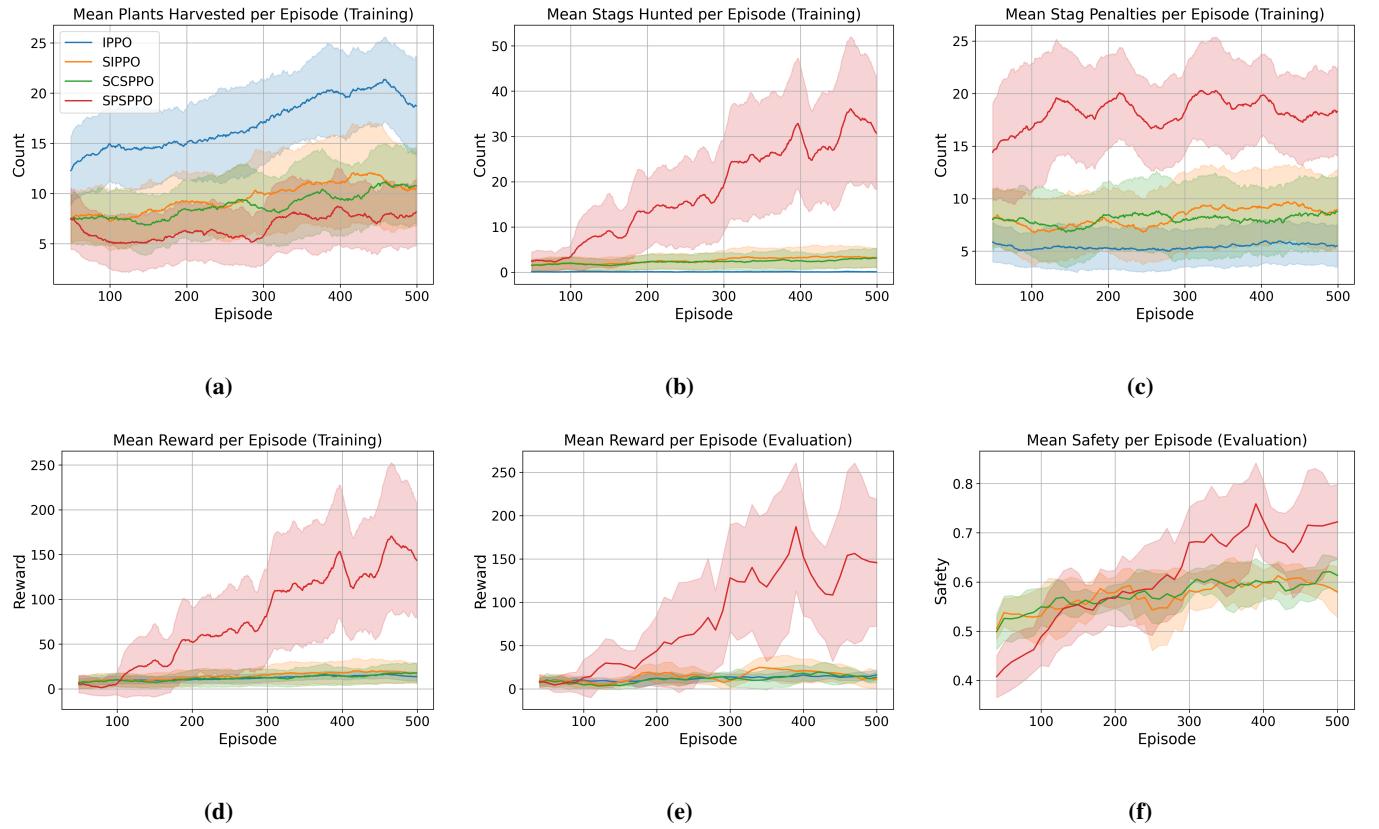
**Figure 12.** Training, evaluation and safety results for *CartSafe* for on- and off-policy DQN-based agents with different exploration strategies: *top row*: off-policy,  $\epsilon$ -greedy; *top middle row*: on-policy,  $\epsilon$ -greedy; *bottom middle row*: off-policy, softmax; and *bottom row*: on-policy, softmax. The lines represent the mean and the shadow the standard deviation over 3 seeds.



**Figure 13.** Training, evaluation, and safety results for the two-player EPGG for PLPG-based agents. The multiplicative factor  $f_t$  is sampled from  $\mathcal{N}(\mu, 1)$ , where  $\mu \in \{0.5, 1.0, 1.5, 2.5, 5.0\}$  for different experiments. The lines represent the mean and the shadow the standard deviation over 5 seeds. Results are smoothed using a rolling average with window size 50.



**Figure 14.** Behavioural and training results during training for *Markov Stag-Hunt* for PLPG-based agents with no shield or two different shields,  $\mathcal{T}_{weak}$  or  $\mathcal{T}_{strong}$ . The lines represent the mean and the shadow the standard deviation over 3 seeds. Results are smoothed using a rolling average with window size 50.



**Figure 15.** Behavioural and training results for *Markov Stag-Hunt* for PLPG-based agents with or without parameter sharing during training, where one of the agents is shielded with  $\mathcal{T}_{strong}$ . The solid lines represent the mean and the shadow the standard deviation over 3 seeds. Results are smoothed using a rolling average with window size 50.