Learned Shields for Multi-Agent Reinforcement Learning

Daniel Melcer
Northeastern University
Boston, MA, USA
melcer.d@northeastern.edu

Christopher Amato
Northeastern University
Boston, MA, USA
c.amato@northeastern.edu

Stavros Tripakis
Northeastern University
Boston, MA, USA
s.tripakis@northeastern.edu

ABSTRACT

Shielding is an effective method for ensuring safety in multi-agent domains; however, its applicability has previously been limited to environments for which an approximate discrete model and safety specification are known in advance. We present a method for learning shields in cooperative fully-observable multi-agent environments where neither a model nor safety specification are provided, using architectural constraints to realize several important properties of a shield. We show through a series of experiments that our learned shielding method is effective at significantly reducing safety violations, while largely maintaining the ability of an underlying reinforcement learning agent to optimize for reward.

CCS CONCEPTS

• **Theory of computation → Multi-agent reinforcement learning**.

KEYWORDS

Multi-Agent, Shielding, Safety

1 INTRODUCTION

Reinforcement learning (RL) has gained prominence as a method for optimizing an agent's behavior to achieve a high reward in a variety of tasks [15, 21, 25]. Various extensions of RL to the multi-agent setting [20, 26] have enabled RL to succeed in domains such as decentralized traffic light control [4], cooperative control of a robot [16], and expert-level play of multiplayer video games [23].

However, a poorly understood reward function may lead to unexpected, undesired, or unsafe behavior [5]. A large body of research has focused on safe reinforcement learning methods, to ensure that a given safety specification is enforced, regardless of the reward function [27]. One approach, shielding [2, 3], focuses not on learning a single safe policy, but on determining the set of all safe actions that agents may take. We find this approach appealing because it allows for the use of any underlying method. A RL method that supports enabled and disabled actions at each state may be protected with *pre-posed* shielding, where the RL agent receives a set of safe actions and must choose from this set. Even if the method requires a fixed set of enabled actions, it may be protected with *post-posed* shielding, where the agent is initially oblivious to the set of safe actions, but upon selecting an unsafe action, the agent's selection is blocked and it receives negative feedback.

However, existing shielding methods are limited to domains in which a model of the environment, or at least a sufficiently detailed approximation, is provided in advance. This is a much stronger assumption than is typically required for reinforcement learning, where the agent learns strictly through environment interaction.

While some domains require avoiding actions that lead to deadlock states where no safe actions are available, many domains, such as those without momentum, generally do not contain deadlock states. As we will also discuss later, we believe that shield learning in deadlock-free domains is a useful subproblem that may lead to a solution for shield learning in domains that contain deadlocks. In the single-agent deadlock-free domain, shield learning may be a straightforward supervised learning problem, after collecting a replay buffer containing safe and unsafe transitions. However, the presence of multiple agents significantly complicates the issue, as each agent must be able to independently select an individual action, with a safe joint action as the result. Furthermore, without a careful design of the learning process, algorithms that operate over the agents' joint action space often require computation exponential in the number of agents [20].

We therefore contribute a method for efficiently learning the set of safe actions available to each agent in a multi-agent setting with safety constraints, without any human-provided information about the environment or its safety specification. We discuss how the presence of a learned shield impacts the underlying training process, and introduce an optimization to avoid inducing a potential instability. We show through a series of experiments that our method is able to effectively learn a safety specification in environments where deadlock states are not a concern, and allow the underlying reinforcement learning agent to safely achieve a high reward in such tasks. Finally, we discuss considerations for extending learned shielding to environments with potential deadlock states.

1.1 Related Work

Shielding in single-agent domains [2, 3] descends from the field of reactive synthesis [6] as a method for ensuring safety while allowing for any underlying learning method to succeed, assuming that a model of the environment is available. While the problem of reactive synthesis is generally undecidable in multi-agent domains [22], it is possible to still use reactive synthesis tools to implement shielding in such domains under the assumption of local communication [7] or full observability [13]. In partially observable domains, decentralized shields can be synthesized on a best-effort basis by encoding the safety constraints as a boolean formula and using a SAT solver [14]. However, while these methods are able to guarantee zero safety violations, they require a human provided abstraction of the environment and safety specification.

There exist other methods for enforcing safety in multi-agent environments; for example, methods such as PPO-Lagrangian [18] and CPO [1] have been extended into the multiagent setting [10], and safe policies may be learned through a sequential agent iteration scheme [12]. While these methods may learn a safe policy,

Proc. of the Adaptive and Learning Agents Workshop (ALA 2025), Avalos, Aydeniz, Müller, Mohammedalamen (eds.), May 19 – 20, 2025, Detroit, Michigan, USA, ala-workshop.github.io. 2025.

we are interested in learning a shield that returns a variety of safe actions, rather than a single safe policy.

Regardless of safety, multi-agent reinforcement learning is generally accomplished via a centralized training and decentralized execution (CTDE) paradigm. For example, MAPPO [26] is a natural extension of PPO [19] that maintains decentralized actors, but uses a centralized critic for more accurate advantage calculations. Several extensions of Q-learning to multi-agent settings use individual utility values for each agent, and introduce a constrained mixing function to combine these into a centralized Q-value such that the best joint action corresponds to the collection of each agents' best individual actions [17, 20, 24].

2 PRELIMINARIES

2.1 Notation

For set X, let $\Delta(X)$ be the set of distributions over X. For distribution $x \in \Delta(X)$, let $supp(x) \subseteq X$ be its support; i.e. the set of all values X with nonzero probability in x. 2^X is the powerset of X.

Given joint action a, we index individual actions as $a_1, \ldots, a_k$.

2.2 SMMDPs

Cooperative fully observable multiagent environments are often characterized as a Multiagent Markov Decision Process (MMDP). We extend its description to include a binary safety specification:[1]

Definition 1 (SMMDP). *A Safety Multiagent Markov Decision Process (SMMDP) is a tuple* $\mathcal{M} = (I = [1..k], S, S_0, A = (A_1 \times \ldots \times A_k), T, R, \gamma, U)$ *where* I *is a set of agents,* S *is a state space,* $S_0 \in \Delta(S)$ *is a distribution over initial states,* A *represents the joint action space, composed of the product of* k *individual action spaces,* $T : S \times A \to \Delta(S)$ *represents the state transition function,* $R : S \times A \times S \to \mathbb{R}$ *is the reward function,* γ *is the reward discount factor, and* $U : S \times A \to \mathbb{B}$ *denotes if a state-action pair causes a safety violation.*

The aim of multi-agent reinforcement learning is to find a set of individual policies—functions $\pi_i : S \to \Delta(A_i)$—that maximizes the expected sum of discounted rewards; i.e. the expected return:

Definition 2 (Expected Return). *Given SMMDP* $\mathcal{M}$ *and a set of policies* $\pi_i : S \to \Delta(A_i)$ *for* $i \in I$*, the expected return of a taking action* a *in state* s *is:*

$$Q^{\pi}(s,a) = \mathbb{E}_{s_0=s;a_0=a,a_{n,i}\sim\pi_i(s_n);s_{n+1}\sim T(s_n,a_n)}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1})\right]$$

Where $a_n = (a_{n,1}, \ldots, a_{n,k})$. Finally, a set of individual policies is safe if they collectively only choose safe actions:

Definition 3 (Safe Policy). *A set of policies* $\pi_1, \ldots, \pi_k$ *is* safe *in SMMDP* $\mathcal{M}$ *if* $\forall s \in S, a \in (supp(\pi_1(s)) \times \ldots \times supp(\pi_k(s))), U(s,a) = \bot$*; i.e. the policies never choose a violating action*

2.3 Mixing Functions

Given a function $Q : S \times A \to \mathbb{R}$, a common task in reinforcement learning involves computing $\max_{a \in A} Q(s,a)$ for some $s \in S$ [21]. However, if A is large—for example, if it is the product of sets $A_1 \times \ldots \times A_k$—then the maximization may be expensive to compute.

[1] We contrast this with Constrained MMDPs, where the sum of real-valued costs must remain below a threshold.

One common approach in MARL is to introduce individual functions $Q_i : S \times A_i \to \mathbb{R}$ for $i \in I$, and to constrain Q such that it satisfies the *Individual-Global Max* principle with respect to the collection of each agents' Q_i:

Definition 4 (Individual-Global Max). *Function* Q *satisfies the* Individual-Global Max (IGM) *principle with respect to* $Q_1, \ldots, Q_k$ *if* $\forall s \in S, \{argmax_{a_i \in A_i} Q_i(s,a_i)\}_{i \in I} = argmax_{a \in A} Q(s,a)$.

Most implementations realize the IGM constraint by *mixing* the outputs of each individual Q_i—a mixing architecture as simple as $Q(s,a) = \sum_{i \in I} Q_i(s,a_i)$ is sufficient to satisfy this constraint [20], but more complex mixing architectures allow for a more general realization of the function class [17, 24].

Using any architecture that satisfies IGM, the maximization may be completed in $O(\sum_{i \in I} |A_i|)$ operations, rather than $O(|A|) = O(\prod_{i \in I} |A_i|)$ operations—each agent may iterate over its individual action space to obtain the maximum, rather than needing to iterate over the joint action space.

2.4 Shielding

Shielding [2, 3] is a class of methods for safe reinforcement learning. Agents are equipped with a shield:

Definition 5 (Shield). *A shield is a function* $\mathcal{H} : S \to 2^A$ *such that* $\forall s \in S, \mathcal{H}(s) \neq \emptyset$.

We focus on *pre-posed* shielding, in which the shield provides a set of actions at each state, such that some desired safety specification is maintained if the agent selects an action from this set. When at state s, shield $\mathcal{H}$ is applied to policy π by setting the probability $\pi(a|s)$ for all actions $a \notin \mathcal{H}(s)$ to 0, and renormalizing the remaining probabilities.

In the multiagent setting [7, 13], a decentralized shield is a set of individual shields $\mathcal{H}_i : S \to 2^{A_i}$ such that each agent can select any individual action from its shield, and the resulting joint action maintains the safety specification; in other words, that each individual shield applied to its respective policy results in a safe set of policies for the environment.

3 PROBLEM STATEMENT & METHOD OVERVIEW

Our safe multiagent reinforcement learning problem may be summarized as follows:

Problem 1. *Given a SMMDP* $\mathcal{M}$*, find a safe policy for each agent* $\pi_i : S \to \Delta(A_i)$ *that maximizes the expected return.*

Our approach uses the basic structure of shielded multiagent reinforcement learning:

(1) Construct a decentralized shield that constrains the agents' action spaces such that they can only choose safe actions.
(2) Given a decentralized shield, learn a decentralized policy that maximizes the expected return under this shield.

When the environment specification and safety constraint are known in advance, a shield may be constructed ahead of time using a reactive synthesis tool [2]. However, because these inputs are not available in our case, we must instead learn the shield by interacting with the environment. As the shield is learned at the same time as

the policy, several new optimizations must be made to the policy learning step as well.

We focus on the case where $\forall s \in S, \exists a \in A, U(s,a) = \bot$—there exists some non-violating action at every state—and discuss environments in which this does not hold in Section 6.

3.1 Shield Construction

To act safely, agents must be restricted such that in state s, they may only select joint actions a where $U(s,a) = \bot$. It may not be efficient to simply learn and use a function that approximates U directly—during action selection, agents will be required to iterate over joint actions until they find a safe action. The number of possible joint actions grows exponentially with the number of agents, leading to scalability challenges. Additionally, if communication is restricted after training is complete, each agent must be able to select its set of safe individual actions independently from other agents.

Therefore, we learn individual safety functions $\mathcal{F}_i : S \times A_i \to [0,1]$ for $i \in I$. As these are learned functions and must have a continuous output to enable gradient descent, we use threshold $t \in (0,1)$ to ultimately determine if an action is allowed by the shield or not. We obtain individual shields by collecting all actions above the threshold: $\mathcal{H}_i(s) = \{a | \mathcal{F}_i(s,a) > t\}$. We find that $t = 0.5$ works well empirically.

The safety functions $\mathcal{F}_i$ must satisfy two constraints. First, as long as each agent selects any action allowed by its individual shield—any action where the individual safety function's output is above the threshold—a violating joint action is not selected.

$$\forall s \in S, \forall a \in A, \bigwedge_{i \in I} \mathcal{F}_i(s, a_i) > t \implies U(s,a) = \bot \tag{1}$$

Second, as each individual shield must have a nonempty set of actions at each state, there must be some action for which each individual safety function returns a value above t:

$$\forall s \in S, \forall i \in I, \exists a_i \in A_i, \mathcal{F}_i(s, a_i) > t \tag{2}$$

Problem 2. *Given SMMDP $\mathcal{M}$, find a set of individual shields $\mathcal{F}_i : S \times A_i \to [0,1]$ for $i \in I$ such that constraints 1 and 2 hold.*

3.1.1 Constraint (1)—Only Safe Actions Selected. Recall that, as described in Section 2.3, a set of functions that satisfy IGM may be used to avoid iterating over a joint action space, and enable agents to independently select from their individual action spaces.

Inspired by the IGM constraint, we define the *Individual-Global Safe* (IGS) principle:

Definition 6 (IGS Principle). *Joint safety function $\mathcal{F} : S \times A \to [0,1]$ satisfies the Individual-Global Safe (IGS) principle with respect to individual safety functions $\mathcal{F}_i : S \times A_i \to [0,1]$ for $i \in I$ if $\forall s \in S, a = (a_1, \ldots, a_k) \in A, \exists i \in I, \mathcal{F}_i(s, a_i) \leq \mathcal{F}(s,a)$.*

Let the safety indicator function $\tilde{\mathcal{F}}(s,a) = \begin{cases} 1 & U(s,a) = \bot \\ 0 & U(s,a) = \top \end{cases}$

If we construct individual functions $\mathcal{F}_1, \ldots, \mathcal{F}_k$ such that $\tilde{\mathcal{F}}$ satisfies IGS with respect to them, then each agent's local decision making will ensure safety:

Theorem 1. *If $\tilde{\mathcal{F}}$ satisfies IGS with respect to $\mathcal{F}_1, \ldots, \mathcal{F}_k$, then $\mathcal{F}_1, \ldots, \mathcal{F}_k$ satisfy constraint (1).*

Proof. By cases on $U(s,a)$; if $U(s,a) = \top$, $\tilde{\mathcal{F}}(s,a) = 0$, and thus $\exists i \in I, \mathcal{F}_i(s,a_i) \leq 0$. As $t \in (0,1)$, $\mathcal{F}_i(s,a_i) < t$, and constraint (1) is satisfied. If $U(s,a) = \bot$, constraint (1) is trivially satisfied.

While we cannot *directly* construct the individual safety functions so that IGS holds with respect to $\tilde{\mathcal{F}}$, we can create a function $\mathcal{F}$ that is structurally constrained to satisfy IGS with respect to $\mathcal{F}_1, \ldots, \mathcal{F}_k$, and then train $\mathcal{F}$ end-to-end to approximate $\tilde{\mathcal{F}}$. If this approximation is succesfully learned, then the learned $\mathcal{F}_1, \ldots, \mathcal{F}_k$ are guaranteed to satisfy constraint (1).

We realize the IGS principle with the following mixing architecture:

$$\mathcal{F}(s,a) = \max\left(D(s,a), \min_{i \in I} \mathcal{F}_i(s, a_i)\right)$$

Where $D : S \times A \to \mathbb{R}$ is an unconstrained learned function.

Theorem 2. *The proposed mixing architecture satisfies IGS.*

Proof. Due to the outer maximization, $\min_{i \in I} \mathcal{F}_i(s,a_i) \leq \mathcal{F}(s,a)$. The minimization then ensures that $\exists i \in I, \mathcal{F}_i(s,a_i) = \mathcal{F}(s,a)$, and therefore $\exists i \in I, \mathcal{F}_i(s,a_i) \leq \mathcal{F}(s,a)$.

3.1.2 Constraint (2)—Some Action Always Available. We have previously treated $\mathcal{F}_i$ as a black box; however, to enforce constraint (2), we impose an internal structure on this function. We structure $\mathcal{F}_i$ as follows:[2]

$$\mathcal{F}_i(s,a_i) = \frac{\textsc{Posact}(\mathcal{F}_i^*(s,a_i))}{\max_{a_i' \in A_i} \textsc{Posact}(\mathcal{F}_i^*(s,a_i'))}$$

where $\mathcal{F}_i^* : S \times A_i \to \mathbb{R}$ is an unconstrained learned function for $i \in I$, and Posact refers to any activation function whose range is the positive reals. We observe that the softplus activation function works well for this. The maximization only occurs over the set of individual actions, and is therefore more efficient to compute compared to an operation that acts over the entire joint action space. This can further be improved by designing $\mathcal{F}_i^*$ such that it outputs values for all individual actions in one pass; for example, if $\mathcal{F}_i^*$ is implemented as a neural network, by using a last layer of size $|A_i|$.

Regardless of the output of $\mathcal{F}_i^*$ itself, there will always be some action at each state for which $\mathcal{F}_i(s,a) = 1$, and is thus above the threshold. Importantly, though not strictly necessary to satisfy the constraint, the structure that we present has the capacity to represent cases where several or all individual actions are enabled.

3.2 Learning With a Shield

Problem 3. *Given SMMDP $\mathcal{M}$ and individual shields $\mathcal{H}_1, \ldots, \mathcal{H}_k$, find individual policies $\pi_1, \ldots, \pi_k$ that are optimal in $\mathcal{M}$, such that $\forall s \in S, i \in I, supp(\pi_i(s)) \subseteq \mathcal{H}_i(s)$.*

We use the learned $\mathcal{H}_1, \ldots, \mathcal{H}_k$ to implement pre-posed shielding [2] as described in Section 2.4; the shield provides action masks that may be used with any reinforcement learning method that supports enabled and disabled actions. We choose MAPPO [26] as our base RL method; after evaluating the underlying policy for agent i on a given state s, the probabilities of any actions not in $\mathcal{H}_i(s)$ are zeroed out, and the remainder is re-normalized.[3]

[2] We additionally stop gradient propagation in the denominator for improved stability.
[3] This is implemented as setting the logits of unsafe actions to $-\infty$ prior to the log-sumexp normalization.

Alshiekh et al. [2] show that shielded reinforcement learning exhibits the same convergence guarantees as the underlying RL method; however, if our shield and agent are trained together in a bootstrapping scenario, the shield changes as the agent is trained. This effectively creates a nonstationary environment, loosening the convergence guarantees. We discuss several mitigations for this nonstationarity in Section 4.2.

4 METHOD DETAIL

While we use the basic individual components discussed in Section 3, integration of these into a complete reinforcement learning system requires some additional work.

4.1 Shield learning

The mixed shield function is trained end-to-end using a standard MSE loss:

$$\mathcal{L}^{\mathcal{F}}(s,a) = \left(\mathcal{F}(s,a) - \tilde{\mathcal{F}}(s,a)\right)^2$$

Where $\tilde{\mathcal{F}}$ is the safety indicator function, defined in Section 3.1.

Due to the softplus activation within $\mathcal{F}_i$, the outputs of $\mathcal{F}_i^*$ may become extremely negative while training in domains where violations are relatively uncommon at first; for example, if agents must perform some amount of exploration prior to encountering an unsafe state. This may lead to floating point numerical issues. Therefore, we instead use a training target of $\tilde{\mathcal{F}}(s,a) = \epsilon_{\mathcal{F}} = 0.01$ when $U(s,a) = \top$; our method only requires that $t > \epsilon_{\mathcal{F}}$.

We additionally note two auxiliary loss functions that empirically aid the training process. First, due to the network architecture, gradients do not propagate to all individual shields for every training example, leading to "floating" values of $\mathcal{F}_i$. We counteract this with the following *determinism loss*.

$$\mathcal{L}^{\mathcal{F}\text{-det}}(s,a) = \sum_{i \in I} \left(\mathcal{F}_i(s,a_i) - \textsc{Round}_{t,\epsilon_{\mathcal{F}}}(\mathcal{F}_i(s,a_i))\right)^2$$

Where $\textsc{Round}_{t,\epsilon_{\mathcal{F}}}$ rounds values greater than t to 1, and values less than t to $\epsilon_{\mathcal{F}}$.

Finally, we would like to discourage the network from learning $\forall s \in S, a \in A, D(s,a) = \tilde{\mathcal{F}}(s,a)$, as otherwise there is little incentive for $\mathcal{F}_i(s,a_i) = 1$ for more than one individual action per state. We accomplish this by adding a *nonredundancy loss*:

$$\mathcal{L}^{\mathcal{F}\text{-nr}}(s,a) = \left(\textsc{Clamp}_{[1,\infty)}(\textsc{StopGrad}(\min_{i \in I} \mathcal{F}_i(s,a_i)) + D(s,a)) - 1\right)^2$$

Note that StopGrad acts as the identity, with the anomalous behavior that $\frac{\delta \textsc{StopGrad}(x)}{\delta x} = 0$.

Intuitively, if both $\min_{i \in I} \mathcal{F}_i(s,a_i) = 1$ and $D(s,a) = 1$, the $D(s,a) = 1$ is redundant. The above loss function pushes $D(s,a)$ to be decreased, without affecting $\mathcal{F}_i(s,a_i)$.

Our final loss function for training $\mathcal{F}$ is as follows:

$$\mathcal{L}^{\mathcal{F}-\text{complete}} = \mathbb{E}_{(s,a)\sim B}\left[\mathcal{L}^{\mathcal{F}}(s,a) + \alpha^{\text{det}}\mathcal{L}^{\mathcal{F}\text{-det}}(s,a) + \alpha^{\text{nr}}\mathcal{L}^{\mathcal{F}\text{-nr}}(s,a)\right]$$

Where B is a replay buffer containing encountered states and actions, as well as the necessary information to compute $\tilde{\mathcal{F}}(s,a)$. We use $\alpha^{\text{det}} = \alpha^{\text{nr}} = 0.01$. We perform gradient-based optimization on this loss function, as described in the Hyperparameters section.[4]

4.2 MAPPO

Action masking while learning a shield complicates the underlying reinforcement learning process, as the environment appears nonstationary from the agent's perspective due to the continually evolving shields, in addition to any nonstationarity from the perspective of each agent due to the other agents' evolving policies.

In particular, when one action is favored during MAPPO training, its probability continually increases in the policy; after many training steps, we have observed the favored action being on the order of 10^{200} times more likely than the next action. If the shield later learns that this action is unsafe, the effect of selecting an action that was previously so unlikely causes catastrophic gradient magnitudes and floating point errors.

We protect against this by preventing the action probabilities from becoming so imbalanced in the first place. The usual method of adding an entropy loss does not fully protect against this; two actions may be relatively likely, leading to a substantial amount of entropy despite the presence of other actions with infinitesimal probability. When both relatively probable actions are marked as unsafe, the catastrophic update problem will still manifest.

Therefore, in addition to the entropy loss, we add a new clipping operation to the training objective. Let $\pi_i(a_i|s) = \textsc{Softmax}_{a_i \in A_i} \pi_i^*(a_i|s)$ be the policy probabilities, obtained using the softmax activation function on unconstrained neural network π_i^*.

Let $\mathcal{L}^{\text{PPO}}(s,a)$ be the standard PPO loss function [19]. We truncate the loss to limit the imbalance in the policy logits to Δ.[5]

$$\mathcal{L}^{PPO-\Delta}(s,a_i) = \begin{cases} \textsc{StopGrad}[\mathcal{L}^{\text{PPO}}(s,a)] & A(a|s) > 0 \wedge \pi_i^*(a_i|s) > \min_{a_i'} \pi_i^*(a_i'|s) + \Delta \\ \textsc{StopGrad}[\mathcal{L}^{\text{PPO}}(s,a)] & A(a|s) < 0 \wedge \pi_i^*(a_i|s) < \max_{a_i'} \pi_i^*(a_i'|s) - \Delta \\ \mathcal{L}^{\text{PPO}}(s,a) & \text{otherwise} \end{cases}$$

We use $\Delta = 20$ in our experiments.[6]

Finally, we add a loss to train the logits of unsafe actions to equal that of the worst safe action:

$$\mathcal{L}^{\text{US}}(s) = \sum_{i \in I} \sum_{a_i \in A_i \setminus \mathcal{H}_i(s)} \left(\pi_i^*(a_i|s) - \textsc{StopGrad}\left[\min_{a_i' \in \mathcal{H}_i(s)} \pi_i^*(a_i'|s)\right]\right)^2$$

5 EXPERIMENTS

We implemented our method for two domains, as shown in Figure 1—a cooperative gridworld navigation task, and a more complex soup preparation task based on the game "Overcooked". In

[4] Gradients are not propagated through the non-differentiable function Round. For min and max, gradients are passed through to the extreme element.

[5] Similarly to $\mathcal{L}^{PPO}$, this may be written with min and Clamp; we use an explicit piecewise version here for clarity.

[6] We speculate that such a modification is likely unnecessary if a method based on Q-learning is used, as all actions' Q-values are trained to a specific target, rather than continually pushed in one direction as in policy-gradient methods. Existing shielding work [7, 14] uses Q-learning with little modification.

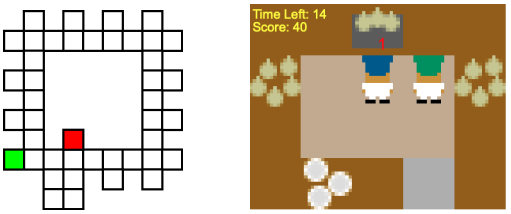


Figure 1: Evaluation environments. (Left) Gridworld navigation domain; agents start in random positions and must reach their respective goals. (Right) Overcooked domain; agents must cooperate to cook an onion soup dish in a cramped kitchen.

both cases, the agent is not given any model of the environment, and must learn the safety specification of collision avoidance from scratch, only receiving feedback when it violates the specification.

In the navigation task, agents receive a modest negative reward when bumping into a wall, a large negative reward when colliding into each other, a large positive reward when both agents reach the goal, and a small negative reward if none of these actions occur. The episode ends upon colliding with the other agent or reaching the goal. In the Overcooked task, agents receive rewards when they perform actions that contribute to order fulfillment, such as putting onions on the stove or plating a complete dish. There is no reward as a result of a collision. Both tasks are undiscounted and use a horizon of 500 time steps. The Overcooked task uses two layouts—the "Cramped Room" layout in which agents are in a 2x3 cell area, and the "Coordination Ring" layout in which agents share a 3x3 area with a counter in the middle.

We compare our method to unshielded MAPPO. Theoretically, agents should learn to avoid unsafe actions in both domains even without shielding—the navigation task imposes a large negative reward for unsafe actions; in Overcooked, colliding with the other chef accomplishes nothing and thus decreases the total reward that may be achieved in the limited time. We also compare to the baseline of reward augmentation, where a large negative reward is added when an unsafe action occurs.

All domains were run with 10 trials; extended hyperparameters are described in the Appendix. The results are shown in Figures 2 and 3. As the results demonstrate, in all domains, our learned shielding method converges to zero unsafe actions per episode.

In the gridworld domain, our method outperforms the other methods for learning the task-specific reward function; because of the small negative reward at each timestep, agents initially learn the local optimum where they collide with each other to immediately end the episode. Our method quickly learns that this is unsafe, and "kicks out" the agents from this local optimum, while the other methods must find the global optimum by chance before they stop taking unsafe actions.

In Overcooked, there is much less of a reward incentive to act safely, so the unshielded agents continually collide late into the training process. Agents with an augmented reward signal fail to learn a useful policy, illustrating the challenge of attacking safety using only reward shaping—too strong a reward signal, and agents are "afraid" to move at all. Finally, the learned shielding method is

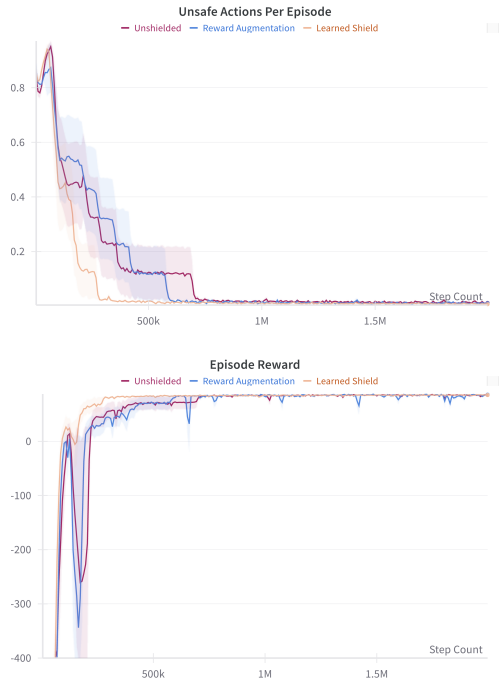
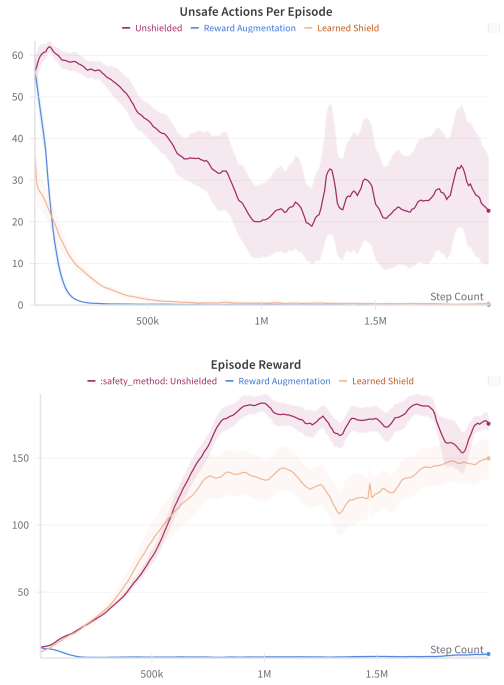


Figure 2: Average number of unsafe actions and reward per episode, and standard error over 10 trials for the gridworld-collision domain, "ISR" layout. Episode reward represents the original reward from the environment, prior to augmentation. Values are smoothed using a moving window average of 10K steps.

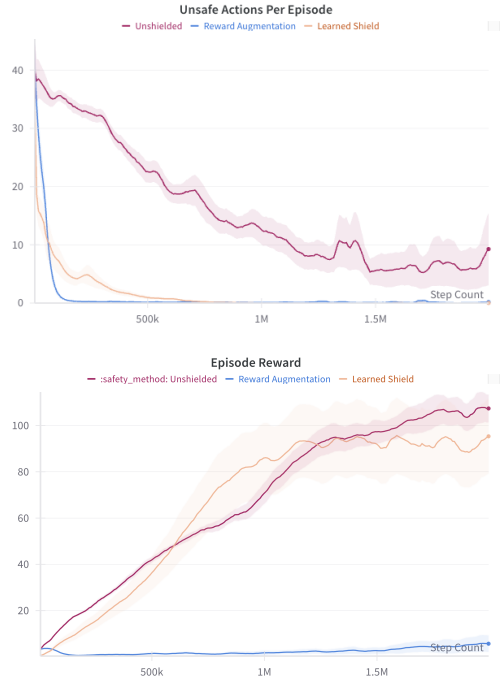
able to achieve substantial reward in this domain, nearly matching the reward obtained by the unshielded agent—it may act slightly too conservatively to achieve the full reward potential. Nevertheless, our method achieves a balance of maintaining safety without acting so conservatively that it fails to achieve reward.

6 DISCUSSION & FUTURE WORK

As discussed in Section 3, our method is designed for environments where for every state, there exists some safe action. This assumption does not necessarily apply to all environments; for example, in some domains with momentum, no action can prevent the agent from violating a safety specification in some states. In particular, many larger or continuous state-space environments require consideration of this. We believe that our method may still be able to work in such domains, by changing $\tilde{\mathcal{F}}$, the learning target of $\mathcal{F}$, such that actions that transition to deadlock states—states with no safe actions available—are themselves unsafe. This could be accomplished by separately learning a safety value function to predict such states; in the multiagent setting, this could be implemented with a sequential update procedure [12], or through an adaptation of a QPLEX-like structure [24]. Given this modified $\tilde{\mathcal{F}}$, it may be possible to re-use the remainder of our shield learning method to solve the shield learning problem in more general domains. We leave further study of this to future work.



(a) "Cramped Room" layout



(b) "Coordination Ring" layout

Figure 3: Results for Overcooked over 10 trials.

Similarly, our implementation requires full observability in cooperative environments. We are not confident that a naive extension to partially observable domains through the use of recurrent networks would be theoretically justified, as this would imply a more complex relationship between individual and joint safety values. Despite the undecidability of decentralized reactive synthesis in general [22], prior work has synthesized shields in a useful subset of partially observable environments, with an available model [14]. Similarly, we believe that there may be a useful subset of partially observable environments where it is possible to learn a shield without a model. Further work is required to create a well-grounded method for handling partial observability or mixed cooperative-competitive environments.

7 CONCLUSION

We have presented, to our knowledge, the first method that extends shielding to multi-agent domains where no model of the environment is provided. We introduce a method of constraining a set of function approximators to follow the IGS principle, and demonstrate several optimizations that allow policy gradient methods to handle the nonstationary effects of a learned shield. Our shield learning method performs well on several domains, learning to safely obtain high task-specific performance.

HYPERPARAMETERS

We further describe the hyperparameters for our training process.

For both environments, all function approximators are instantiated as neural networks with three hidden layers of size 1024, 1024, and 256, with relu activations [9] and Xavier normal initializations [8]. We maintain a replay buffer with all observed transitions, up to the 2×10^6 step limit. Every 1000 steps, we perform 10 shield training steps with a minibatch size of 160 (16 sequences of length 10 each). Every 16 episodes, we perform 32 MAPPO training steps with a minibatch size of 320 (32 sequences of length 10 each). We use a PPO clipping parameter of 0.1, GAE $\lambda = 0.95$, entropy loss coefficient of 0.1, and $\Delta = 20$. All other losses, besides $\mathcal{L}^{\mathcal{F}}$ and $\mathcal{L}^{PPO-\Delta}$ have a coefficient of 0.01. We use the Adam optimizer [11] with a learning rate of 10^{-4} for both the learned shielding module and the policy module.

No formal search procedure was utilized to obtain these hyperparameters; they were selected for the gridworld domain based on observed stable training and adequate performance for all agents, and re-used for Overcooked without further tuning. The sole modification for Overcooked is that all neural networks for the gridworld domain use a one-hot transformation of several state variables as input, while the state provided by Overcooked is passed to all networks without transformation. However, other domains that are more complex, or distinct from each other, may require environment-specific tuning.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, Sydney, NSW, Australia, 22–31. https://proceedings.mlr.press/v70/achiam17a.html

[2] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe reinforcement learning via shielding. In

Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32. AAAI Conference on Artificial Intelligence, New Orleans, LA, 10.

[3] Roderick Bloem, Bettina Könighofer, Robert Könighofer, and Chao Wang. 2015. Shield synthesis: Runtime enforcement for reactive systems. In *Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 21*. Springer, International Conference on Tools and Algorithms for the Construction and Analysis of Systems, London, UK, 533–548.

[4] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. 2020. Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems* 21, 3 (2020), 1086–1095. https://doi.org/10.1109/TITS.2019.2901791

[5] Jack Clark and Dario Amodei. 2016. Faulty reward functions in the wild. https://openai.com/research/faulty-reward-functions

[6] Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (Eds.). 2018. *Handbook of Model Checking*. Springer International Publishing, New York, NY. https://doi.org/10.1007/978-3-319-10575-8

[7] Ingy ElSayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. 2021. Safe Multi-Agent Reinforcement Learning via Shielding. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems* (Virtual Event, United Kingdom) *(AAMAS '21)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 483–491.

[8] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 9)*, Yee Whye Teh and Mike Titterington (Eds.). PMLR, Chia Laguna Resort, Sardinia, Italy, 249–256. https://proceedings.mlr.press/v9/glorot10a.html

[9] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 15)*, Geoffrey Gordon, David Dunson, and Miroslav Dudík (Eds.). PMLR, Fort Lauderdale, FL, USA, 315–323. https://proceedings.mlr.press/v15/glorot11a.html

[10] Shangding Gu, Jakub Grudzien Kuba, Yuanpei Chen, Yali Du, Long Yang, Alois Knoll, and Yaodong Yang. 2023. Safe multi-agent reinforcement learning for multi-robot control. *Artificial Intelligence* 319 (2023), 103905. https://doi.org/10.1016/j.artint.2023.103905

[11] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

[12] Zeyang Li and Navid Azizan. 2024. Safe Multi-Agent Reinforcement Learning with Convergence to Generalized Nash Equilibrium. arXiv:2411.15036 [cs.LG] https://arxiv.org/abs/2411.15036

[13] Daniel Melcer, Christopher Amato, and Stavros Tripakis. 2022. Shield Decentralization for Safe Multi-Agent Reinforcement Learning. *Advances in Neural Information Processing Systems* 36 (2022), 13.

[14] Daniel Melcer, Christopher Amato, and Stavros Tripakis. 2024. Shield Decomposition for Safe Reinforcement Learning in General Partially Observable Multi-Agent Environments. In *Reinforcement Learning Conference*. RLJ, Amherst, MA, USA, 8.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (Feb. 2015), 529–533. https://doi.org/10.1038/nature14236

[16] Bei Peng, Tabish Rashid, Christian A. Schroeder de Witt, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. FACMAC: Factored Multi-Agent Centralised Policy Gradients. arXiv:2003.06709 [cs.LG] https://arxiv.org/abs/2003.06709

[17] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *J. Mach. Learn. Res.* 21, 1, Article 178 (jan 2020), 51 pages.

[18] Alex Ray, Joshua Achiam, and Dario Amodei. 2017. Benchmarking Safe Exploration in Deep Reinforcement Learning. https://cdn.openai.com/safexp-short.pdf

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG] https://arxiv.org/abs/1707.06347

[20] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2017. Value-Decomposition Networks For Cooperative Multi-Agent Learning. arXiv:1706.05296 [cs.AI] https://arxiv.org/abs/1706.05296

[21] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA.

[22] Stavros Tripakis. 2004. Undecidable Problems of Decentralized Observation and Control on Regular Languages. *Inform. Process. Lett.* 90, 1 (April 2004), 21–28. https://doi.org/10.1016/j.ipl.2004.01.004

[23] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (Nov. 2019), 350–354. https://doi.org/10.1038/s41586-019-1724-z

[24] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2021. QPLEX: Duplex Dueling Multi-Agent Q-Learning. arXiv:2008.01062 [cs.LG] https://arxiv.org/abs/2008.01062

[25] Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmehr Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. 2022. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature* 602, 7896 (Feb. 2022), 223–228. https://doi.org/10.1038/s41586-021-04357-7

[26] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. arXiv:2103.01955 [cs.LG] https://arxiv.org/abs/2103.01955

[27] Weiye Zhao, Tairan He, Rui Chen, Tianhao Wei, and Changliu Liu. 2023. State-wise Safe Reinforcement Learning: A Survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, Edith Elkind (Ed.). International Joint Conferences on Artificial Intelligence Organization, Macao, 6814–6822. https://doi.org/10.24963/ijcai.2023/763 Survey Track.