ON SOLVING COOPERATIVE DECENTRALIZED MARL PROBLEMS WITH SPARSE REINFORCEMENTS

Anonymous authors
Paper under double-blind review

1 ABSTRACT

Decentralized decision makers learn to cooperate and make decisions in many domains including (but not limited to) search and rescue, drone delivery, box pushing and fire fighting problems. In these cooperative domains, a key challenge is one of sparse rewards, i.e., rewards/reinforcements are obtained only in a few situations (e.g., on extinguishing a fire, on moving a box) and in most other situations there is no reward/reinforcement. The problem of learning with sparse reinforcements is extremely challenging in cooperative Multi-Agent Reinforcement Learning (MARL) problems due to two reasons: (a) Compared to the single agent case, exploration is harder as multiple agents have to be coordinated to receive the reinforcements; and (b) Environment is not stationary as all the agents are learning at the same time (and therefore change policies) and therefore the limited (due to sparse rewards) good experiences can be quickly forgotten.

One approach that is scalable, decentralized and has shown great performance in general MARL problems is Neural Fictitious Self Play (NFSP). However, since NFSP averages best response policies, a good policy can be drowned in a deluge of bad best-response policies that come about due to sparse rewards. In this paper, we provide a mechanism for imitation of good experiences within NFSP that ensures good policies do not get overwhelmed by bad policies. We then provide an intuitive justification for why self imitation within NFSP can improve performance and how imitation does not impact the fictitious play aspect of NFSP. Finally, we provide a thorough comparison (experimental or descriptive) against relevant cooperative MARL algorithms to demonstrate the utility of our approach.

2 INTRODUCTION

Cooperative Multi-Agent Reinforcement Learning (MARL) is an important framework for learning agent policies in multiple domains including but not limited to disaster rescue Nanjanath et al. (2010); Parker et al. (2016), fire fighting Oliehoek et al. (2008) and box pushing Seuken & Zilberstein (2012). In these problems, a team of agents (or robots) coordinate to accomplish tasks (find people, extinguish fires, and push boxes to destinations) in uncertain domains. There are multiple key challenges in these problems of interest: (a) Uncertainty in movement or in accomplishing tasks; (b) Coordination of decentralized entities to accomplish tasks (e.g., big fires require multiple fire engines or pushing a large box may require multiple robots; (c) Affected global state: Global state (representing status of tasks) can be impacted by agent actions; and most importantly (d) Sparse rewards: rewards are obtained by an agent only when tasks are accomplished and there are only a few tasks.

Due to its relevance in many team problems, research in cooperative MARL is extensive. There are multiple threads of relevant research in cooperative MARL. First, we have team learning approaches Haynes et al.; Haynes & Sen (1995); Sen & Sekaran (1995); Boutilier (1996); Claus & Boutilier (1998) where a single learner learns policies for a team of agents. Since there is a single learner, approaches from single agent learning can be employed for optimizing performance of the team. However, team learning approaches suffer from curse of dimensionality, where state and action space increases exponentially with number of agents. Furthermore, it may not be realistic to assume centralization of information, especially if the agents are decentralized.

The second thread of research has focussed on concurrent learning Agogino & Tumer (2006); Tampuu et al. (2017), where agents learn concurrently to avoid the curse of dimensionality and centralization of information. However, this is challenging as the RL problem faced by an individual agent is no longer stationary, especially as other agents can also change their policies.

In order to address the non-stationarity issue due to concurrent learning of multiple agents, a thread of research has focussed on methods with a centralized critic. One of the leading approaches in this space is called COMA Foerster et al. (2018) that employs an actor critic model with a centralized critic (which takes state, action information from all agents) and decentralized actors that are trained independently using local information of individual agents. Another leading approach is by Nguyen et al. (2017) to solve cooperative problems with large numbers of homogenous agents and anonymous interactions. However, it relies on having non-global states and transition function decomposability given number of agents. This is not feasible in domains of interest in this paper and since it is based on actor critic architecture, it will have same issues as other MARL approaches with sparse rewards.

The last thread of relevant research has employed game theory to develop decentralized learning methods Hu & Wellman (2003); Heinrich et al. (2015); Heinrich & Silver (2016). One of the leading approaches is the neural fictitious self play method Heinrich & Silver (2016), which employs ideas from the well known fictitious play Brown (1951) method. Given the focus on equilibrium for game theoretic methods, these approaches can get stuck in bad local optima in the case of cooperative problems. The key advantage of these approaches, specifically NFSP is decentralized learning at scale.

Even though the research in cooperative MARL is extensive, existing work is unable to provide good policies (as demonstrated in experimental results) in the presence of sparse rewards. The issue of sparse rewards is also present in single agent RL Oh et al. (2018), however, the problem is exacerbated in multi-agent problems due to two reasons: (1) Exploration is harder as multiple agents have to be coordinated to receive the rewards; and (b) Environment is not stationary since multiple agents are learning together. Due to difficulty in exploration finding good policies is difficult and non-stationarity can make the algorithms forget good policies.

Since NFSP is adept at handling non-stationarity, we provide a new approach called Neural Fictitious Self Imitation and Play (NFSIP), that extends on NFSP in two major ways to handle sparse rewards: (a) We incorporate self imitation into NFSP, so as to replay past good experiences and ensure effective exploration; (b) All agents employ an average best response policy to deal with non-stationarity in NFSP. However, with sparse rewards, number of good policies are very few and averaging can make a good policy irrelevant. We provide supervised reinforcement based policy averaging to ensure good policies remain relevant.

We demonstrate that our approach is able to get significant improvement in performance over leading MARL approaches on three different problem domains from literature.

3 Background: Neural Fictitious Self Play (NFSP)

In Fictitious play (FP), a popular game-theoretic model of learning, agents repeatedly play a game, choosing the best response against average strategies of their opponents at each iteration. The average strategies converge to a Nash equilibrium for zero-sum games, potential games and identical interest games (i.e., cooperative multi-agent problems). FP is a theory on a normal-form representation, where each agent acts only once per one game, which is not suited to real problems at scale.

To overcome the scalability issue, Heinrich et al. (2015); Heinrich & Silver (2016) proposed an appropriately approximated (hence scalable to large-scale games) method for FP referred to as Neural Fictitious Self Play (NFSP). As with FP, agents in NFSP repeatedly play a game, storing their experiences in memory. Instead of computing the full-width best response strategy (i.e., compute best response by playing the entire game), they learn an approximate best response using Deep Q-Networks (DQN) Mnih et al. (2015). And instead of averaging their full-width strategies, they learn an approximate average strategy by using supervised learning (SL). Heinrich & Silver (2016) on deep neural networks.

Deep Q network with parameters θ^Q is trained using the following loss function:

$$\mathcal{L}(\theta^Q) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{M}_{RL}}\left[\Big(r + max_{a'}Q(s',a'|\theta^{Q'}) - Q(s,a|\theta^Q)\Big)^2\right]$$

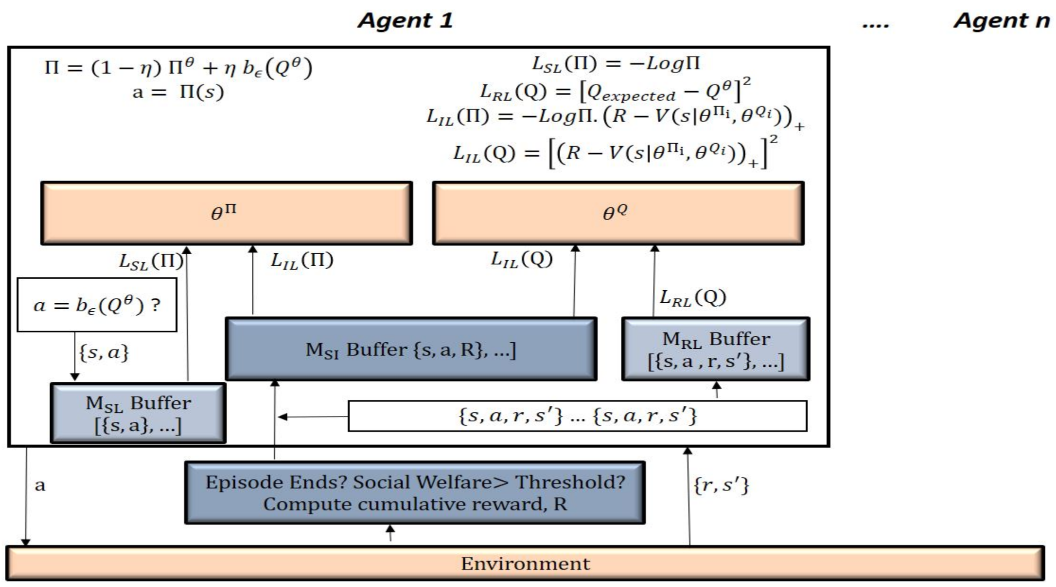
Agent 1
....
Agent n
$\Pi = (1-\eta)\,\Pi^{\theta} + \eta\, b_{\epsilon}(Q^{\theta})$
$a = \Pi(s)$
$L_{SL}(\Pi) = -Log\Pi$
$L_{RL}(Q) = [Q_{expected} - Q^{\theta}]^2$
$L_{IL}(\Pi) = -Log\Pi.\,(R - V(s|\theta^{\Pi_i}, \theta^{Q_i}))_+$
$L_{IL}(Q) = [(R - V(s|\theta^{\Pi_i}, \theta^{Q_i}))_+]^2$
θ^{Π}
θ^{Q}
$L_{SL}(\Pi)$
$L_{IL}(\Pi)$
$L_{IL}(Q)$
$L_{RL}(Q)$
$a = b_{\epsilon}(Q^{\theta})$?
$\{s, a\}$
M_{SI} Buffer {s, a, R}, ...]
M_{RL} Buffer [{s, a , r, s'}, ...]
M_{SL} Buffer [{s, a}, ...]
$\{s, a, r, s'\}$ ... $\{s, a, r, s'\}$
a
Episode Ends? Social Welfare> Threshold? Compute cumulative reward, R
$\{r, s'\}$
Environment

Figure 1: NFSIP

Average policy learning deep network with parameters θ^{Π} is trained by minimizing the loss of the policy:

$$\mathcal{L}(\theta^{\pi}) = \mathbb{E}_{(s,a)\sim\mathcal{M}_{SL}}\Big[-log(\Pi(s,a|\theta^{\Pi})\Big]$$

where $\mathcal{M}_{RL}$ and $\mathcal{M}_{SL}$ refer to the stored Reinforcement Learning and Supervised Learning experiences respectively.

3.1 Imitation Learning

Imitation learning (IL) enables a learner to imitate expert behavior given expert demonstrations. A wide variety of IL methods have been proposed in the last few decades. The simplest IL method among those is Behavioral Cloning (BC) Pomerleau (1991) which learns an expert policy in a supervised fashion without environment interactions during training. BC can be the first IL option to consider when enough demonstrations are available. Since it is often hard to obtain a large number of demonstrations in real-world environments, BC is often not the best choice for real-world IL scenarios. IL requires experts to generate the expert demonstrations, which may not be feasible in large scale problems.

Oh et al. (2018) provided a Self Imitation Learning (SIL) approach where (good) experiences generated during exploration are stored in a prioritized buffer based on cumulative reward achieved. During training it samples the experiences from this buffer and trains the neural networks only if the current Q/Value network is predicting a lower value for these experiences. This ensures that we only utilize good experiences in places where there is scope for improvement. It does so by computing the following value for experiences:

$$(\max(0, R(s,a) - V(s)))$$

where $R(s,a)$ = cumulative reward, is sum of immediate rewards from that state,action till end of episode and V(s) is state based baseline value.

4 Neural Fictitious Self Imitation Play, NFSIP

In this section, we describe our main algorithm, NFSIP for learning in the presence of sparse reinforcements. Algorithm 1 provides the pseudo code and Figure 1) provides the flow and dependencies. For each agent, in NFSIP and NFSP, we update the average policy network and Q-network parameters based on all the experiences. Since there are sparse reinforcements, it is imperative that updates from "good" experiences (i.e., ones that improve social welfare) are not overwritten by "bad" experiences. Therefore, in NFSIP, we have a separate self imitation loop at the end of each episode to not forget the learning from "good" experiences. In this self imitation loop, we update both the average policy and Q-network parameters based on difference in the reward obtained from the episode and the current value function estimate.

All agents in NFSIP also learn based on experiences generated from simultaneous play with other agents. An NFSIP agent interacts with its fellow agents and memorizes its experience of state transitions and its own best response behaviour in three memories, M_{RL}, M_{SL} and M_{SI}. NFSIP treats these memories as three distinct datasets suitable for deep reinforcement learning, supervised policy learning and self imitation respectively.

The NFSIP agent trains a neural network (Q), to predict action values from data in M_{RL} and M_{SI} using off-policy reinforcement learning. The resulting network defines the agents' approximate best response strategy, $b_\epsilon(Q)$, which selects a random action with probability ϵ and otherwise chooses the action that maximizes the predicted action values. The agent trains a separate neural network (Π), to imitate its own past best response behaviour using supervised classification on the data in M_{SL} and M_{SI}. This network maps states to action probabilities and defines the agent's average strategy while appropriately taking into account of good experiences stored in M_{SI}. During execution, the agent chooses its actions from a mixture of its two strategies, $b_\epsilon(Q)$ and Π.

In lines 4-11 of algorithm 1, each agent executes actions, stores the experiences in M_{RL} and also in M_{SL}, and uses the stored experiences to train the Q-network (action-value network) and policy networks. Once episode ends, in lines 12-17, we update the self imitation buffer, M_{SI} with experiences if social welfare (welfare of the entire system, including all agents) is higher than the set threshold for social welfare (best reward achieved so far). These experiences are updated to include cumulative rewards. In lines 18-24, Q and Π networks are trained with experiences from self imitation buffer, M_{SI} if Q-network is predicting a lower value for these experiences as compared to their actual (cumulative) reward.

We now provide two key insights employed in our algorithm to ensure improved performance.

4.1 Weighted Generalized Weakened Fictitious Play

Leslie & Collins (2006) have defined a generalization of fictitious play for approximate best responses as follows:

$$\pi^{t+1} \in (1-\alpha^{t+1})\pi^t + \alpha^{t+1} \cdot b_{\epsilon^t}(Q^t)$$

where $\alpha^t \to 0$, $\epsilon^t \to 0$, $||Q^t - R(\pi^t)|| \to 0$ as $t \to \infty$

NFSP employs maximum log likelihood (using Loss as negative log likelihood) for learning the above mixture of past policy, π^t and current approximate best response policy, $b_{\epsilon^t}(Q^t)$ based on the observed samples (i.e., best response actions taken at each iteration). The standard maximum likelihood principle implicitly places equal weight on each of the observations in the sample, but depending on the extent to which the model and the true data generating process deviate this can be improved upon. Taking the example of coin toss, if after 1000 iterations, if we observed 700 heads and 300 tails, maximum likelihood will predict a biased coin with 0.7 and 0.3 probability. However, this is incorrect as the sampled data was biased. This issue happens frequently in sparse reinforcement setting as sparse but good experiences come by rarely. So, samples data is bound to have rare occurrences of them, causing maximum likelihood (for average policy network) to result in bad local optima.

One of the ways to improve the model is to use weighted maximum likelihood (such methods has been employed in many domain such as risk management in Finance Steude (2011), image denoising for image processing Deledalle et al. (2009)). Steude (2011) to minimize the risk, down weighted the observations that bear a high probability of being destructive outliers. they show that it can considerably improve the forecast accuracy for a variety of data sets and different time series models can be realized. Deledalle et al. (2009) derived the weights in a data driven manner. The weights are iteratively refined based on both the similarity between noisy patches and the similarity of patches extracted from the previous estimate.

For solving MARL with sparse reward setting, we build on similar ideas. Specifically, we increase weight for better experiences in both policy and Q-networks. These weights are dynamically updated based on the current state of learning. Since we only want to increase the weight of good experiences, we will not have negative weight in the process. For average policy network, we employ the following additional loss based on experiences in self imitation memory:

$$\mathbb{E}_{(s,a,R)\sim M_{SI}}\Big[-log(\Pi(s,a))\cdot[R(s,a)-V(s)]_+\Big]$$

where

$$[R(s,a)-V(s)]_+ = max(0, R(s,a)-V(s))$$

On similar lines, we also add an additional weight to the Q-network loss based on self imitation memory

$$\big(r(s,a)+\gamma\max_{a'}Q'(s',a')-Q(s,a)\big)^2+\alpha\cdot\big([R(s,a)-V(s)]_+\big)^2$$

4.2 Self Imitation Learning for Cooperative Multiple Agent Problems

Self imitation learning in single agent case imitates past "good" experiences and prioritizes learning with those "good" experiences. However, in multi-agent problems, due to simultaneous learning of agents, past good experience for an agent may not be a "good" experience if other agents have changed their policy. More importantly, in a cooperative setting, individual good experiences may not be maximizing social welfare. So we make following changes:

- We judge the goodness of any experience not just based on its own reward but also based on social welfare. We only store experiences for self imitation if social welfare is above a certain threshold value. We start with a very low threshold value and gradually increase it as we explore better experiences (providing higher social welfare).
- Presence of multiple learning agents makes the environment non stationary and therefore in order to avoid utilizing old experiences (that may not be valid any more), we periodically remove expert data (self) generated for self imitation process. We do so when we encounter a better social welfare solution and adjust the threshold value accordingly. This will ensure that there always expert experiences to train with that are not too old and provide higher social welfare.
- We train only with those experiences where neural network is predicting a lower value than the actual value (cumulative reward) obtained by the agent. Therefore, the max operator is given by:

$$(.)_+ = \begin{cases} max(O,(R(s,a)-V(s))) & \text{if } W >= W_T \\ 0 & \text{otherwise} \end{cases}$$

  Where $R(s,a)$ = Cumulative reward of agent, W = Welfare of the entire system (social welfare) and W_T = Threshold value for social welfare

5 Discussion

In this section, we provide a discussion on key challenges that exist typically in cooperative MARL problems and how NFSIP addresses these challenges.

First, we discuss about a problem faced by many centralized learners Sen & Sekaran (1995); Boutilier (1996); Claus & Boutilier (1998); Lowe et al. (2017); Foerster et al. (2018), i.e., curse of dimensionality. As the size of state and action spaces grows exponentially with number of agents, centralized learners can face severe scalability issues. Since NFSIP (like NFSP) is a decentralized method, the curse of dimensionality does not pose a challenge as each learning agent can run on a different computing thread. Another approach that has been employed for addressing curse of dimensionality is by exploiting homogeneity and anonymity in agent models Nguyen et al. (2017). However, in problems of interest in this paper, due to the presence of a global state (task states) that is affected by individual agent actions, the method employed to generate counts of agents becomes inapplicable.

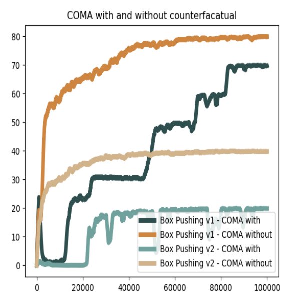

Figure 2: Credit assignment

Algorithm 1: Neural Fictitious Self Imitation and Play, NFSIP

```
1:  Initialize policy network($\theta^{\Pi}$), action-value network($\theta^{Q}$) and target action-value network($\theta^{Q'}$)
    networks
2:  best reward achieve so far = $-\infty$
3:  while Not Converged do
4:      policy = $\begin{cases} b_\epsilon(Q) & \text{with probability} \quad \eta \\ \Pi & \text{with probability} \quad 1-\eta \end{cases}$
5:      for every time step do
6:          Simulate agents for 1 step
7:          Store experiences in $M_{RL}$
8:          Store experiences in $M_{SL}$ if agent took best response action ($b_\epsilon(Q)$)
9:          for all agents do
10:             Sample from $M_{RL}$, train action-value network:
                $\mathcal{L}(\theta^Q) = \mathbb{E}_{(s,a,r,s')}\Big[\Big(r + max_{a'}Q(s',a'|\theta^{Q'}) - Q(s,a|\theta^Q)\Big)^2\Big]$
11:             Sample from $M_{SL}$, train policy network: $\mathcal{L}(\theta^\pi) = \mathbb{E}_{(s,a)}\Big[-log(\Pi(s,a|\theta^\Pi)\Big]$
12:     if Episode reward $>$ best reward achieve so far then
13:         Reset prioritized experience buffer
14:         best reward achieve so far = Episode reward
15:     if Episode reward $>=$ best reward achieve so far then
16:         Compute cumulative reward, $R$
17:         Store experiences in prioritized experience buffer prioritized on $R$
18:     for some iteration do
19:         for all agents do
20:             Sample from prioritized replay buffer
21:             Train action-value network: $\mathcal{L}(\theta^Q) = \mathbb{E}_{(s,a,R)}\Big[([R(s,a) - V(s|\theta^Q,\theta^\Pi)]_+)^2\Big]$
22:             Train policy network:
                $\mathcal{L}(\theta^\pi) = \mathbb{E}_{(s,a,R)}\Big[-log(\Pi(s,a|\theta^\Pi)) \cdot [R(s,a) - V(s|\theta^Q,\theta^\Pi)]_+)\Big]$
23:             $[R(s,a) - V(s)]_+ = \max\big(0, R(s,a) - V(s)\big)$
24:             $V(s) = \sum_a \pi(s,a) \cdot Q(s,a)$
25:     Update target action-value network periodically
```

Second, we consider the credit assignment problem. Since agents receive reward as a team, a key challenge while learning is understanding how much each agent contributed to receiving the reward. A leading approach Foerster et al. (2018), employs marginal utilities, i.e., difference in reward with the agent included and reward without the agent included. However, in problems of interest in this paper, agents have to coordinate to complete tasks (i.e, extinguishing a fire, pushing a box). That is to say, if n agents are required to accomplish a task, then $n-1$ agents will not be able to complete the task. So, marginal utility for every agent will be equal to the total reward to be received by the entire team of n agents. Since that would be inaccurate, we do not go for marginal utility based credit assignment. Instead, we employ the reward strategy described by Panait *et al.* Panait & Luke (2005), where rewards are divided equally among all agents.

Third, we consider the non-stationarity introduced due to multiple agents learning at the same time. We employ the NFSP type of policy averaging and short experience buffers to reason with non-stationarity.

Finally, the issue of sparse reward has not received much attention in MARL literature, even though it has been studied extensively in single agent literature Oh et al. (2018); Zhu et al. (2018); Večerík et al. (2017); Hester et al. (2018). Through a synergistic combination of Neural Fictitious Self Play (NFSP) and Self Imitation Learning (SIL)[1], NFSIP is able to provide significant improvements on complex domains with sparse reinforcements.

[1]NFSP handles non-stationarity to ensure SIL is able to learn from the right experiences. SIL ensures good experiences are not forgotten so that NFSP can move to higher quality equilibria.

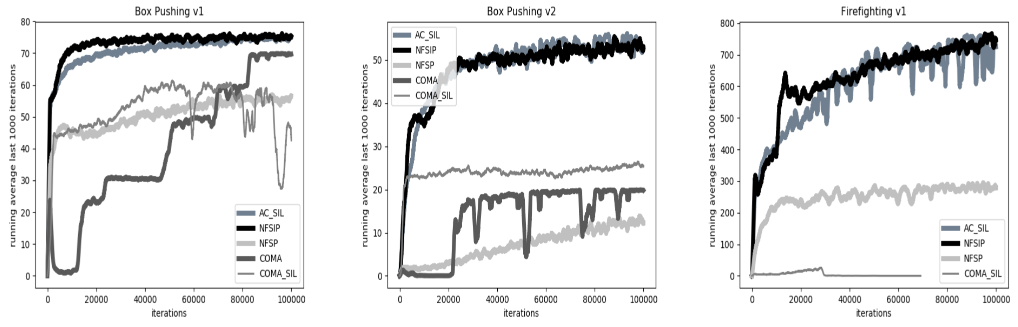

Figure 3: Box pushing v1, v2 and Fire Fighting v1

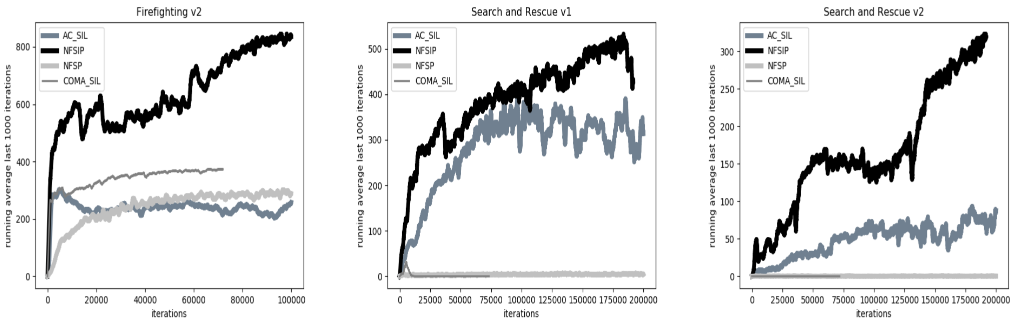

Figure 4: Fire Fighting v2, Search and Rescue v1 and v2

6 Experimental Section

In this section, we evaluate the performance of our NFSIP approach in comparison to leading approaches for cooperative MARL. We perform the comparison on three different problem settings from literature: (a) Box Pushing Seuken & Zilberstein (2012); (b) Fire Fighting Oliehoek et al. (2008); and, (c) Search and Rescue Nanjanath et al. (2010); Parker et al. (2016). We extended these problem settings to ones with many agents and larger state space.

We compare against the following leading approaches for cooperative MARL: (a) COMA Foerster et al. (2018); (b) NFSP Heinrich & Silver (2016); (c) AC-SIL: Multi-agent extension of SIL Oh et al. (2018); (d) COMA_SIL: An SIL extension for COMA.

We now provide more details on the specific problem domains:

Box pushing problem Seuken & Zilberstein (2012): Multiple agents need to coordinate and push boxes of different sizes to their goal locations in a grid world. Each agent has 6 possible actions to take: $\{$move left, move right, move up, move down, act on the task, stay$\}$. To successfully push a box, certain number of agents need to push the box simultaneously. For this domain, we created simpler instances with smaller grid sizes as benchmark algorithms were unable to learn at all on larger problem instances. We considered a 4x4 grid with 5-agents in box pushing. There are 4 boxes. We created different versions of this problem by changing the number of agents required to push the boxes: (V1) Any single agent can push the box; and (V2): To push any box at least 2 agents need to cooperate and simultaneously act on it.

Firefighting problem Oliehoek et al. (2008): In this problem setting we have a 6x6 grid with 10 agents (fire trucks), fires are spread over different locations. Fire trucks need to act on fires to put them out, number of trucks needed to put out the fire depends on its intensity (low/high). We created different versions of the problem as follows: (V1): 2 agents can put out the fire with probability 0.9, more than 2 agents can put out the fire with probability 1; and (V2): Intensity of fire will increase from low to high with probability 0.2 at every time step. Low intensity fire: “2 agents can put it out probability 0.9, more than 2 agents with probability 1". High intensity fires: "2 agents can put it out with probability 0.75, 3 agents can put it out with probability 0.9 and more than 3 agents can do it with probability 1".

Search and Rescue Nanjanath et al. (2010); Parker et al. (2016): Here different types of agents (such as firetrucks and ambulances) need to coordinate with each other. In this problem setting we have a 6x6 grids with 5 ambulances and 5 firetrucks. Number of firetrucks and ambulances needed to complete the task depends on difficulty of the scenario. Here we created different scenarios as follows: (V1): Minimum 1 fire truck and 1 ambulance needs to cooperate to complete the task; and (V2): Difficulty of the search and rescue scenario will increase from low to high with probability 0.2 if operation is not completed. If difficulty level is low then minimum 1 ambulance and 1 fire truck can complete search and rescue, if difficulty level is high then minimum 2 ambulances and 2 firetrucks are needed to carry out the operation.

Before we provide the main result, we show that the counterfactual way of handling the credit assignment fares badly in problems of interest in this paper. Figure 2 shows that COMA without counterfactual performs better on both box pushing problems. We have made the same observation in other problems as well. In COMA without counterfactual version, it should be noted that we used same credit assignment scheme as we used for our NFSIP approach.

Here are the key observations from the charts in Figures 3 and 4:

- On the simplest problems, i.e., ones in box pushing, COMA is able to learn good policies. However, NFSIP and AC-SIL perform the best even on these simplest problems.
- NFSIP is able to outperform both NFSP and COMA on all 6 scenarios
- NFSIP is able to perform as good as or better than AC_SIL. In the last scenario (Search and Rescue V2), NFSIP is able to get a result that is 6 times that of AC_SIL.
- Due to counterfactual baseline computation for every action, COMA is very slow as compared to our approach. On all 6 problem settings, each agents has 6 possible actions to choose from, on average COMA is more than 6 times slower than NFSIP. In our problem settings, on 6x6 grids, where action set size is 6, NFSIP is on average taking between 1-2 days for training, COMA is taking between 1-2 weeks. Since COMA need to compute counterfactual baseline for every action, COMA will take more and more time as we move to a larger action set.

In all experiments, in our approach, all agents share parameters in both policy as well as best response network. i.e, there is one policy network and one best response network that takes agents Ids as input to distinguish between them.

Neural Network Details: In all networks (Q/Policy networks for NFSP, actor/critic networks for COMA) we have 2 hidden layers with 32 nodes each, after every hidden later we used layer norm. In all experiments for NFSP and NFSIP following are the parameters details: $\{LearningRate_Q$, $LearningRate_\Pi$, η, $\epsilon\}$ =$\{10^{-3}$, 10^{-4}, 0.2, 0.5$\}$. In all experiments with COMA and AC_SIL following are the parameters details: $\{LearningRate_{Critic}$, $LearningRate_{Actor}$, $\epsilon\}$ =$\{10^{-3}$, 10^{-4}, 0.2*0.5 = 0.1$\}$. We gradually reduce epsilon value as follows, after every 500 iteration we reduce epsilon to a factor of 0.98.

REFERENCES

Adrian K Agogino and Kagan Tumer. Quicr-learning for multi-agent coordination. 2006.

Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pp. 195–210, 1996.

George W. Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 13(1):374–376, 1951.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. 1998.

Charles-Alban Deledalle, Loïc Denis, and Florence Tupin. Iterative weighted maximum likelihood denoising with probabilistic patch-based weights. *IEEE Transactions on Image Processing*, 18 (12):2661–2672, 2009.

Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Thomas Haynes and Sandip Sen. Evolving behavioral strategies in predators and prey. In *International Joint Conference on Artificial Intelligence*, pp. 113–126. Springer, 1995.

Thomas Haynes, Kit Lau, and Sandip Sen. Learning cases to compliment rules for conflict resolution in multiagent systems.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *International Conference on Machine Learning*, pp. 805–813, 2015.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.

David S Leslie and Edmund J Collins. Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298, 2006.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pp. 6379–6390, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Maitreyi Nanjanath, Alexander J Erlandson, Sean Andrist, Aravind Ragipindi, Abdul A Mohammed, Ankur S Sharma, and Maria Gini. Decision and coordination strategies for robocup rescue agents. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pp. 473–484. Springer, 2010.

Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. Policy gradient with value function approximation for collective multiagent planning. In *Advances in Neural Information Processing Systems*, pp. 4319–4329, 2017.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. *arXiv preprint arXiv:1806.05635*, 2018.

Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

James Parker, Ernesto Nunes, Julio Godoy, and Maria Gini. Exploiting spatial locality and heterogeneity of agents for search and rescue teamwork. *Journal of Field Robotics*, 33(7):877–900, 2016.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

Sandip Sen and Mahendra Sekaran. Multiagent coordination with learning classifier systems. In *International Joint Conference on Artificial Intelligence*, pp. 218–233. Springer, 1995.

Sven Seuken and Shlomo Zilberstein. Improved memory-bounded dynamic programming for decentralized pomdps. *arXiv preprint arXiv:1206.5295*, 2012.

Sven C Steude. Weighted maximum likelihood for risk prediction. 2011.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

Matej Večerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.