MAPEL: Multi-Agent Pursuer-Evader Learning using Situation Report

Sagar Verma
CVN, CentraleSupélec, Université Paris-Saclay
sagar.verma@centralesupelec.fr

Richa Verma
TCS Innovation Lab
richa15054@iiitd.ac.in

P.B. Sujit
IIIT Delhi
sujit@iiitd.ac.in

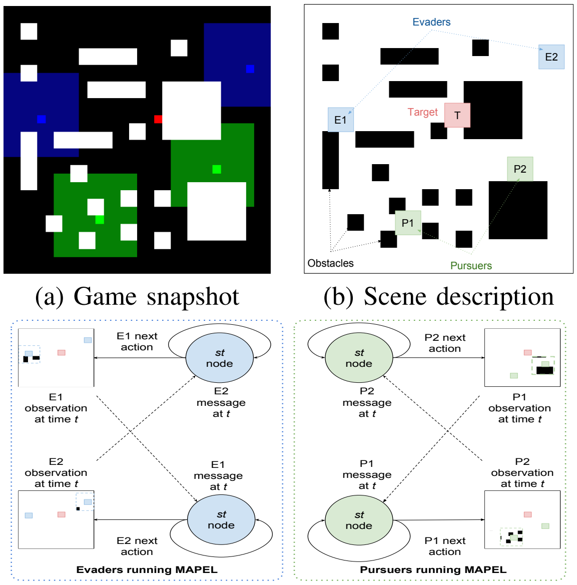
Abstract*—In this paper, we consider a territory guarding game involving pursuers, evaders and a target in an environment that contains obstacles. The goal of the evaders is to capture the target, while that of the pursuers is to capture the evaders before they reach the target. All the agents have limited sensing range and can only detect each other when they are in their observation space. We focus on the challenge of effective cooperation between agents of a team. Finding exact solutions for such multi-agent systems is difficult because of the inherent complexity. We present Multi-Agent Pursuer-Evader Learning (MAPEL), a class of algorithms that use spatio-temporal graph representation to learn structured cooperation. The key concept is that the learning takes place in a decentralized manner and agents use situation report updates to learn about the whole environment from each others' partial observations. We use Recurrent Neural Networks (RNNs) to parameterize the spatio-temporal graph. An agent in MAPEL only updates all the other agents if an opponent or the target is inside its observation space by using situation report. We present two methods for cooperation via situation report update: a) Peer-to-Peer Situation Report (P2PSR) and b) Ring Situation Report (RSR). We present a detailed analysis of how these two cooperation methods perform when the number of agents in the game are increased. We provide empirical results to show how agents cooperate under these two methods.

***Index Terms*—multi-agent learning; deep reinforcement learning; recurrent neural network**

I. Introduction

Multi-agent systems have received much attention in the past decade [?], [?], [?], [?]. In such systems, agents share a common environment, where they act independently or in cooperation with each other to achieve a combined goal. We focus on the problem where multiple agents achieve a single task in cooperation with each other. In such problems, the agents must have the ability to handle unknown and uncertain scenarios and take the success of the whole team into account.

A multi-agent pursuit-evasion game complexity depends on many variables like the type of environment, the observation of the agents, their actions, cooperation strategy, and the reward structure. Being complex and dynamic, pursuit-evasion problems are challenging to solve [?]. Such complexities have been addressed by stochastic modeling of agents motion in [?], [?]. There has been growing interest in modeling the game, in which the evader is intelligent and has certain sensing capabilities [?]. This paper focuses on the problem of partial observation of agents where structured message passing is used for cooperation.



(a) Game snapshot

(b) Scene description

(c) Proposed method

Fig. 1: Top-left image shows the rendering of our multi-agent pursuer-evader game. Top-right image shows the labelled description of the game environment. Bottom image shows the proposed cooperation method.

We present a zero-sum game based on pursuit and evasion between two teams of equal number agents. Since either the pursuer or the evader wins the game, therefore, the game can be represented as a zero-sum game. The multi-agent pursuit-evasion problem is shown in Figure 1. We assume partial observability of the environment to ensure that the solution is usable in many real-world applications that closely correspond to the task in hand. However, learning becomes more difficult under partial observation along with complex interactions between agents and the environment. Each agent perceives the environment locally and even though the effect of other agents' actions on the environment is visible but the agents, themselves, are not. Reinforcement learning (RL) has been used to solve multi-agent pursuit-evasion in [?], [?], [?], [?]. Recent works like [?], [?], [?], [?] use deep reinforcement learning for different multi-agent problems where centralized policy learning is employed. All these works deal with full observation and are not suitable for our problem. We propose MAPEL, a class of deep reinforcement learning based methods that uses spatio-temporal graphs for structured cooperation between agents under partial observation.

To solve our multi-agent pursuit-evasion game, we present MAPEL which uses spatio-temporal graphs to structure the cooperation between agents in a team. We propose using abstract messages called situation reports which are shared among agents for cooperation. We present two different methods for situation report update which are based on dense and sparse communication. MAPEL can handle pursuers and evaders which move at the same speed throughout the game, which means that neither pursuers nor evaders have an advantage over each other. We show that MAPEL cooperation methods lead to a high degree of cooperation between agents. We also show how the two cooperation methods perform when the number of agents in the team is increased.

The remainder of this paper is organized as follows. Section 2 mentions the existing works related to this paper. Section 3 describes the problem and its formulation as a multi-agent reinforcement learning (MARL) problem. Section 4 presents MAPEL and other proposed baselines. Section 5 explains the experimental setup followed by the results and their explanation in section 6. Section 7 concludes this paper.

II. Related Work

Reinforcement learning has been successfully used to play games like Atari [?] and Go [?]. In [?], the authors suggest an approach based on hierarchical RL for the same, while enabling the players to learn through tasks with less complexity. Multi-agent reinforcement learning (MARL) consists of a set of learning agents that share a common environment [?]. Learning in such a framework is fundamentally difficult because of the interaction arising between the agents and the environment and amongst themselves. Conventional decentralized learning techniques like Q learning for each agent [?] assume the other agents to be a part of the environment. Such methods don't work in multi-agent settings because the theoretical convergence guarantee no longer holds and makes the learning unstable due to the fact that changes in the policy of any agent will affect the policies of the other agents, as well [?].

Joint action learning or centralized policy learning is one way to do multi-agent reinforcement learning. [?] present a deep policy inference Q-network that targets multi-agent systems composed of controllable agents. A centralized policy for the controllable agent is learned from its raw observations. [?] presents joint and independent policy learning methods. In an independent policy learning method, the joint learned policy is transferred to individual agents in an iterative manner. [?] discusses why centralized policy learning fails in case of multi-agent setting and presents methods to learn policy for heterogeneous agents as well as homogeneous agents. Sunehag et al. [?] discuss the problem of "lazy agents" which is when some agents remain inactive when a centralized policy learning is used. They present a value-decomposition network which enables better reward sharing between agents to solve the problem of inactive agents.

Decentralized learning requires effective cooperation between different agents. [?] suggest learning with opponent-learning awareness method in which each agent anticipates other agent's policy. This method only works for complete observation. [?] discusses the problem of experience replay in multi-agent deep reinforcement learning (MA-DRL). They state that transitions stored in experience replay memory (ERM) can become outdated because agents update their policies in parallel. They apply leniency to MA-DRL by mapping agents state-action pairs to decaying temperature values that control the amount of leniency applied towards negative policy updates that are sampled from the ERM. They also state that this help in better cooperation among agents. [?] use actor-critic to learn policies for complex cooperation. [?] uses centralized critic to estimate the Q-value, decentralized actors are used to optimize agents' policies, and counterfactual baselines are used to solve multi-agent credit assignment problem. [?] presents co-evolution methods to learn better coordination between agents. [?] present a method to solve cooperation between agents that can act selfishly.

Some multi-agent problems can be explicitly described as graphs. [?] presents cooperative reinforcement learning for multiple agents in StarCraft game. [?] uses graph representation with reinforcement learning for coordination and cooperation in multi-agent patrol task. Efficient state representation based on the distance between agents and different game entities is used to reduce the observation state complexity. [?] presents a flag coordination game where graph structure is explicitly present and is utilized to model multi-agent coordination. Some problems where there are no explicit graph structures present, game states can be decomposed into some weak time-varying structures. Such structures can be learned using factor graph representation and graphical learning methods. [?] discusses use cases where cooperation is explicitly required. A genetic algorithm variation is used to solve the adaptive team of agents (ATA) problem. Their method can adapt an agent to a new role based on the overall structure of the environment. [?] presents joint policy learning method for coordinated reinforcement learning through structured communication between agents. [?] presents a way to decompose a global Q-function into local Q-function based on the task decomposition between agents expressed using factor graphs. [?] divide agents into cliques based on specific tasks. [?] uses factor graphs to learn implicit structures present in multi-agent settings. Factor graphs reduce the action and observation space and learning agents' policy becomes easier.

In the literature, RL has been used earlier for the classic pursuer-evader game [?]. In [?], a learning technique for multi-player pursuit-evasion games is presented for discrete state and action spaces. The proposed algorithm is only applicable for multi-player pursuit-evasion games with superior pursuers (in terms of speed). The article [?] is another work suggesting a technique using learning in differential multi-player pursuit-evasion games that have superior evaders. In [?] hierarchical decomposition is used to solve games having two pursuers and one evader.

III. Problem Definition

We model the multi-agent pursuer-evader problem as a grid world of dimension $M \times N$ in which obstacles are placed randomly (uniform distribution $\mathcal{N}(0, \sigma)$). In this grid, there are P pursuers, E evaders, and a single target T. At any time t, a pursuer $p \in P$ has the global knowledge about all pursuer locations and the current target location in the environment. An evader $e \in E$ is assumed to know the locations of all other evaders and the target. We assume each agent can sense a rectangular region of length l and width w. However, the agents cannot sense on the other side of the obstacle. That is, a pursuer can detect an evader if they are in line-of-sight and within the sensed region. The speed of all the pursuers and the evaders is given by v and remains constant throughout the game. The target, T remains stationary throughout the game.

A game starts with randomly sized obstacles placed on the grid at random locations as shown in Figure 1 (for a 2-pursuers vs 2-evaders game). The target is spawned at a random location near the middle of the grid $(M/2, N/2)$ and it is of length t_s. The pursuers and the evaders are randomly spawned on the opposite sides of the grid. The pursuers and the evaders can move to any of the adjacent cells of the grid only if the cell is either empty or occupied by any of the agents. An agent reaches the target when its location is same as the target's location. Also, a pursuer captures an evader only if their locations are the same. Once an evader is captured by a pursuer, it cannot move anywhere else but the pursuer can move to an adjacent cell after catching the evader.

There are three conditions for a game to complete.

1) An evader reaches the target, in which case the evaders win the game.
2) A pursuer reaches the target before an evader, in which case the pursuers win the game.
3) All the evaders are captured by the pursuers, in which case the pursuers win the game.

Based on the three different winning criteria we have the following reward structure:

1) When the evaders win by capturing the target, a reward of $w^e = 0.5$ is awarded to them and a penalty of $w^p = -0.5$ is given to the pursuers.
2) When the pursuers win by reaching the target before the evaders, a reward of $w^p = 0.5$ is awarded to, and a penalty of $w^e = -0.5$ is given to the evaders.
3) When the pursuers win by capturing all the evaders, a reward of $w^p = 1$ is awarded to the pursuers and a penalty of $w^e = -1$ is given to the pursuers.

Rewards are equally divided among all the agents of a team. This makes it sure that agents in a team do not compete with each other.

IV. Methods

In this section we first present a naive method in which an agent greedily moves towards the target, followed by the second method which is a multi-agent formulation of deep Q-learning and then we introduce the proposed method MAPEL with two different cooperation strategies.

A. Naive Method

A naive agent tries to move towards the target, T. Each agent has a partial view of the environment and knows the location of the other agents of its team. It also knows the location of the target. A naive agent moves towards the target in a straight line. If the next location on the line of sight towards the target is obstructed, it randomly chooses an adjacent location that is closest to the line. If a pursuer observes an evader in its observation space, it computes the shortest path to the evader and chooses its next location along that path. Similarly, if a pursuer/evader observes the target in its observation space, it computes the shortest path to the target and chooses its next location along that path. Also, if a pursuer observes the target and an evader or multiple evaders in its field of view, it computes the shortest paths to all of them and chooses its next location along the path that has the smallest length.

B. Multi-agent Q-learning

An N-agent stochastic game $\mathcal{E}$ is formalized by the tuple $\mathcal{E} = (\mathcal{S}, \mathcal{A}^1, \ldots, \mathcal{A}^N, \mathcal{R}^1, \ldots, \mathcal{R}^N, \mathcal{T}, \gamma)$, where $\mathcal{S}$ denotes the state space, and $\mathcal{A}^i$ is the action space of agent $j \epsilon \{1, \ldots, N\}$. The reward function for agent j is defined as $\mathcal{R}^i : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \rightarrow \mathbb{R}$, determining the immediate reward. The transition probability is given by $\mathcal{T} : \mathcal{S} \times \mathcal{A}^1 \times \ldots \mathcal{A}^N \rightarrow Pr(\mathcal{S})$. $Pr(\mathcal{S})$ is the collection of probability distributions over the state space $\mathcal{S}$. The goal of agents is to find a policy π which maximizes the expected return G_t, which is the discounted sum of rewards given by $G_t = \sum_{i=1}^{N} \sum_{\tau=t}^{T} \gamma^{\tau - t} \mathcal{R}_\tau^i$, where T is the time-step when an episode ends, t denotes the current time-step, $\gamma \epsilon [0, 1)$ represents the reward discount factor, and $\mathcal{R}_\tau^i$ is the reward received at time-step τ by agent $\mathcal{A}^i$.

The agents choose actions according to their policies. For agent i, the corresponding policy is defined as $\pi^i : \mathcal{S} \rightarrow Pr(\mathcal{A}^i)$, where $Pr(\mathcal{A}^i)$ is the collection of probability distributions over agent i's action space $\mathcal{A}^i$. The joint policy of all the agents is given by $\pi : \pi^1 \times, \cdots \times \pi^N$. The joint actions of all the agents is given by $a : \mathcal{A}^1 \times, \ldots, \times \mathcal{A}^N$. The value function of agent i given state s under the joint policy π is written as the expected cumulative discounted future reward:

$$v_\pi^i(s) = v^i(s; \pi) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\pi, p} \big[r_t^i | s_0 = s, \pi \big] \tag{1}$$

The Q-function can then be defined within the framework of N-agent game based on the Bellman equation given the value function in equation (1) such that the Q-function $Q_\pi^i : \mathcal{S} \times \mathcal{A}^1 \times, \ldots, \mathcal{A}^N \rightarrow \mathbb{R}$ of agent i under the joint policy π can be formulated as

$$Q_\pi^i = R^i(s, a) + \gamma \mathbb{E}_{s' - p} \big[v_\pi^i(s') \big], \tag{2}$$

where s' is the state at the next time step. The value function v_π^i can be expressed in terms of the Q-function in equation (2) as

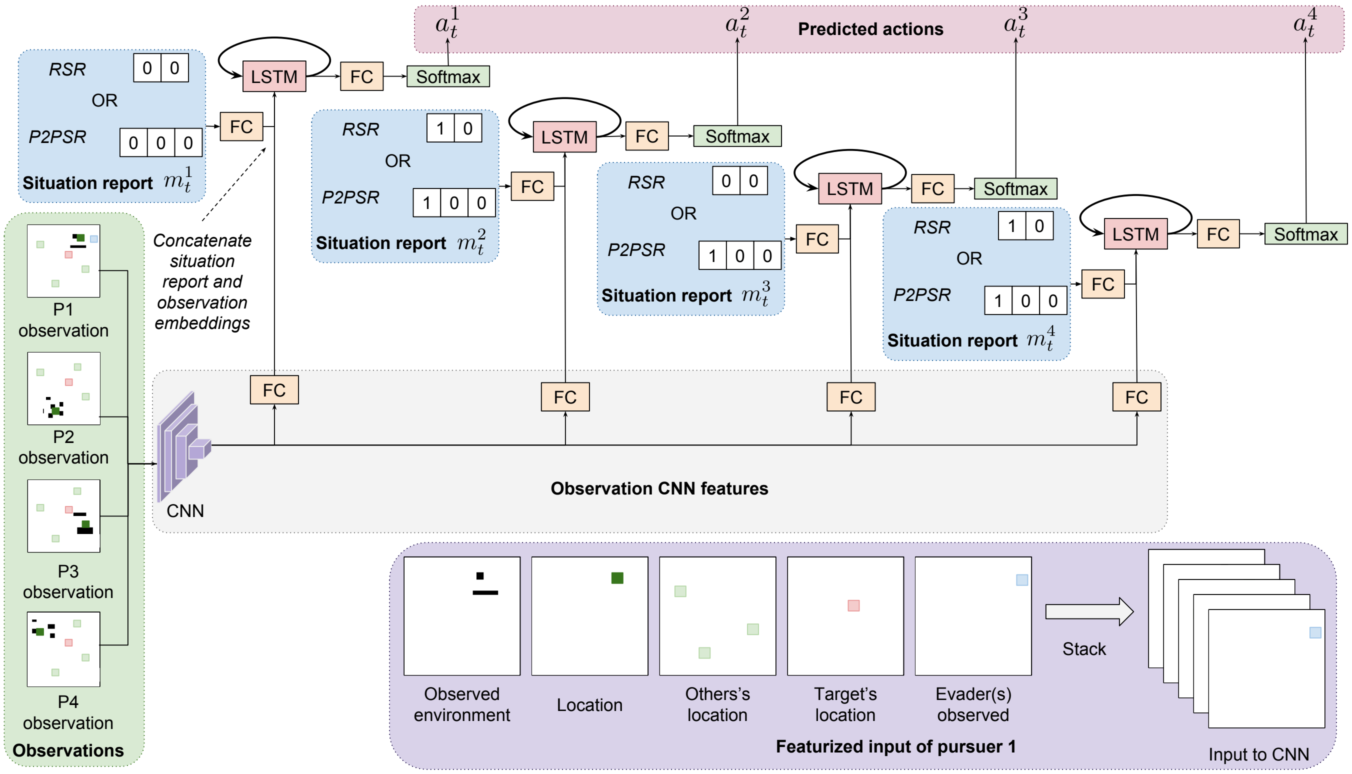


Fig. 2: Generalized architecture of MAPEL.

$$v^i_\pi = \mathbb{E}_{a \ \pi}\big[Q^j_\pi(s,a)\big]. \tag{3}$$

The Q-function for N-agent game in equation (2) extends the formulation for a single-agent game by considering the joint action taken by all agents a, and by taking the expectation over the joint action in equation (3).

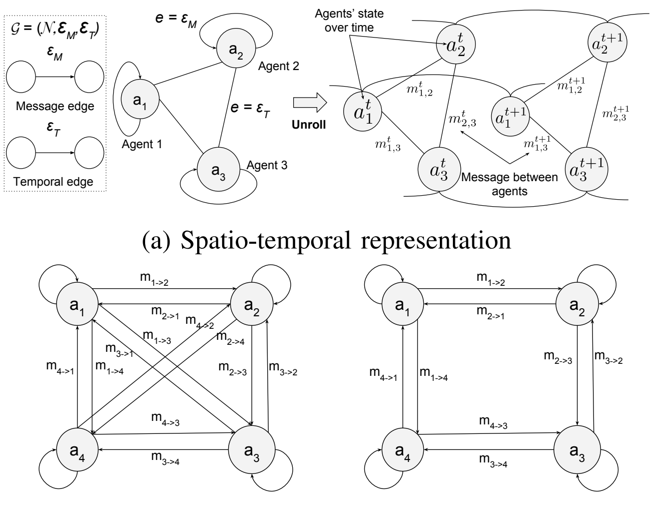
C. Multi-Agent Pursuer-Evader Learning (MAPEL)

In the Q-learning method presented in the previous section, the joint policy π is dependent only on the current observation of all the agents combined. It is impossible for an agent to know anything about the observation of the other agents in that setting. Also, the size of the combined observation and action spaces increases exponentially with the number of agents. For a large number of agents, this could be problematic.

In this section, we present a spatio-temporal (st) architecture called MAPEL which allows agents to learn their individual policies by sharing their observations with each other by cooperating via situation reports. We represent a team of agents as an st-graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}_M, \mathcal{E}_T)$, where $\mathcal{N}$ denotes the total number of agents, $\mathcal{E}_M$ is the total number of edges between the agents i.e. the edges used to pass situation reports, and $\mathcal{E}_T$ is the number of edges connecting agents at time T. Figure 3 shows an example st-graph capturing agent-agent interactions during a game. In the unrolled st-graph, two agents at a given time step t are connected with an undirected *spatio-temporal* edge $e = (a_i, a_j)\epsilon\mathcal{E}_M$, and two nodes at adjacent time steps are connected with an undirected *temporal* edge *iff* $(a_i, a_j)\epsilon\mathcal{E}_T$.

We parameterize the nodes $\mathcal{N}$ and edges $\mathcal{E}_T$ using RNNs in our st-graph. The edges $\mathcal{E}_M$ are used by nodes to pass situation reports to each other. The situation reports are used by agents to compute their actions at time t. The network architecture of MAPEL is illustrated in figure 2, each agent is represented by an RNN, the agents compute their observational features using a CNN and pass their own observations to other agents via situation reports. Each agent uses the situation reports received by other agents along with its current observation to compute its next action. RNN maintains history information about an agent. The situation report coupled with RNN is used to handle partial observability. Situation report provides an abstract and clear representation of the observation. This helps in reducing the hidden state representation noise which arises due to other agents changing their strategies. For example, if a pursuer observes the target or an evader, it can inform other pursuers about its observation via situation report. This could help the other pursuers in changing their decision to not go in the direction of this particular pursuer and search in other areas for the target or other evaders.

In real-world applications, we can have hundreds of agents and interaction among all of them may not be possible due to some physical constraints or simply because of high computational complexity. In most of the cases, it is not necessary to have dense communication between all the agents. Sparse communication structures can be used to learn effective cooperation. We present two situation report update methods that use the structures present in the game.

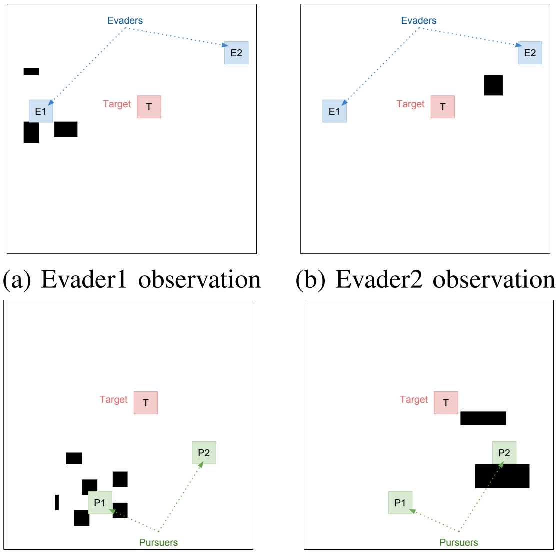


(a) Spatio-temporal representation

(b) P2PSR

(c) RSR

Fig. 3: Spatio-temporal representation used for cooperation between agents. Top image shows cooperation between three agents using spatio-temporal graph unrolled in time. Bottom-left image shows peer-to-peer situation report method between four agents and bottom-right image shows ring situation report method between four agents.



(a) Evader1 observation

(b) Evader2 observation

(c) Pursuer1 observation

(d) Pursuer2 observation

Fig. 4: Partial observation of individual agents in 2-pursuers vs 2-evaders game. Each agent can see everything in its observation space and knows about the target and other team members' locations. An agent cannot see the observation of its team members.

Peer-to-Peer Situation Report (P2PSR): In Peer-to-Peer Situation Report method, all the agents can share situation report with each other. This is the case of dense communication. This means that in our st-graph representation for $\mathcal{N}$ nodes, we have $\mathcal{E}_M = \mathcal{N}(\mathcal{N}-1)/2$ edges. Figure 3 shows the st-graph representation of P2PSR. This type of cooperation is required when an agent wants to know what other agents are observing so that it does not explore their regions. The objective of the agents then becomes to minimize the search time and exploration area to complete the task.

Ring Situation Report (RSR): In Ring Situation Report method, agents are randomly chosen to form a ring. Each agent can only pass messages to its adjacent agents. The st-graph representation for this type of cooperation is given by Figure 3. For $\mathcal{N}$ nodes, we have $\mathcal{E}_M = \mathcal{N}$ edges for $\mathcal{N} > 2$. This type of cooperation can be used to cordon off an area and search inside it. This does not require all the agents to know about the other agents' observations. An agent only needs to know what its adjacent agents are observing. This decreases the number of messages required to cooperate.

V. Experimental Setup

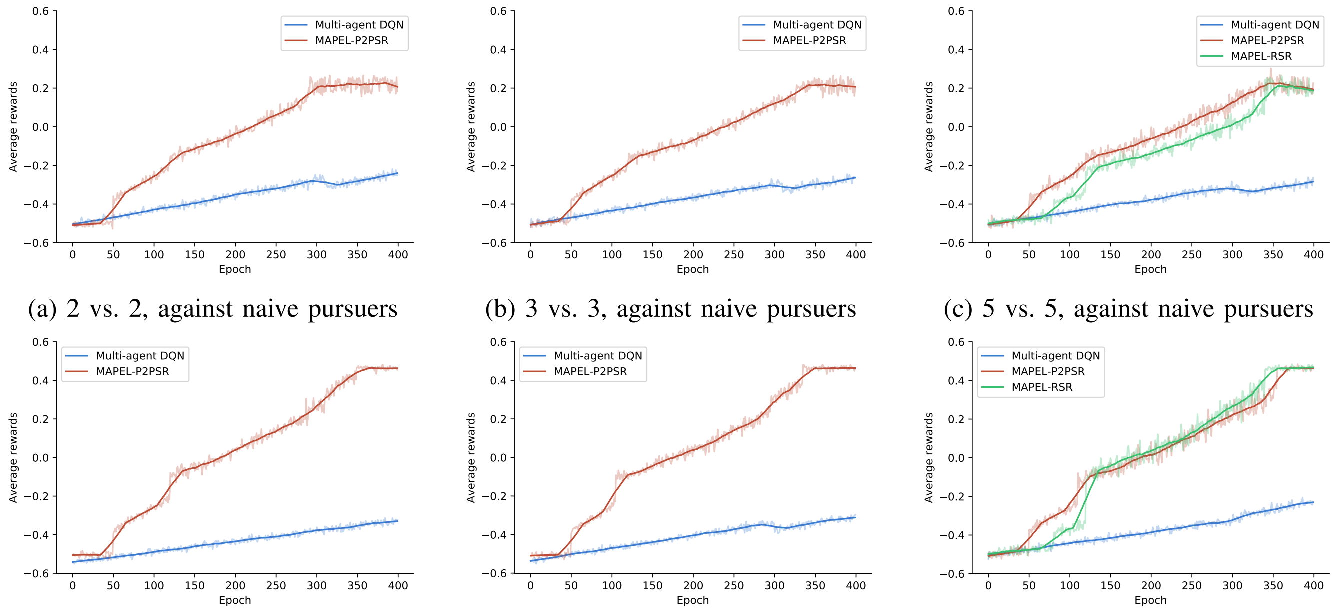
We perform our experiments on the multi-agent pursuer-evader environment presented in section III. We begin with explaining the environment representation, agent observation featurization, and representation of messages under different situation report methods. All the experiments have been conducted on a workstation with 1.2 GHz CPU, 256 GB RAM, NVIDIA V100 GPU and running Ubuntu 18.04. We use PyTorch [**?**] for network implementation.

A. Environment

The environment is a grid world composed of multiple grids of size 32×32. Figure 4 shows partial observation of agent in a 2 vs 2 game. The white regions are the empty regions where evaders and pursuers can move. An agent considers all grid cells outside its observation space to be empty. Evaders are blue, pursuers are green, and the target is red. Figure 4 (a) shows the observation space of evader 1, it can see all the grid cells in its observation space, it knows the locations of other evaders and the target. Similarly figure 4 (b) shows the observation space of evader 2. Figure 4 (c) shows the observation space of pursuer 1, it can see all the grid cells in its observation space, it knows the locations of other pursuers and the target. Similarly, the observation space of pursuer 2 is represented in figure 4 (d).

B. Agent Observation

For Q-learning, we need to represent an agent's observation as meaningful features. In our experiments, we found that raw RGB frames provide good observational for 2 vs 2 games but fail to generalize for more number of agents. We represent each type of entity in our environment as separate channels. We have five channels in our feature space, each of size 32×32. In the bottom left portion of figure 2, we show the featurization of observation of one of the pursuers in a 4 vs 4 game. The first channel shows the observation space of the agent, the second channel shows the position of the agent itself, the third channel shows the position of other agents, the fourth one shows the location of the target, and the fifth channel shows the location of the opponent(s) observed. This feature representation accurately incorporates all the information observed by an agent.



(a) 2 vs. 2, against naive pursuers (b) 3 vs. 3, against naive pursuers (c) 5 vs. 5, against naive pursuers

(d) 2 vs. 2, against naive evaders (e) 3 vs. 3, against naive evaders (f) 5 vs. 5, against naive evaders

Fig. 5: Learning curve comparison of different methods in different scenarios.

C. *Message Representation*

In P2PSR, an agent a_i receives a situation report m_i^t in form of a vector of size $\mathcal{N}-1$ at time t. The $\mathcal{N}-1$ elements of the vector represent the messages from other agents, if the value of an element in the vector is 1, then it means the corresponding agent has seen the target or opponent(s) in its observation space. An element with value 0 means that the observation space is empty. In RSR, an agent a_i receives a situation report message m_i^t in form of a vector with size 2 from two of its adjacent agents at time t.

D. *Training*

We train multi-agent DQN for learning evaders against naive pursuers and multi-agent DQN for learning pursuers against naive evaders. We train both MAPEL cooperation methods against naive agents. All our models are trained for 400 epochs, 500 episodes per epoch. We use Adam [**?**] optimizer to train all our models. Learning rate is varied over epochs, it starts with 0.001 and decays at every 200 epochs by one-tenth. To ensure exploration, ϵ-greedy starts at 1.0 and ends at 0.1. A discount factor of 0.99 is used. While training multi-agent DQN models, a history length of 5 observations is used. We use a batch size of 64 in all our experiments. We vary the number of agents per team from 2 to 5 for all our models. Following are the model variations that we train,

1) MA-DQN pursuers against naive evaders.
2) MA-DQN evaders against naive pursuers.
3) MAPEL-P2PSR pursuers against naive evaders.
4) MAPEL-P2PSR evaders against naive pursuers.
5) MAPEL-RSR pursuers against naive evaders.
6) MAPEL-RSR evaders against naive pursuers.

E. *Evaluation*

We evaluate our MA-DQN, MAPEL-P2PSR, and MAPEL-RSR evaders against naive pursuers and vice-versa. 100,000 episodes are used for all the evaluations. Average reward is reported for both evaders and pursuers. In the case of pursuers running different methods, we also report the total number of times pursuers were able to capture all the evaders. We call these results as "complete wins".

VI. Results

Figure 5 compares different models' learning curve under the different number of agents for both evaders and pursuers. For the number of agents $\mathcal{N} = 1, 2, 3$, both MAPEL cooperation methods, i.e., P2PSR and RSPRP have the same message length. In such cases, there is no fundamental difference between these methods. Therefore, we only train both methods when team sizes are more than 3. Figure 5 (a) shows the learning curve for evaders with MA-DQN and MAPEL-P2PSR when the number of agents is 2 for both evaders and pursuers. Similarly figure 5 (b) is for a 3 vs. 3 scenario for evaders against naive pursuers. Figure 5 (d) and (e) are for pursuers with MA-DQN and MAPEL-P2PSR against naive evaders when the numbers of agents are 2 and 3 respectively. Figure 5 (c) and (f) show all three methods for evaders and pursuers against their naive opponents when the number of agents is 5.

In all the scenarios, pursuers are able to score better rewards than evaders. We believe that the pursuers are able to learn about the strategy where capturing all the evaders maximizes their rewards. From figure 5 (c) and (f), it is evident that MAPEL-P2PSR for pursuers learns about capturing all evaders quickly as compared to MAPEL-RSR. After 350 epochs both the methods converge to same average rewards which

	Naive		MA-DQN		MAPEL-P2PSR		MAPEL-RSR	
Scenario	Average reward	Complete wins	Average reward	Complete wins	Average reward	Complete wins	Average reward	Complete wins
2 vs. 2	0.159	9.77%	-0.274	3.13%	0.431	14.62%	NA	NA
3 vs. 3	0.161	10.23%	-0.235	3.17%	0.396	15.79%	NA	NA
4 vs. 4	0.162	10.07%	-0.217	2.92%	0.479	16.71%	0.456	15.92%
5 vs. 5	0.165	10.13%	-0.213	2.72%	0.483	16.23%	0.468	15.92%

TABLE I: Evaluation result of different methods for pursuers against naive evaders in different scenarios.

Scenario	Naive	MA-DQN	MAPEL-P2PSR	MAPEL-RSR
2 vs. 2	0.134	-0.279	0.419	NA
3 vs. 3	0.153	-0.247	0.429	NA
4 vs. 4	0.157	-0.225	0.423	0.416
5 vs. 5	0.161	-0.217	0.417	0.419

TABLE II: Evaluation result of different methods for evaders against naive pursuers in different scenarios.

shows that MAPEL-RSR has similar learning capabilities as MAPEL-P2PSR. We believe this is due to the fact that in the case of MAPEL-P2PSR, all pursuers know about all other pursuers' observations explicitly which helps them in knowing about "capture all evaders" strategy early. In the case of MAPEL-RSR, more epochs are required to learn about this strategy.

Table I compares the average rewards and complete wins of different methods for pursuers against naive evaders in four scenarios, i.e., 2 vs. 2, 3 vs. 3, 4 vs. 4, and 5 vs. 5. It can be seen that the naive method performs better than MA-DQN in all the scenarios. On rendering a few episodes, we find that MA-DQN pursuers are not able to find the shortest paths as compared to the naive method. For some of the successful episodes, we find that pursuers are able to beat the opponents when some of the team members are closer to the target as compared to the evaders. In 4 vs 4 and 5 vs 5 scenarios, we can see that MAPEL-P2PSR is ahead of MAPEL-RSR by 0.023 and 0.025 units of average reward respectively. This is in line with our earlier hypothesis that MAPEL-P2PSR is better at learning about "capture all evaders" strategy because of dense communication. This is evident from the "complete wins" in column 7 and 9. The difference in the average reward is still less when compared to difference in "complete wins" between the two methods.

Table II compares the average rewards and complete wins of different methods for evaders against naive pursuers in four scenarios, i.e., 2 vs. 2, 3 vs. 3, 4 vs. 4, and 5 vs. 5. Similar to the case of evaders, the naive method performs better than MA-DQN in all the scenarios. We also observe that the rewards from MAPEL-P2PSR and MAPEL-RSR for evader are smaller than the pursuers. The reason for this is that pursuers can learn about "capture all evaders" strategy to get more reward whereas pursuers don't have any such strategy to maximize their rewards further. The reason MAPEL methods perform better than naive and MA-DQN methods is that evaders can avoid the regions where pursuers have been observed by some members of the team.

VII. Conclusions and Future Work

In this paper, we presented a variation of multi-agent pursuit-evasion game with partial observability. We also present MAPEL for multi-agent cooperative reinforcement learning to solve the game. We compare proposed MAPEL with two benchmarks; the naive method which is a greedy solution and a multi-agent DQN formulation. We perform experiments with varying number of agents to show the generalizability of the MAPEL cooperation methods. We empirically show that MAPEL cooperation methods are better at learning cooperation strategy by reporting the results of "capture all evaders" in the case of pursuers.

In the future, our goal would be to test the transfer-ability of MAPEL methods to games with more number of agents. We would also like to experiment under different game conditions like opponents with different speeds, non-equal team sizes, moving target, etc. We would also like to find effective ways of analyzing and comparing proposed cooperation methods.

References

[1] J. Y. Kuo, H.-F. Yu, K. F.-R. Liu, and F.-W. Lee, "Multiagent cooperative learning strategies for pursuit-evasion games," *Mathematical Problesm in Engineering*, 2015.

[2] H. He, J. Boyd-Graber, K. Kwok, and H. D. III, "Opponent modeling in deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, vol. 48, pp. 1804–1813.

[3] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the 11th International Conference on International Conference on Machine Learning*, 1994, pp. 157–163.

[4] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to navigate in complex environments," *CoRR*, vol. abs/1611.03673, 2016.

[5] A. Antoniades, H. J. Kim, and S. Sastry, "Pursuit-evasion strategies for teams of multiple agents with incomplete information," pp. 756–761, 2003.

[6] J. P. Hespanha, G. J. Pappas, and M. Prandini, "Greedy control for hybrid pursuit-evasion games," 2001.

[7] L. J. Guibas, J.-C. Latombe, S. M. Lavalle, D. Lin, and R. Motwani, "A visibility-based pursuit-evasion problem," *International Journal of Computational Geometry & Applications*, vol. 09, pp. 471–493, 1999.

[8] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 662–669, 2002.

[9] L. E. Parker, "Distributed algorithms for multi-robot observation of multiple moving targets," *Autonomous Robots*, vol. 12, no. 3, pp. 231–255, May 2002.

[10] C. H. Yong and R. Miikkulainen, "Coevolution of role-based cooperation in multiagent systems," *IEEE Trans. on Auton. Ment. Dev.*, vol. 1, no. 3, pp. 170–186, Oct. 2009.

[11] A. T. Bilgin and E. Kadioglu-Urtis, "An approach to multi-agent pursuit evasion games using reinforcement learning," in *International Conference on Advanced Robotics*, 2015, pp. 164–169.

[12] Q. Zhang, D. Zhao, and F. L. Lewis, "Model-free reinforcement learning for fully cooperative multi-agent graphical games," in *2018 International Joint Conference on Neural Networks*, 2018, pp. 1–6.

[13] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Neural Information Processing Systems*, 2017.

[14] Z.-W. Hong, S.-Y. Su, T.-Y. Shann, Y.-H. Chang, and C.-Y. Lee, "A deep policy inference q-network for multi-agent systems," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 1388–1396.

[15] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," in *Proceedings of the 16th International Conference on Autonomous Agents and MultiAgent Systems*, 2017, pp. 464–473.

[16] 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.

[17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354, 2017.

[18] J. Liu, S. Liu, H. Wu, and Y. Zhang, "A pursuit-evasion algorithm based on hierarchical reinforcement learning," in *International Conference on Measuring Technology and Mechatronics Automation*, 2009, vol. 2, pp. 482–486.

[19] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews*, 2008.

[20] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[21] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[22] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," 2017.

[23] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems*, 2017, pp. 66–83.

[24] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 2085–2087.

[25] J. N. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch, "Learning with opponent-learning awareness," in *Proceedings of the 17th Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 122–130.

[26] G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani, "Lenient multi-agent deep reinforcement learning," in *Proceedings of the 17th Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 443–451.

[27] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," 2018.

[28] C. H. Yong and R. Miikkulainen, "Cooperative coevolution of multi-agent systems," Tech. Rep., 2001.

[29] F. L. Pinheiro and F. P. Santos, "Local wealth redistribution promotes co-operation in multiagent systems," in *Proceedings of the 17th Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 786–794.

[30] K. Shao, Y. Zhu, and D. Zhao, "Cooperative reinforcement learning for multiple units combat in starcraft," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, pp. 1–6.

[31] Z. Hu and D. Zhao, "Reinforcement learning for multi-agent patrol policy," in *9th IEEE International Conference on Cognitive Informatics*, 2010, pp. 530–535.

[32] D. K. Marzagão, N. Rivera, C. Cooper, P. McBurney, and K. Steinhöfel, "Multi-agent flag coordination games," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 2017, pp. 1442–1450.

[33] B. D. Bryant and R. Miikkulainen, *A Neuroevolutionary Approach to Adaptive Multi-agent Teams*, pp. 87–115, 2018.

[34] C. Guestrin, M. G. Lagoudakis, and R. Parr, "Coordinated reinforcement learning," in *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002, pp. 227–234.

[35] Z. Zhang and D. Zhao, "Clique-based cooperative multiagent reinforcement learning using factor graphs," *Journal of Automatica Sinica*, vol. 1, no. 3, pp. 248–256, 2014.

[36] Z. Zhang and D. Zhao, "Cooperative multiagent reinforcement learning using factor graphs," in *4th International Conference on Intelligent Control and Information Processing*, 2013, pp. 797–802.

[37] C. Amato and F. A. Oliehoek, "Scalable planning and learning for multiagent pomdps," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015, pp. 1995–2002.

[38] R. Isaacs, *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*, 1999.

[39] R. Liu et al., "A novel approach based on evolutionary game theoretic model for multi-player pursuit evasion," in *International Conference on Computer, Mechatronics, Control and Electronic Engineering*, 2010, vol. 1, pp. 107–110.

[40] H. Wang, Q. Yue, and J. Liu, "Research on pursuit-evasion games with multiple heterogeneous pursuers and a high speed evader," in *27th Chinese Control and Decision Conference*, 2015, pp. 4366–4370.

[41] A. Alexopoulos, T. Schmidt, and E. Badreddin, "Cooperative pursue in pursuit-evasion games with unmanned aerial vehicles," in *International Conference on Intelligent Robots and Systems*, 2015, pp. 4538–4543.

[42] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *Neural Information Processing Systems Workshop*, 2017.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.