A Memetic Multi-Agent Demonstration Learning Approach with Behavior Prediction

Yaqing Hou
Interdisciplinary Graduate School
Nanyang Technological University, Singapore 639748
houy0003@e.ntu.edu.sg

Yifeng Zeng
School of Computing
Teesside University
Middlesborough, UK
y.zeng@tees.ac.uk

Yew-Soon Ong
School of Computer Engineering
Nanyang Technological University, Singapore 639748
asysong@ntu.edu.sg

ABSTRACT

Memetic Multi-Agent System (MeMAS) emerges as an enhanced version of multi-agent systems with the implementation of meme-inspired agents. Previous research of MeMAS has developed a computational framework in which a series of memetic operations have been designed for implementing multiple interacting agents. This paper further endeavors to address the specific challenges that arise in more complex multi-agent settings where agents share a common setting with other agents who have different and even competitive objectives. Particularly, we propose a memetic multi-agent demonstration learning approach (MeMAS-P) with improvement over existing work to allow agents to improve their performance by building candidate models and accordingly predicting behaviors of their opponents. Experiments based on an adapted minefield navigation task have shown that MeMAS-P could provide agents with ability to acquire increasing level of learning capability and reduce the candidate model space by sharing meme-inspired demonstrations with respect to their representative knowledge and unique candidate models.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Experimentation

Keywords

multiple agents; demonstration learning; memetic algorithms

1. INTRODUCTION

Memetic Multi-Agent System (MeMAS) has emerged as an enhanced version of multi-agent system (MAS) wherein all meme-inspired agents acquire increasing learning capacity and intelligence through meme evolution independently or via social interaction [1][8]. Taking inspiration from the Darwin's theory of natural selections and Darkins' notion of a meme [13], MeMAS introduces a computational framework in which a series of memetic operations have been designed for modeling multiple interacting agents, whereby memes form the fundamental building blocks of agents' mind universe, and can be transmitted to others. Importantly, when compared to the state-of-the-art multi-agent learning approaches, such as Advice Exchange Model (AE) [18], the modelling of multiple agents with MeMAS exhibits significant superiority in achieving a greater level of adaptivity and effective problem-solving. MeMAS has been successfully applied into several problem domains, such as the minefield simulation task [15] and 3D interactive video games [9].

Existing MeMAS has focused on a simple mutli-agent setting where all agents tend to have same or similar behaviors, and objectives. In case that multiple agents have different objectives, the modeling of multiple agents becomes rather difficult since agents are required to learn an "efficient" strategy to interact with their opponents. Nevertheless, searching for an optimal interaction strategy is a hard problem since its effectiveness depends mostly on the strategies of their opponents involved. Thus, it is further important to extend the agents' learning ability to recognize the strategies of other agents present in the environment.

In this paper, our interest lies in addressing the specific challenges that arise in more realistic and complex multi-agent settings where agents have different and even competitive objectives. Particularly, beyond the formalism of MeMAS and taking meme as basic unit of demonstrations, we propose a memetic multi-agent demonstration learning approach (MeMAS-P) with improvement over existing work to allow subject agents to improve their performance by predicting behavior of their opponents. More specifically, we make the contributions into the following two important components in the development of MeMAS-P:

- Given the instance data that was recorded from previous interaction activities, subject agents in the MeMAS-P are designed to automatically build candidate models of their opponents and predict their behavior by solving the models;
- MeMAS-P provides the ability for subject agents to enhance their learning performance and reduce the complexity of behavior prediction by sharing meme-inspired demonstrations with respect to their internal representative knowledge while also the available candidate models.

In the present study, a popular "Temporal Difference - Fusion Architecture for Learning and Cognition" (FALCON)

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), John Thangarajah, Karl Tuyls, Stacy Marsella, Catholijn Jonker (eds.), May 9–13, 2016, Singapore.*

[23] is employed as the connectionist reinforcement learning machine of our agents due to its well-established fast and online self-learning capabilities. Further, to investigate the effectiveness of the proposed MeMAS-P, we conduct the experiment on an adapted Minefield Navigation Task (MNT) [16]. The empirically results show that MeMAS-P could effectively improve the learning performance of subject agents while reducing the behavior prediction complexity through the embedded demonstration learning techniques.

The rest of the paper is organized as the follows: Section 2 provides an overview of the related research literature. Section 3 begins with a brief discussion of reinforcement learning which is followed by an introduction of MeMAS. Further, study of the proposed MeMAS-P which is composed of the behavior prediction and multi-agent demonstration learning is presented in Section 4. Section 5 presents a comprehensive empirical study of the proposed MeMAS-P based on an adapted Minefield Navigation Task. Finally, we conclude this paper with some brief remarks in Section 6.

2. LITERATURE REVIEW

Reinforcement learning (RL) is an interaction-based paradigm wherein agents learn to adjust their actions with the goal of maximizing the reward signals from the environment. Existing approaches for developing RL agents, such as temporal difference [22], dynamic programming [6], policy search [3], *etc.*, focus on the independent learning through exploring the environment. While such techniques have had great success in individual learning and software applications, they require a large amount of data and high exploration times and have limited applications in many realistic and complex problem domains.

On the other hand, learning from demonstration (LfD) is proposed as an algorithm that provides agents with demonstrations from a supposed expert, from which agents derive suitable learning policies and hence increase their learning speed [2]. Proposed techniques span a wide range of policy learning methods, ranging from classification [11] to regression [17], and utilizing a variety of interaction methods, such as natural language [21], kinesthetic teaching [7], observation [4] and advice exchanging [18]. Recently, LfD has also been successfully introduced into RL. For example, Torrey *et al.* [26] studied how a RL agent can best instruct another agent using a limited amount of demonstrations while Brys *et al.* [5] investigated the intersection of RL and LfD and further used the expert demonstrations to speed up learning through a reward shaping process. Nevertheless, existing LfD approaches to date focus almost exclusively on the problem of a single agent being taught by a single teacher demonstrator while the multi-agent demonstration learning has less been explored.

LfD can be problematic because of the number of trials necessary to gather sufficient samples to learn correctly. The problem becomes compounded in a multi-agent setting due to the potentially much larger design space arising from the number of and interactions between the agents. Currently, the research of LfD starts to take steps toward reducing the complexity. Multiple autonomous agents request advice and provide demonstrations for each other [18], and agents with similar behaviors learn from each other by leveraging broadcasted demonstrations [24]. Further, in a recent memetic multi-agents system, demonstrations have also been defined as memes and transferred among agents via an evolutionary imitation process [15]. However, most of the aforementioned research focus on the simple mutli-agent setting where agents tend to have same or similar behaviors, and objectives.

In this paper, we contribute to the development of LfD in multi-agent RL system. Particularly, using an example inspired from minefield navigation task, we investigate the proposed MeMAS-P in addressing the complex MAS problems where agents have competitive objectives.

3. PRELIMINARIES

This section begins with a brief introduction of reinforcement learning which is followed by an introduction of Memetic Multi-Agent System (MeMAS).

3.1 Reinforcement Learning

RL is a paradigm for an agent to optimize its behaviors through reward and punishment received from environment. A standard RL problem is typically formulated as a Markov decision process [22] which is composed of $< S, A, T, R >$ tuples where S indicates the observable environmental states, A denotes the available actions, T describes the stochastic transition between state s and s', usually defined as $T(s|a, s')$; R is the reward function that determines the scalar reward of a transition.

In case of the multiple-step decision making problems, the action selection should not only depend on the current states, since the agent can only know the merit of an action beyond several steps in the future. Thus, Temporal Difference (TD) method, such as Q-learning, is proposed to estimate for a learning system to perform an action a given state s. The iterative estimation of the value function $Q(s, a)$ is usually defined by:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\delta$$

where $\alpha \in [0, 1]$ is the learning parameter, and δ is the temporal-difference error:

$$\delta = R(s, a, s') + \gamma max_{a'} Q(s', a') - Q(s, a)$$

where R is the immediate reward value of performing action a given state s, $\gamma \in [0, 1]$ is the discount parameter, and $max_{a'} Q(s', a')$ is the maximum estimated value of the next state s'.

In this work, we consider a self-organizing neural network "Temporal Difference - Fusion Architecture for Learning and Cognition" (FALCON) [23] (which is depicted in Fig. 1) as the connectionist RL machine of autonomous agents. FALCON employs a three-channel neural network architecture and comprises three input fields, namely, a sensory field F_1^{c1} for representing states, a motor field F_1^{c2} for representing actions, and a feedback field F_1^{c3} for representing reward values. It has a cognitive field F_2 for the acquisition and storage of learning knowledge, which encodes a relation among the patterns in the three input channels. Particularly, in FALCON dynamics, the baseline vigilance parameter ρ^k is an extremely important parameter since it controls the level of knowledge generation where $\rho^k \in [0, 1]$ *for* $k = 1, 2, 3$.

3.2 Memetic Multi-Agent System

In the design of MeMAS, which is depicted in Fig. 1, the core evolutionary process contains meme representation, meme expression, meme assimilation, meme internal evolution and meme external evolution. Particularly, the meme

in memetic, as with genes in genetic, is represented as the building block of cultural transmission and replication. Further, meme representation represents the stored memes which is respectively described internally as the learned knowledge in agents' mind universe (Memotype) and externally as the manifested demonstrations that can be observed by others (Sociotype). Meme expression is the process for individual to express their internal memotypes as demonstrations to others while meme assimilation is the way for an individual to capture these demonstrations and update them into their respective mind universe. Notably, the expression of such memotypes might take the form of the domain-specific knowledge that infect other agents' perceptions, minds, etc. In the present study, we consider a manifestation of neuronal memes as memory items that fill agents' mind universe.

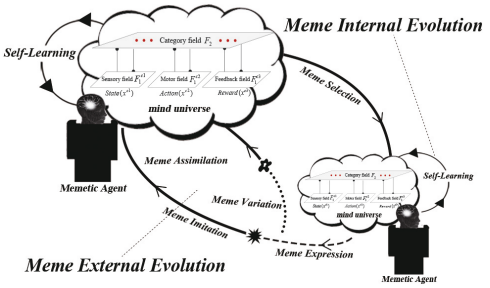


Figure 1: Illustration of the Memetic Multi-Agent System in which FALCON (inside the mind universe) is used as the learning machine for memetic agents.

Meme internal evolution and meme external evolution comprise the main behavioral learning aspects. Meme internal evolution is the process for agents to update their mind universe by self learning. Meme external evolution, on the contrary, serves to model the social interaction among agents. The implementation framework of MeMAS is outlined in Alg. 1. As indicated in the implementation, the meme evolution process is made up of a sequence of learning trials which will continue until the ultimate conditions, such as mission numbers, are satisfied. During the learning process, a population of memetic agents firstly conduct the meme internal evolution within the environment of interest (see line 5). In case that an agent identifies a teacher agent via meme selection, the meme external evolution happens to instruct how an agent learns from the others mainly via meme transmission process (see line 7-12).

MeMAS has been implemented in several applications. Particularly, in a simple minefield simulation task, Feng *et al.* [15] discuss the efficiency of MeMAS in improving the learning performance of agents by modeling the human-like social interactions. Further, Chen *et.al* [9] considers a scenario where agents have different memotypes, and MeMAS in an adapted minefield simulation task shows emergent behaviors of learning agents. Besides, a practical 3D interactive game, namely "Home Defence" has also been studied where non-player characters trained by MeMAS interact naturally with human-players online.

4. PROPOSED MEMAS-P

We propose a memetic multi-agent demonstration learning framework (MeMAS-P) where subject agents could develop effective interactive mechanisms with existing opponent agents by predicting their behaviors. The implementation of MeMAS-P is depicted in Fig. 2, wherein the behavior prediction and multi-agent demonstration learning is defined as two major components.

Algorithm 1: Pseudo Code of MeMAS

```
1  Begin:
2  Initialization: Generate the initial agents
3  while stop conditions are not satisfied do
4      for each current agent agt(c) do
5          Perform meme internal evolution process
           /*Meme external evolution*/
6          if identify teacher agent via meme selection
           then
7              Perform meme expression with agt(s)
               under the state of agt(c)
8              Perform meme variation with probability γ
9              /*γ is the frequency probability of variation
               process*/
10             Perform meme transmission for agt(c) to
               assimilate agt(s)'s action
11         /*Perform action from meme external evolution
           if identified teacher agent, else perform action
           from meme internal evolution*/
12         /*End meme internal evolution*/
13 End
```

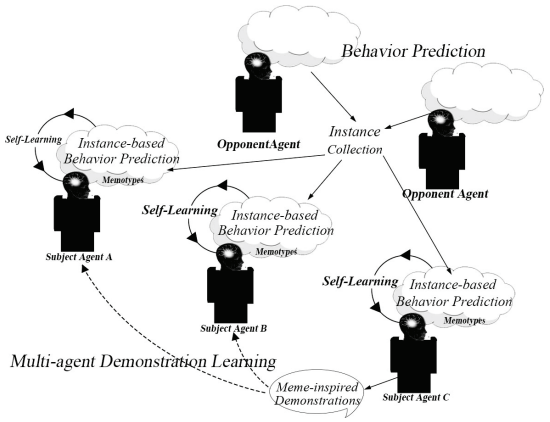


Figure 2: A Generic Illustration of the proposed MeMAS-P.

4.1 Learning Problem Specification

The learning problem of subject agents (defined by MDPs in Section 3.1) in MeMAS-P is made up of a sequence of simulations which will continue until the ultimate conditions, such as success levels, are satisfied. For each current step, subject agents select an appropriate action from a set of possible actions and learn the association from their current states, selected actions to the rewards received from the environment. Further, during the learning process, subject agents are likely to interact with one-or-more other agents, which could also influence individual rewards. Different to subject agents, the opponents that have competitive objectives follow static strategies that are unknown to subject

agents. Both subject agents and their opponents act simultaneously at each time step while their actions cannot be directly observed by others.

4.2 Behavior Prediction

In complex multi-agent systems, subject agents are likely to encounter other opponent agents and may need to interact with them to achieve their own goals. In this case, they are required to be aware of the behavior or strategies of their opponents and hence design an efficient interactive solution to handle such encounters. However, agents are usually autonomous and their strategies are private. Therefore, it is essential to extend the the learning ability of subject agents to recognize the capabilities, strategies, or models of their opponents that are presented in the environment.

In our approach, subject agents are expected to build the interactive strategy by predicting the behavior of their opponents. Particularly, the prediction is achieved by solving opponents' models that are trained from the available data instances. Comparing with predicting behavior directly from the data, the model-based behavior prediction has better scalability since the knowledge is reusable. Also, if subject agents get access to the true model of the opponent, they can predict its behavior in any environment and conduct more sensible decisions.

Since the built models (or strategies) of opponent agents are not always known with certainty, subject agents need to maintain a number of candidate models and assign them confidence of being correct. When none of the available models is likely to be exactly correct, we select the one of them that better accounts for the collected data. Further, the confidence that is assigned to candidate models being correct can be adjusted online based on how well they predict the behavior of opponent agents. That means, in case that opponent agents make changes on their strategies during the interaction, subject agents can tune the confidence levels that were assigned to candidate models accordingly.

Alg. 2 summarizes the detailed learning process of subject agents with behavior prediction of their opponents. Specifically, before the learning process, the instances of opponent agents, which are formatted as a set of <S,A,R> tuples, are firstly collected to train all candidate models $M = \{M_1, M_2, ..., M_l\}$. For example, given a <S,A,R> instance, a FALCON candidate model will set a chosen action A, a given states S and a received reward R as inputs of vector F_1^{ck}. Then, it learns to associate the state F_1^{c1}, the action F_1^{c2} and the reward F_1^{c3} and blends them into the cognitive field F_2. Notably, in the study of RL, such instance based methods have been widely leveraged in predicting RL models [19][25]. Once the training process have been completed, subject agents will be assigned with a set of candidate models and begin the online learning process (see step 2-5).

Particularly, the behavior prediction happens whenever a subject agent detects any opponent agent $agt(o)$ nearby (see line 12). More specifically, an ϵ-greedy model selection scheme is used to balance the exploration and exploitation of candidate models (see line 13-16). It selects a candidate model M_s of the highest confidence value $conf(M_l)$ with probability 1 - ϵ ($0 \leq \epsilon \leq 1$), or chooses a random model otherwise. The value of ϵ gradually decays with behavior prediction times of each subject agent. Subsequently, the subject agent will simulate the virtual operation for each available action a and obtain a virtual state $s(agt(c))'$ (see

Algorithm 2: Self Learning with Behavior Prediction

Input: Candidate models $M = \{M_1, M_2, ...M_l\}$, Collected data $D = \{D_1, D_2, ...D_k\}$, where $D_k =< S_k, A_k, R_k >$, Current state s, Action set $A = [a_1, a_2, ..., a_i]$

1 **Begin**:
2 **for** *all* $M_l \in M$ **do**
3 **for** *all* $D_k \in D$ **do**
4 **Set** input vector $< S_k, A_k, R_k >$ in M_l
5 **Training** with vector $< S_k, A_k, R_k >$
6 **Initialize** subject agents and their opponents
7 **Assign** each subject agent with candidate models M
8 **while** *stop conditions are not satisfied* **do**
9 **for** *each subject agent* $agt(c)$ **do**
10 **Perform** *self learning* process
11 /*Behavior Prediction*/
12 **if** *detect an opponent agent* $agt(o)$ **then**
13 **if** $Rand > scheme(\epsilon)$ **then**
14 $M_s = Random\{M_l : for\ all\ model\ M\}$
15 **else**
16 $M_s = Max\{conf(M_l) : for\ all\ modelM\}$
17 **for** *each available action* $a_i \in A$ **do**
18 $s(agt(c))' = Virtual\_Perform(agt(c), a_i)$
19 $a_o = Predict(M_s, s(agt(c))')$
20 $s(agt(o))' = Virtual\_Perform(agt(o), a_o)$
21 **if** *fail interaction after virtual move* **then**
22 **Discard** a_i from action set A
23 **Get** a_s from A with highest reward of self learning in $agt(c)$
24 **Continue** *self learning* process
25 **Update** $conf(M_s)$
26 **End**

line 18). The received state $s(agt(c))'$ is further used to infer the corresponding state of $agt(o)$, and thus predicts its action a_o by activating the internal knowledge of model M_s (see line 19). Upon receiving the results after virtual performing a_o, subject agents will decide whether to discard an action from the available action set A (see line 21-22). Then, agent $agt(c)$ will select an action a_s from the set A with the highest reward yield. After each step of all agents, the M_s leading to more success interactions is updated with greater confidence $conf(M_s)$ of being chosen for subsequent behavior prediction (see line 25):

$$\begin{cases} conf(M_s) = conf(M_s) + reward, & \text{if succeeds} \\ conf(M_s) = conf(M_s) - penalty, & \text{if fails} \end{cases}$$

where reward and penalty are positive integers and $conf(M_l) \in [-100, 100]$.

4.3 Multi-Agent Demonstration Learning

Intuitively, the predicted candidate model derived from RL data could reveal the true behavior of opponent agents if subject agents are aware of the true model of their opponents. However, to achieve such results could be extremely time-consuming, and even impossible due to the large space of candidate models ascribed to opponent agents. In the

past few years, most researchers have concentrated on reducing candidate model complexity of other agents. Among them, some work focuses on reducing the solution complexity by compressing the model space of other agents [14]. The second category, instead of constructing and solving various models, infers the model of their opponent directly from available behavior data [28][12][10]. Nevertheless, most of existing works to date focus on the learning of models for single-agent decision making process and does not integrate the behavior prediction method in a multi-agent setting.

Different to existing approaches, we propose a multi-agent demonstration learning approach for enhancing subject agents' learning capability while also reducing their candidate model complexity. In particular, the demonstration learning between agents with unique learning capabilities happens by means of imitation. In present multi-agent settings, all subject agents learn in the same environment and have the same acting abilities. Therefore, the demonstration learning could offer the advantage for student agents to behave at approximately the same performance level as their selected teachers. Further, when multiple candidate models are available, subject agents with the demonstration learning approach do not need to train all the models and they can acquire increasing level of predictive capability from their partners by sharing demonstrations with respect to their unique candidate models. The demonstration learning approach thus reduces candidate model complexity by assigning a small set of models to subject agents.

The multi-agent demonstration learning process in MeMAS-P is summarized in Alg. 3. More specifically, the subject agent will be assigned with a set of candidate models $M^{agt} = \{M_1^{agt}, ..., M_n^{agt}\}$, where agt is the index of the subject agent and n is the number of models in M^{agt} (see line 2). Further, the current agent $agt(c)$ will check if there exists a teacher agent $agt(t)$ that has higher fitness value $Fit(agt(t))$ (see line 4). Once a selection is made, the subject agent $agt(c)$ passes its state to the selected teacher agent $agt(t)$ (see line 6). Meanwhile, if agent $agt(c)$ detects an opponent agent $agt(o)$ nearby, it will also pass the inferred state $s(agt(o))$ of the detected opponent to the teacher agent $agt(t)$ (see line 7-8). By simulating the states of current agent $s(agt(c))$ and the detected opponents agent $s(agt(o))$, the teacher agent $agt(t)$ first conducts behavior prediction according to its own models M^t and discards failure actions from the available action set A (see line 9-10). It further predicts the action a_t from the action set A by activating its internal learned knowledge and transmits it to agent $agt(c)$. Then, if transmitted action a_t is available, agent $agt(c)$ will imitate and assimilate the action a_t into its mind universe by means of self learning (see line 13). In addition, a probability γ is defined in line 12 to give the intrinsic innovation tendency of selected actions in the multi-agent demonstration learning process. This variation process serves to prevent the agent $agt(c)$ from learning from teacher agents blindly.

4.4 Complexity Analysis

As aforementioned, predicting accurate behaviors of opponent agents is computationally intractable due to the complexity of candidate models and thus ways of mitigating the computational intractability are critically needed. Since the complexity is predominantly due to the space of candidate models, we therefore focus on reducing the model space of subject agents while avoiding a significant loss in optimality.

Algorithm 3: Multi-Agent Demonstration Learning

Input: Candidate models $\{M^1, M^2, ...M^{agt}\}$, where $M^{agt} = \{M_1^{agt}, ..., M_n^{agt}\}$, Current state s, Action set $A = [a_1, a_2, ..., a_i]$

1 **Begin**:
2 **Assign** each subject agent with candidate model M^{agt}
3 **for** *each subject agent* $agt(c)$ **do**
4 **if** *identify* $\{agt(t)|Fit(agt(c) < Fit(agt(t))\}$ **then**
5 **Get** $s(agt(c))$ of current $agt(c)$
6 **Pass** $s(agt(c))$ to the teacher agent $agt(t)$
7 **if** *detect an opponent agent* $agt(o)$ **then**
8 **Pass** $s(agt(o))$ to agent $agt(t)$
9 **Perform** *Behavior Prediction* with M^t
10 **Discard** failure actions from action set A
11 **Get** a_t from A with highest reward of self learning in $agt(t)$
12 **Perform** variation on a_t with probability γ
13 **Get** $a_s = a_t$ from $agt(t)$, otherwise get a_s from A with highest reward of self learning in $agt(c)$
14 The left steps are same as that in self learning process (see Alg. 2)
15 **End**

Specifically, in our framework, the entire candidate models M are partitioned into different partial sets according to the number of subject agents: $M = \{M_1^1, M_2^1, M_3^2, ...M_l^{agt}\}$ where M^{agt} aggregates the candidate model set labelled by the same subject agent mark. Then, subject agents will only train candidate models from their corresponding partial sets. According to the proposed demonstration learning process, these non-familial peer subject agents with unique candidate models M^{agt} could communicate socially and takes advantage from each other (see Alg. 3). Hence, the proposed MeMAS-P with demonstration learning can reduce the candidate model space of each subject agent in the multi-agent system.

5. EMPIRICAL STUDY

Minefield navigation task (MNT) has been known as one of the most classical and widely used minefield simulation task where all agents have the same objective to complete the navigating task successfully. Taking the classical MNT as an example, we validate the effectiveness of MeMAS-P in an adapted setting where agents may have similar or conflicting objectives.

5.1 Minefield Navigation Task

In a classic minefield navigation task, all green tanks (subject agent, denoted by *Navigator*) have the same objective to navigate across a minefield and arrive at the target (red flag) within given limited time frame. The tank shall avoid any mines or other agents. In our novel competitive scenario (depicted in Fig. 3), we further import a new kind of red tanks (denoted as *Predator*) with the objective to tracking down the navigators before they flees to the red flag. Notably, the red tanks cannot be destroyed by the mines.

Both navigators and predators that spawn randomly within the field at the beginning of each trial run are equipped with sonar sensors so that they have access to a set of detections, including mine detection, *Navigator* detection, *Preda-*

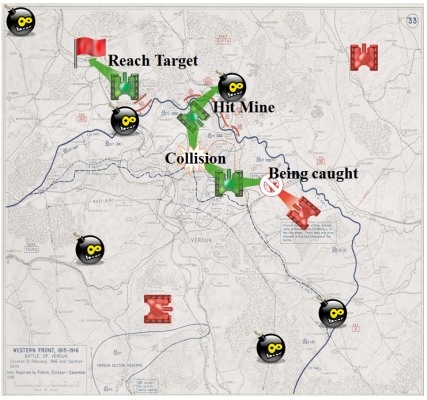


Figure 3: A screenshot of the adapted minefield navigation task with the illustration of Mission Endings.

Table 1: Summary of the parameter setting in the proposed MeMAS-P.

Falcon Parameters	
Choice Parameters $(\alpha^{c1}, \alpha^{c2}, \alpha^{c3})$	$(0.1, 0.1, 0.1)$
Learning Rates $(\beta^{c1}, \beta^{c2}, \beta^{c3})$	$(1.0, 1.0, 1.0)$
Contribution Parameters $(\gamma^{c1}, \gamma^{c2}, \gamma^{c3})$	$(0.5, 0.5, 0)$
Baseline Vigilance Parameters $(\rho^{c1}, \rho^{c2}, \rho^{c3})$	$(0.2, 0.2, 0.5)$
Temporal Difference Learning Paramters	
TD learning rate α	0.5
Discount Factor γ	0.1
Initial Q-value	0.5
ϵ-greedy Model Selection Parameters	
Initial ϵ value	1
ϵ decay rate	0.0005
Demonstration Variation Parameter	
Frequency of Variation γ	0.1

tor detection and target bearing. According to these detected states, each autonomous agent then performs one of the five possible actions at each step with a 180° forward view: $A = \{LEFT, LEFT\ FRONT, FRONT, RIGHT\ FRONT, RIGHT\}$. Particularly, a navigator is rewarded with a positive value 1 if it arrives at the target while a predator receives a reward 1 only if it catches one of the navigators before they reach the red flag. Otherwise, if an agent hits a mine, collides with others, be caught, or does not reach the target within the given limited time steps, it will be assigned with 0 reward. In our experimental study, we have a total of 6 autonomous robots, 6 mines and 1 target flag which are randomly generated over trials in a 12×12 field. A trial completes only if all navigators reach the target, being caught by the predators, hit a mine or other navigators, or exceed 30 time steps.

5.2 Experimental Settings

In our study, we conduct the experiments for 30 independent simulations with each of which has a total of 3000 missions. The parameter settings of FALCON and the proposed MeMAS-P configured in the present experimental study are summarized in Table 1. These configurations of the learning machines are considered to be consistent with previous studies [23] for the purpose of fair comparisons. The results with respect to the following metrics are then reported:

- MN: the number of training missions;
- SR: the average success rate of the agents on completing the missions.

5.3 Brief Study of MeMAS

The objective of the present experimental study is to investigate the performance of MeMAS with all 6 FALCON navigators on completing the classic Minefield Navigation Tasks. For the purpose of comparison, two state-of-the-art multi-agent demonstration learning approaches, namely Advice Exchange Model (AE) [18] and Parallel Transfer Learning (PTL) [24] are also implemented. Specifically, agents in AE with poor performance learn from their elitist by seeking their advice while agents in PTL learn from others by taking advantage from their broadcasted knowledge. Fig. 4 depicts the learning trends of navigators under these three demonstration learning approaches and the conventional MAS where agents cannot learn from others. As can be observed, all demonstration learning approaches outperform the conventional MAS. Particularly, MeMAS is noted to achieve superior performance in terms of success rate throughout the learning process when compared to the existing state-of-the-art multi-agent demonstration learning approaches.

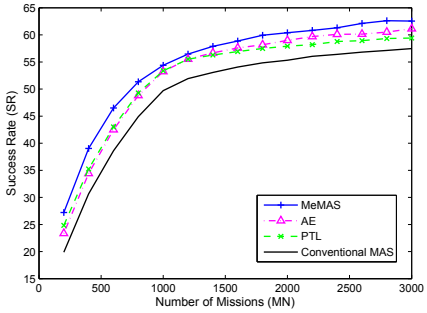


Figure 4: Success rates of navigators under MeMAS, PTL, AE and conventional MAS on completing the missions in classic Minefield Navigation Tasks.

5.4 Study of the MeMAS-P with Actual Model of Predators

We further consider a more complex MNT involving a mixture of 3 navigators and 3 predators where both navigators and predators employ FALCON with $\rho^k = (0.2, 0.2, 0.5)$ as their RL learning machines. Notably, the predator has been well trained before the online learning process of navigators. In this set of experiment, we assume all 3 navigators are aware of the actual trained FALCON model ascribed to predators and thus have the ability to predict the exact behaviors of their opponents.

As aforementioned, the proposed MeMAS-P is composed of two major components, namely behavior prediction and multi-agent demonstration learning. To study the effectiveness of each component, we investigate the average performance of navigators with respect to the following different learning abilities for comparison consideration:

- Navigator-BP: navigators with behavior prediction;
- Navigator-DL: navigators with mutli-agent demonstration learning;
- Navigator-BD: navigators with both behavior prediction and mutli-agent demonstration learning;
- Navigator-Conv: navigators under the conventional MAS that has no behavior prediction and mutli-agent demonstration learning.

Fig. 5 shows the learning trends of different kinds of navigators in terms of success rates. Notably, the success rate of Navigator-Conv in this novel MNT scenario is much lower than that under the classic MNT in the previous subsection. The results indicate that the navigators suffer from the interaction with predators in the novel MNT and have a high probability being caught by their opponents. On the other hand, Navigator-BP achieves significantly higher performance in terms of success rate than Navigator-Conv. This demonstrates that the behavior prediction process in MeMAS-P can effectively help navigators survive from their encounters with predators.

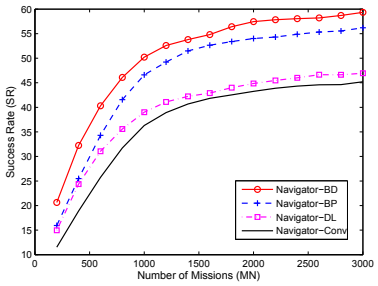


Figure 5: Success rates of navigators with and without MeMAS-P components on completing the missions in the novel Minefield Navigation Tasks.

Further, Navigator-DL and Navigator-BD also report superiority in attaining higher success rates than navigators that have no demonstration learning process. This is attributed to the multi-agent demonstration learning process which endows navigators with capacities to benefit from the demonstrations sharing from the better performing agents, thus accelerating the learning rate of the navigators in solving the missions.

5.5 Study of MeMAS-P with candidate FALCON models

In this experiment, we consider a more common scenario where the exact trained model of the predator is not accessible for navigators. Instead, we assign navigators with a number of candidate models which will be trained using instance data of predators that are recorded from previous interaction activities. According to these trained candidate models, navigators could further predict actions of predators, and use these predictions in place of predators' behaviors.

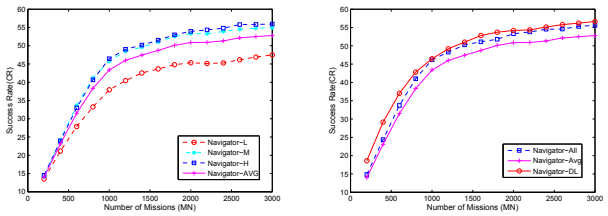
To facilitate discussion, we consider several FALCON models as candidates that assign different values to the baseline vigilance parameter ρ^k. Table 2 shows the details of parameter settings and the complete results pertaining to the success rate of navigators with different candidate models. Particularly, we classify all candidate models into three sets according to their ρ^k values, known as *1)* a low parameter set $\rho^k \in [0, 0.1]$, *2)* a medium parameter set $\rho^k \in [0.1, 0.5]$ and *3)* a high parameter set $\rho^k \in [0.5, 1]$, respectively. The average performance of navigators with respect to the following settings is then reported:

- Navigator-S: navigators that are assigned with a single candidate model;
- Navigator-All: navigators that are assigned with all candidate models;
- Navigator-L,M,H: navigators that are assigned with candidate models of the low, medium and high parameter sets, respectively;
- Navigator-AVG: average performance of Navigator-L, Navigator-M and Navigator-H;
- Navigator-DL: 3 navigators are assigned with candidate models of low, medium, high parameter set, respectively. Each can learn from their partners according to multi-agent demonstration learning process;

Table 2: Performance of navigators with different candidate models.

#	FALCON Candidate Model: Model Set	FALCON Candidate Model: (ρ^1, ρ^2, ρ^3)	Navigator Performance: Success Rate		
1	Low	(0.05,0.05,0.05)	48.97	48.67	
2		(0.05,0.05,0.1)	49.82		
3	Medium	(0.2,0.2,0.5)	54.97	54.94	55.59
4		(0.5,0.5,0.5)	54.74		
5	High	(0.5,0.5,0.9)	55.52	55.94	
6		(0.9,0.9,0.9)	55.92		

We first investigate the performance of Navigator-S in terms of success rate. Notably, Navigator-Ss with different FALCON candidate models of different baseline vigilance parameter ρ^k report distinct success rates after the learning process (see Table 2, column 4). Particularly, navigators with FALCON candidate model of $\rho^k = (0.9, 0.9, 0.9)$ obtain significant higher success rate than that of $\rho^k = (0.05, 0.05, 0.05)$. Such difference among various FALCON candidate models highlights the importance of selecting a correct candidate model for navigators to predict the behavior of predators. On the other hand, Navigator-All significantly outperforms most of Navigator-Ss. At the end of the learning process, it attains a success rate of 55.59% (see Table 2, column 6), only a bit lower than Navigator-S with the candidate model of highest ρ^k (see Table 2, row 6, column 4). The outstanding performance of Navigator-All verifies the effectiveness of the proposed behavior prediction process in making choice of the most reliable candidate models.



(a) Navigator-L,M,H&AVG. (b) Navigator-All,AVG&DL.

Figure 6: Success rates of navigators using all and part of candidate FALCON models on completing the missions in the novel Minefield Navigation Tasks.

Further, we investigate the performance of navigators with the proposed multi-agent demonstration learning. For comparison consideration, we first study the learning performance of Navigator-L, M and H, respectively, and report their average success rate by Navigator-AVG in Fig. 6(a). The success rates obtained by Navigator-All, Navigator-AVG and Navigator-DL are further depicted in Fig. 6(b). Compared to Navigator-AVG, both Navigator-ALL and Navigator-DL have demonstrated superiority in attaining higher success rates. Particularly, Navigator-DL is noted to obtain

success rates that are similar to Navigator-All. This is attributed to the reason that navigators with the multi-agent demonstration learning approach can take advantage from their partners that have better learning capabilities.

Table 3: Computing time taken by Navigator-All and Navigator-DL for training candidate models and behavior prediction.

		Navigator-DL				Navigator-All
#	Metrics	Low ρ^k	Medium ρ^k	High ρ^k	Avg	Avg
1	MT (s)	0.47	0.69	3.91	1.69	5.07
2	PN	8119	8359	10594	9024	9856
3	PT (s)	2.21	4.43	45.60	17.41	21.73

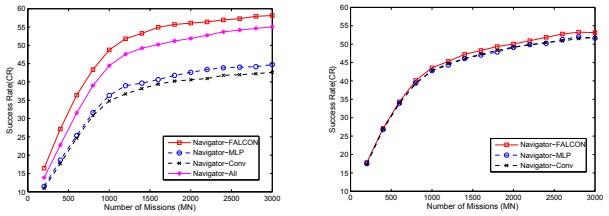
Moreover, in order to quantitatively evaluate how effective the proposed demonstration learning approach reduces the candidate model complexity of subject agents, the computational cost taken per simulation by Navigator-All and Navigator-DL for behaviors predicting is reported in Table 3, where MT is the training time of candidate models, PN is the numbers of prediction and PT is the computational time for behavior prediction. When referring to MT, Navigator-All tends to be more computationally expensive than Navigator-DL since subject agents in Navigator-All need to train all six models while that in Navigator-DL are only assigned with two models. Particularly, when we have a larger agent system with much more candidate models, demonstration learning exhibits significantly better scalability. For example, when 10 agents are in the system and 100 models are available, Navigator-DL reduces the training cost to a tenth of the required amount for Navigator-All.

On the other hand, Navigator-DL has also been observed to outperform Navigator-All by reporting lower computational cost in terms of PT (17.41s, see Table 3, row 3). This is due to the reason that, peer subject agents from Navigator-DL can take advantage form partners with unique candidate model sets. As can be observed, the subject agent with candidate model of high ρ^k tends to share its predicted demonstrations with others and hence obtains the most number of PNs (10594), much more than those obtained by subject agent with low ρ^k and medium ρ^k (8119 and 8359, see Table 3, row 2). Therefore, the proposed MeMAS-P is demonstrated to have the superiority in both reducing the candidate model spaces of subject agents, while also improving their generic learning performance.

5.6 Extensive Study of MeMAS-P with Different RL Candidate Models

Last but not least, we investigate the performance of the proposed MeMAS-P where navigators use different RL techniques as the candidate models of their opponents. In particular, we study the scenario where not only FALCON but a classical multi-layer perceptron (MLP) [27] with gradient descent based back propagation [20] are considered as the candidate models. For a fairness consideration, both the MLP and FALCON candidate models are configured to be consistent with the studies in [23] and will also be trained using available RL instances of predators recorded from previous interactive activities.

The first experiment is conducted to verify the effectiveness of FALCON and MLP as candidate models when predicting predator' behavior. In particular, three navigators in this experiment are assigned with FALCON, MLP and



(a) Navigators with demonstration learning.

(b) Navigators with no demonstration learning.

Figure 7: Success rates of navigators using FALCON and/or MLP as candidate models on completing the missions in the novel Minefield Navigation Tasks.

no candidate models, respectively, in the same environment. Fig. 7(a) depicts the learning performance in terms of success rate of navigators with respect to the following settings:

- Navigator-FALCON: the navigator using FALCON as the candidate model;
- Navigator-MLP: the navigator using MLP as the candidate model;
- Navigator-All: the navigator using both FALCON and MLP as candidate models;
- Navigator-Conv: the navigator under the conventional MAS that has no behavior prediction of predators.

As can be observed, Navigator-FALCON demonstrates its superiority in attaining much higher success rates than Navigator-MLP and Navigator-Conv. This is reasonable since the FALCON candidate model is the same as the exact learning model of predators, and hence could predict comparatively accurate behaviors.

Further, we endow all navigators with the ability to learn from their partners' valuable demonstrations. According to the result in Fig. 7(b), all of the three navigators with the proposed demonstration learning approach have reported success rates that are approximate to Navigator-All. Particularly, Navigator-MLP and Navigator-Conv show approximately 7.0% and 9.1% improvements in success rates, respectively. The improved performance demonstrates the efficacy of proposed MeMAS-P in improving the learning performance of navigators using different RL techniques as the candidate models in the adapted MNT.

6. CONCLUSIONS

In this paper, we have presented a memetic multi-agent demonstration learning framework (MeMAS-P) for modeling multiple interacting agents. The framework improves on early memetic techniques by providing an instance-based behavior prediction approach for subject agents to interact with present opponents. To reduce the candidate model complexity, we have provided a demonstration learning approach for subject agents to leverage sharing from their partners. The effectiveness of the proposed MeMAS-P has been testified in an adapted minefield navigation task. Future work could investigate incremental approaches for predicting opponents' behavior from online data.

Acknowledgments

This research is supported by the National Research Foundation Singapore under its Interactive Digital Media (IDM) Strategic Research Programme.

REFERENCES

[1] G. Acampora, J. M. Cadenas, V. Loia, and E. M. Ballester. A multi-agent memetic system for human-based knowledge selection. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(5):946–960, 2011.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[3] J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, pages 319–350, 2001.

[4] D. C. Bentivegna, C. G. Atkeson, A. UDE, and G. Cheng. Learning to act from observation and practice. *International Journal of Humanoid Robotics*, 1(04):585–611, 2004.

[5] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé. Reinforcement learning from demonstration through shaping. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[6] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press, 2010.

[7] S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 255–262. ACM, 2007.

[8] X. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan. A multi-facet survey on memetic computation. *IEEE Transactions on Evolutionary Computation*, 15(5):591–607, 2011.

[9] X. Chen, Y. Zeng, Y.-S. Ong, C. S. Ho, and Y. Xiang. A study on like-attracts-like versus elitist selection criterion for human-like social behavior of memetic mulitagent systems. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1635–1642. IEEE, 2013.

[10] Y. Chen, P. Doshi, and Y. Zeng. Iterative online planning in multiagent settings with limited model spaces and pac guarantees. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1161–1169. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[11] S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34(1):1, 2009.

[12] R. Conroy, Y. Zeng, M. Cavazza, and Y. Chen. Learning behaviors in agents systems with interactive dynamic influence diagrams. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 39–45. AAAI Press, 2015.

[13] R. Dawkins. The selfish gene. *Oxford: Oxford University Press*, 1976.

[14] P. J. Doshi. Decision making in complex multiagent contexts: A tale of two frameworks. *AI Magazine*, 33(4):82, 2012.

[15] L. Feng, Y.-S. Ong, A.-H. Tan, and X.-S. Chen. Towards human-like social multi-agents with memetic automaton. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 1092–1099. IEEE, 2011.

[16] D. Gordon and D. Subramanian. A cognitive model of learning to navigate. In *Proc. 19th Conf. of the Cognitive Science Society*, volume 25, page 271, 1997.

[17] D. H. Grollman and O. C. Jenkins. Dogged learning for robots. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2483–2488. IEEE, 2007.

[18] E. Oliveira and L. Nunes. Learning by exchanging advice. *In R.Khosla, N. Ichalkaranje, and L. Jain, editors, Design of Intelligent Multi-Agent Systems, chapter 9. Spring, New York, NY, USA*, 2004.

[19] D. Ormoneit and Ś. Sen. Kernel-based reinforcement learning. *Machine learning*, 49(2-3):161–178, 2002.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning representations by back-propagating errors*. MIT Press, Cambridge, MA, USA, 1988.

[21] P. E. Rybski, K. Yoon, J. Stolarz, and M. M. Veloso. Interactive robot task training through dialog and demonstration. In *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*, pages 49–56. IEEE, 2007.

[22] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[23] A.-H. Tan, N. Lu, and D. Xiao. Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evaluative feedback. *Neural Networks, IEEE Transactions on*, 19(2):230–244, 2008.

[24] A. Taylor, I. Dusparic, E. Galván-López, S. Clarke, and V. Cahill. Transfer learning in multi-agent systems through parallel transfer. In *Theoretically Grounded Transfer Learning at the 30th International Conference on Machine Learning (ICML). Omnipress*, 2013.

[25] M. E. Taylor, N. K. Jong, and P. Stone. Transferring instances for model-based reinforcement learning. In *Machine learning and knowledge discovery in databases*, pages 488–505. Springer, 2008.

[26] L. Torrey and M. Taylor. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1053–1060. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[27] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, U.K, May 1989.

[28] Y. Zeng, Y. Chen, and P. Doshi. Approximating behavioral equivalence of models using top-k policy paths. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1229–1230. International Foundation for Autonomous Agents and Multiagent Systems, 2011.