Self-Organizing Cognitive Agents and
Reinforcement Learning in Multi-Agent Environment

Ah-Hwee Tan   and   Dan Xiao

School of Computer Engineering, Nanyang Technological University

Nanyang Avenue, Singapore 639798

{asahtan,xiao0002}@ntu.edu.sg

Abstract

This paper presents a self-organizing cognitive archi-tecture, known as TD-FALCON, that learns to function through its interaction with the environment. TD-FALCON learns the value functions of the state-action space esti-mated through a temporal difference (TD) method. The learned value functions are then used to determine the opti-mal actions based on an action selection policy. We present a specific instance of TD-FALCON based on an e-greedy action policy and a Q-learning value estimation formula. Experiments based on a minefield navigation task and a minefield pursuit task show that TD-FALCON systems are able to adapt and function well in a multi-agent environ-ment without an explicit mechanism for collaboration.

1. Introduction

A key characteristic of Autonomous agents is the abil-ity to function and adapt by themselves in a complex and dynamic environment. As described in the reinforcement learning paradigm [1], an autonomous agent typically oper-ates in a sense, act, and learn cycle. First, the agent obtains sensory input from its environment representing the current state (**S**). Depending on the current state and its knowledge and goals, the agent selects and performs the most appro-priate action (**A**). Upon receiving the feedback in terms of rewards (**R**) from the environment, the agent learns to ad-just its behaviour in the motivation of receiving positive re-wards in the future.

This paper describes a natural extension of self-organizing neural networks known as Adaptive Res-onance Theory [2] for developing a cognitive model that learns and adapts its behaviour through the inter-action with the environment. Although various models of ART and their predictive (supervised learning) ver-sions [3, 4] have been widely applied to pattern analy-sis and recognition tasks, there have been very few attempts to use ART-based networks for reinforcement learn-ing and building autonomous systems. In contrast to pre-dictive ART that performs supervised learning through the pairing of input patterns and teaching signals, the pro-posed neural architecture, known as FALCON (Fusion Ar-chitecture for Learning, COgnition, and Navigation) [5], is capable of learning multiple mappings simultane-ously across multi-modal input patterns, involving states, actions, and rewards, in an online and incremental man-ner.

A specific class of FALCON models, called TD-FALCON, learns the value functions of the state-action space estimated through temporal difference (TD) algo-rithms. A key advantage of TD methods is that they can be used for multiple-step prediction problems, wherein the in-formation on the correctness of an action can only be avail-able after several steps into the future. We have developed various types of TD-FALCON systems using temporal dif-ference methods, including Q-learning and SARSA. The learned value functions are then used to determine the opti-mal actions based on an action selection policy. To achieve high performance, we develop a hybrid action selection pol-icy that favours exploration initially and gradually leans towards exploitation in the later stage of the learning pro-cess.

Autonomous agents seldom work in isolation. To oper-ate in a multi-agent environment, an agent faces many new challenges. First of all, the agent is affected by the actions of the other agents and in turns its action also affects the envi-ronment and the other agents. Secondly, as agents continue to learn, their behaviours change over time and the environ-ment becomes highly dynamic. An agent must therefore be able to predict the actions or to model the reasoning pro-cesses of the others in some way.

To investigate how TD-FALCON performs in a multi-agent setting, experiments have been conducted based on a minefield navigation task as well as a minefield pursuit task. The former involves a number of autonomous vehi-cles (AVs) learning to navigate through obstacles to reach

a stationary target (goal) within a specified number of steps [6]. Experimental results show that using the proposed TD-FALCON model, the AVs adapt amazingly well and learn to perform the task rapidly despite the presence and the interference of the other agents.

The second task involves multiple agents pursuing a moving target in the minefield. The task can be seen as a combination of the minefield navigation task and a typical predator/prey pursuit task [7]. Our experiments show that the agents need not have in-depth knowledge of the other agents' underlying models to accomplish the task successfully. Observing the environment and learning in an reactive manner enable the agents to predict the possible actions of other agents and thus potential outcomes of its own actions.

The rest of the paper is organized as follows. Section 2 presents the FALCON architecture and the associated learning and prediction algorithms. Section 3 presents the TD-FALCON algorithms, specifically, the action selection policy and the value function estimation mechanism. Section 4 reports our experiments and findings on the minefield navigation simulation task and the minefield pursuit task. Section 5 discusses related work. The final section concludes and highlights possible future effort.

2. FALCON ARCHITECTURE

FALCON is an extension of predictive Adaptive Resonance Theory (ART) networks. Whereas predictive ART models [3, 4] learn multi-dimensional mappings between the input and output patterns, FALCON formulates cognitive codes associating multi-modal patterns across the various input channels.
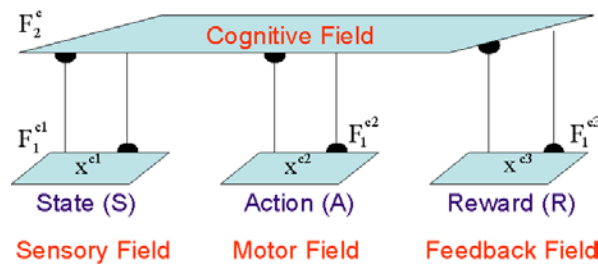

Figure 1. The FALCON architecture.

For reinforcement learning, FALCON makes use of a 3-channel architecture, consisting of a sensory field F_1^{c1} for representing the current state, an action field F_1^{c2} for representing the available actions, a reward field F_1^{c3} for repre-

senting the values of the feedback received from the environment, and a cognitive field F_2^c for encoding the relations among the values in the three input channels (Figure 1).

The network dynamics of the FALCON architecture can be used to support a myriad of cognitive operations. We describe how FALCON can be used to predict and learn value functions for reinforcement learning below.

Input vectors: Let $\mathbf{S} = (s_1, s_2, \ldots, s_n)$ denote the state vector, where s_i indicates the sensory input i. Let $\mathbf{A} = (a_1, a_2, \ldots, a_m)$ denote the action vector, where a_i indicates a possible action i. Let $\mathbf{R} = (r, \bar{r})$ denote the reward vector, where $r \in [0, 1]$ and $\bar{r} = 1 - r$. Complement coding has been found effective in ART systems in preventing the code proliferation problem.

Activity vectors: Let $\mathbf{x}^{ck}$ denote the F_1^{ck} activity vector. Let $\mathbf{y}^c$ denote the F_2^c activity vector.

Weight vectors: Let $\mathbf{w}_j^{ck}$ denote the weight vector associated with the jth node in F_2^c for learning the input patterns in F_1^{ck}. Initially, all F_2^c nodes are uncommitted and the weight vectors contain all 1's.

Parameters: The FALCON's dynamics is determined by choice parameters $\alpha^{ck} > 0$ for $k = 1$ to 3; learning rate parameters $\beta^{ck} \in [0, 1]$ for $k = 1$ to 3; contribution parameters $\gamma^{ck} \in [0, 1]$ for $k = 1$ to 3 where $\sum_{k=1}^{3} \gamma^{ck} = 1$; and vigilance parameters $\rho^{ck} \in [0, 1]$ for $k = 1$ to 3.

The dynamics of FALCON in learning and predicting, based on fuzzy ART operations [8], is described in the following sections.

2.1. Predicting Value Functions

Under the predicting mode, FALCON receives input values in one or more fields and predicts the values for the remaining fields. Upon input presentation, the input fields receiving values are initialized to their respective input vectors. Input fields not receiving values are initialized to $\mathbf{N}$, where $N_i = 1$ for all i. For predicting value functions, the state and action vectors are presented to FALCON. Therefore, $\mathbf{x}^{c1} = \mathbf{S}$, $\mathbf{x}^{c2} = \mathbf{A}$, and $\mathbf{x}^{c3} = \mathbf{N}$.

The predicting process of FALCON consists of three key steps, namely code activation, code competition, and activity readout, described as follows.

Code activation: A bottom-up propagation process first takes place in which the activities (known as choice function values) of the cognitive nodes in the F_2^c field are computed. Given the activity vectors $\mathbf{x}^{c1}, \ldots, \mathbf{x}^{c3}$, the choice function T_j^c of each F_2^c node j is computed as follows:

$$T_j^c = \sum_{k=1}^{3} \gamma^{ck} \frac{|\mathbf{x}^{ck} \wedge \mathbf{w}_j^{ck}|}{\alpha^{ck} + |\mathbf{w}_j^{ck}|}, \qquad (1)$$

where the fuzzy AND operation $\wedge$ is defined by $(\mathbf{p} \wedge \mathbf{q})_i \equiv min(p_i, q_i)$, and the norm $|.|$ is defined by $|\mathbf{p}| \equiv \sum_i p_i$ for

vectors $\mathbf{p}$ and $\mathbf{q}$. In essence, the choice function T_j computes the match of the input vectors with their corresponding weight vectors of the F_2^c node j with respect to the norm of the individual weight vectors.

Code competition: A code competition process follows under which the F_2^c node with the highest choice function value is identified. The system is said to make a choice when at most one F_2^c node can become active after code competition. The winner is indexed at J where

$$T_J^c = \max\{T_j^c : \text{for all } F_2^c \text{ node } j\}. \tag{2}$$

When a category choice is made at node J, $y_J^c = 1$; and $y_j^c = 0$ for all $j \neq J$. This indicates a winner-take-all strategy.

Activity readout: The chosen F_2^c node J performs a readout of its weight vectors to the input fields F_1^{ck} such that

$$\mathbf{x}^{ck}(\text{new}) = \mathbf{x}^{ck}(\text{old}) \wedge \mathbf{w}_J^{ck}. \tag{3}$$

The resultant F_1^{ck} activity vectors are thus the fuzzy AND of their input values and their corresponding weight vectors. Finally, the reward vector $\mathbf{R}$ associated with the input state and action vectors $\mathbf{S}$ and $\mathbf{A}$ is given by $\mathbf{R} = \mathbf{x}^{c3}$.

2.2. Learning Value Functions

For learning value functions, the state, action, and reward vectors are presented to FALCON. Therefore, $\mathbf{x}^{c1} = \mathbf{S}$, $\mathbf{x}^{c2} = \mathbf{A}$, and $\mathbf{x}^{c3} = \mathbf{R}$. Under the learning mode, FALCON performs code activation and code competition (as described in the previous section) to select a winner J based on the activity vectors $\mathbf{x}^{c1}$, $\mathbf{x}^{c2}$, and $\mathbf{x}^{c3}$. To complete the learning process, template matching and template learning are performed as described below.

Template matching: Before code J can be used for learning, a template matching process checks that the weight templates of code J are sufficiently close to their respective input patterns. Specifically, a resonance occurs if for each channel k, the *match function* m_J^{ck} of the chosen code J meets its vigilance criterion:

$$m_J^{ck} = \frac{|\mathbf{x}^{ck} \wedge \mathbf{w}_J^{ck}|}{|\mathbf{x}^{ck}|} \geq \rho^{ck}. \tag{4}$$

Whereas the choice function measures the similarity between the input and weight vectors with respect to the norm of the weight vectors, the match function computes the similarity with respect to the norm of the input vectors. The choice and match functions work cooperatively to achieve stable coding and maximize code compression.

When a resonance occurs, learning then ensues, as defined below. If any of the vigilance constraints is violated, mismatch reset occurs in which the value of the choice function T_J^c is reset to 0 for the duration of the input presentation. The search process then repeats to select another F_2^c node J until a resonance is achieved.

Template learning: Once a node J is selected for learning, for each channel k, the weight vector $\mathbf{w}_J^{ck}$ is modified by the following learning rule:

$$\mathbf{w}_J^{ck(\text{new})} = (1 - \beta^{ck})\mathbf{w}_J^{ck(\text{old})} + \beta^{ck}(\mathbf{x}^{ck} \wedge \mathbf{w}_J^{ck(\text{old})}). \tag{5}$$

The learning rule adjusts the weight values towards the fuzzy AND of their original values and the respective weight values. The rationale is to learn by encoding the common attribute values of the input vectors and the weight vectors. For an uncommitted node J, the learning rates β^{ck} are typically set to 1. For committed nodes, β^{ck} can remain as 1 for fast learning or below 1 for slow learning in a noisy environment.

3. TD-FALCON

TD-FALCON incorporates Temporal Difference (TD) methods to estimate and learn value functions, specifically, the function of state-action pairs $Q(s, a)$ that indicates the goodness for a learning system to take a certain action a in a given state s. Such value functions guide the action selection mechanism, *the policy*, to achieve a balance between exploration and exploitation, so as to maximize the reward over time. A key advantage of using TD methods is that they can be used for multi-step prediction problems, wherein the merit of an action can only be known after several steps into the future.

Following the reinforcement learning paradigm, TD-FALCON operates in a sense, act, and learn cycle as summarized in Table 1. Given the current state s and a set of available actions $\mathcal{A}$, the FALCON network is used to predict the value of performing each available action. The predicted values are then processed by an action selection strategy (also known as policy) to select an action. Upon receiving a feedback (if any) from the environment after performing the action, a TD formula is used to estimate the value of the next state. The value is then used as the teaching signal for FALCON to learn the association of the current state and the chosen action to the estimated value. It is important to note that although TD-FALCON learns to estimate the value of an action, whose consequence can only be known several steps into the future, currently we only consider learning to act one step at a time into the future.

3.1. Action Selection Policy

Action selection policy refers to the strategy used to pick an action from the set of the available actions for an agent to take in a given state. It is the policy referred to in step 3 of the TD-FALCON algorithm described in Table 1. To achieve a balance between exploration and exploitation, the e-greedy policy selects the action with the highest value with a probability of $1 - e$, where e is a constant between 0

IEEE
COMPUTER
SOCIETY

1. Initialize the FALCON network.
2. Given the current state s, for each available action a in the action set $\mathcal{A}$, predict the value of the action $Q(s, a)$ by presenting the corresponding state and action vectors $\mathbf{S}$ and $\mathbf{A}$ to FALCON.
3. Based on the value functions computed, select an action a from $\mathcal{A}$ following an action selection policy.
4. Perform the action a, observe the next state s', and receive a reward r (if any) from the environment.
5. Estimate the value function $Q(s, a)$ following a TD formula given by $\Delta Q(s, a) = \alpha \mathrm{TD}_{err}$.
6. Present the corresponding state, action, and reward (Q-value) vectors ($\mathbf{S}$, $\mathbf{A}$, and $\mathbf{R}$) to FALCON for learning.
7. Update the current state by s=s'.
8. Repeat from Step 2 until s is a terminal state.

Table 1. Generic dynamics of TD−FALCON.

and 1, and takes a random action, with probability e [9]. In other words, the policy will pick the action with the highest value with a total probability of $1 - e + \frac{e}{|A(s)|}$ and any other action with a probability of $\frac{e}{|A(s)|}$, where $A(s)$ denotes the set of the available actions in a state s and $|A(s)|$ denotes the number of the available actions.

With a fixed e value, the agent will always explore the environment with a constant level of randomness. In practice, it is beneficial to have a higher e value in the initial stage to encourage exploration of alternatives and a lower e value in the later stage to optimize the performance by exploiting familiar actions. A decayed e-greedy policy is thus proposed to gradually reduce the value of e over time. The rate of decay is typically inversely proportional to the complexity of the environment as problems with a larger input and action space will take a longer time to explore.

3.2. Value Function Estimation

One key component of the TD-FALCON (Step 5) is the iterative estimation of value functions Q(s,a) using a temporal difference equation

$$\Delta Q(s, a) = \alpha T D_{err} \qquad (6)$$

where $\alpha \in [0, 1]$ is the learning parameter and TD_{err} is a function of the current Q-value predicted by FALCON and the Q-value newly computed by the TD formula.

Using the Q-learning rule, the temporal error term is computed by

$$TD_{err} = r + \gamma \max_{a'} Q(s', a') - Q(s, a) \qquad (7)$$

where r is the immediate reward value, $\gamma \in [0, 1]$ is the discount parameter, and $\max_{a'} Q(s', a')$ denotes the maximum

estimated value of the next state s'. The update rule is applied to all states that the agent traverses. With value iteration, the value function $Q(s, a)$ is expected to converge to $r + \gamma \max_{a'} Q(s', a')$ over time.

In general, there is no restriction on the values of reward r and thus the value function $Q(s, a)$. However, in FALCON and many other pattern associators, it is commonly assumed that all input values are bounded between 0 and 1. One solution is to incorporate a scaling term into the Q-learning updating equation directly. The bounded Q-Learning rule is given by

$$\Delta Q(s, a) = \alpha T D_{err} \left(1 - Q(s, a) \right). \qquad (8)$$

With the scaling term $1 - Q(s, a)$, the adjustment of Q values will be self-scaling so that they will not be increased beyond 1. The learning rule thus provides a smooth normalization of Q values. If the reward value r is constrained between 0 and 1, we can guarantee that the Q values will be bounded between 0 and 1 as well.

4. Experiments

4.1. The Minefield Navigation Task

The minefield navigation task involves a number of autonomous vehicles (AVs) navigating through a minefield to a randomly selected target position. In each trial, the AVs start at randomly chosen positions in the field. The objective is to reach the target position in a specified time frame without hitting a mine. The target and the mines remain stationary during the trial. A trial ends when each of the AVs reaches the target (success), hits a mine (failure), or runs out of time.

Minefield navigation and mine avoidance is a non-trivial task. As the configuration of the minefield is generated randomly and changes over trials, the AVs need to learn strategies that can be carried across experiments. In addition, each AV has a rather coarse sensory capability with a 180 degree forward view based on five sonar sensors, used for sensing mines and the boundary of the minefield. For each direction i, the sonar signal is measured by $s_i = \frac{1}{1 + d_i}$, where d_i is the distance to an obstacle (that can be a mine or the boundary of the minefield) in the i direction. Other input attributes of the sensory (state) vector include the range and the bearing of the target from the current position. In each time step, individual AVs can choose one out of the five possible actions, namely move left, move diagonally left, move straight ahead, move diagonally right, and move right.

In a multi-agent (i.e. multiple AVs) environment, the minefield navigation task gets even more challenging. First, the AVs need to avoid collisions with each other as a collision will result in failures of all colliding agents. Second, as the AVs are non-stationary, an agent would need to predict
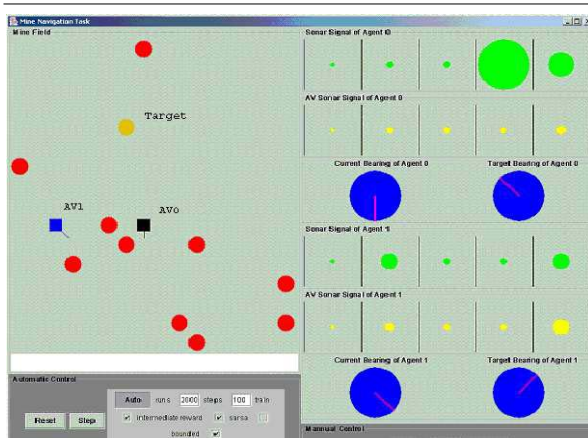
Figure 2. The multi-agent minefield navigation simulator.

Figure 3. The success rates of agent teams on the minefield navigation task.

the movement of its neighbouring AVs in order to avoid collisions. For monitoring the other AVs, a separate set of five sonar sensors is added to the agents' sensory representation. This is necessary as agents and mines have very different characteristics and should be tracked separately. In fact, our initial experiments without the additional sonar sensors produced very poor results.

The complexity for learning the multi-agent minefield navigation problem is determined by the dimension of the sensory(state) and action spaces. The state-action space is given by $S^{10} \times \mathcal{A} \times \mathcal{B}$, where $S = [0, 1]$ is the value range of the ten sonar signals, $\mathcal{A}$ is the set of available actions, and $B = \{0, 1, \ldots, 7\}$ is the set of possible target bearings. In the continuous form, this problem poses a great challenge for traditional reinforcement learning systems. Even in binary case in which the sonar signals are discretized, there are still at least $2^{10} * 5 * 8$ (approximately 40,000) possible combinations of states and actions.

Our experiments were based on a 16 by 16 minefield containing 10 mines. In each trial, each individual AV repeats the cycles of sense, act, and learn, until it reaches the target, hits a mine, or exceeds 30 sense-act-learn cycles. A reward of 1 is given when the AV reaches the target. A reward of 0 is given when the AV hits a mine. At each step of the trial, an immediate reward is estimated by computing a utility function $U = \frac{1}{1+rd}$, where rd is the remaining distance between the current position and the target position. When the AV runs out of time, the reward is computed using the utility function based on the remaining distance to the target. The reward scheme as designed encourages the AVs to approach the target and at the same time, avoid stepping into a mine.

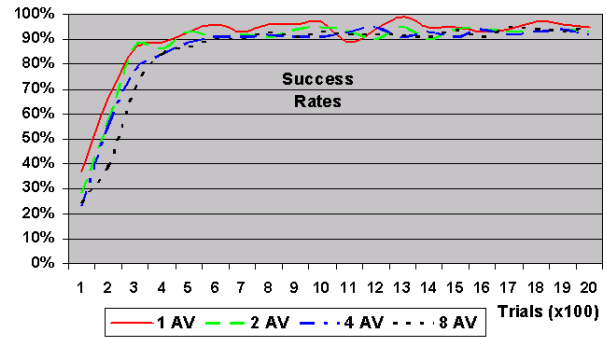All AVs were based on TD-FALCON with Bounded Q-

learning and used the same set of the parameter values: choice parameters $\alpha^{ck} = 0.001$ and learning rate parameters $\beta^{ck} = 1.0$ for $k = 1, 2, 3$; contribution parameters $\gamma^{c1} = \gamma^{c2} = 0.5, \gamma^{c3} = 0.0$; and vigilance parameters $\rho^{c1} = 0.5, \rho^{c2} = 0.2, \rho^{c3} = 0.5$. For temporal difference learning, the learning rate α was fixed at 0.5, the discount factor γ was set to 0.95, and the initial Q values were set to 0. For action selection, the decayed e-greedy policy was used with e initialized to 0.6 and decayed at a rate of 0.002.

Each AV learned from scratch through the feedback signals received from the environment. We repeated the experiments for 2000 trials. In each trial, the initial locations of the AVs, the location of the target, as well as the locations of the mines, were randomly generated. The success rate of the agent team in a trial was determined by the percent of AVs that successfully reach the target within the specified time in the trial.

Figure 3 summarizes the performance of the agent teams consisting of 1, 2, 4, and 8 AVs in terms of success rates averaged at 100-trial intervals. The success rates increased very fast right from the beginning. By the end of 500 trials, almost all teams can achieve more than 90 percent success rates. Teams with fewer agents produced better results.

To evaluate in quantitative terms how well the AVs traverse from the starting positions to the targets, we define a measure called *normalized step* given by $step_n = \frac{step}{sd}$, where $step$ is the number of sense-act-learn cycles taken to reach the target and sd is the shortest distance between the starting and target positions. A normalized step of 1 means the AV has taken the optimal(shortest) path to the target.

Figure 4 depicts the normalized steps taken by the agent teams consisting of 1, 2, 4, and 8 AVs averaged at 100-trial intervals. With just one agent, the normalized step converged to almost 1 after 1000 trials. Note that the *normalized step* values can seldom be 1's as detours are needed
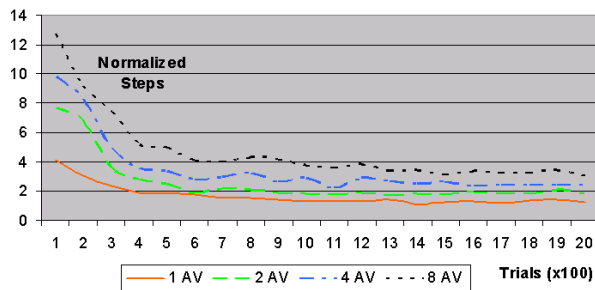
Figure 4. The average normalized steps taken by agent teams to reach the target over 2000 trials.
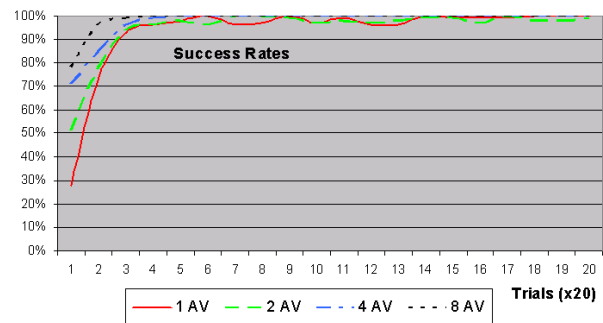

Figure 5. The success rates of agent teams on the minefield pursuit task.

when mines are on the optimal paths. Also, with more AVs in the system, the paths would always be less than optimal as the agents need to avoid collision with the mines as well as with each other.

4.2. The Minefield Pursuit Task

In this set of the experiments, modifications were made to the minefield navigation task to create a minefield pursuit scenario. The AVs function as predators to pursue a target, which moves in a random direction in each time step. Predators that collide or step into a mine are taken out of the system. If any of the remaining predators catches the prey (by occupying the same position), a trial is deemed as successful. When no predator remains in the system or no predator can capture the target within a specific time, the trial is deemed as a failure. The minefield pursuit task is more challenging than a typical predator and prey task [10] as each predator needs to avoid stepping into mines while pursuing the target.

The feedback scheme used in these experiments is as follows: A reward of 1 is given when a predator captures the target. A reward of 0 is given when a predator hits a mine. A reward of 0 is also given to predators who collide with each other. Intermediate feedback signals are presented to the predators during the trials using the utility function. The reward scheme encourages the predator to pursue the target, and at the same time, avoid colliding with each other or stepping into a mine.

Figure 5 summaries the performance of the agent teams consisting of 1, 2, 4, and 8 AVs in terms of success rates. We observe that agent teams with more agents tend to do better than smaller teams. This is expected as bigger teams will have a higher chance of eventually capturing the target. Nevertheless, all teams of various size can achieve close to 100% success rates after 400 trials.

5. Related Work

There has been extensive work done on reinforcement learning in multi-agent systems [10, 11]. Following the framework of Stone and Veloso, FALCON-based agents as described are homogeneous, deliberative, and non-communicating.

The minefield navigation task described in this paper is similar to the NRL navigation and mine avoidance domain. To tackle the problem, Gordan and Subramanian [6] developed two sets of cognitive models, one for predicting the next sonar and bearing configuration based on the current sonar and bearing configuration and the chosen action; and the other for estimating the desirability of the sonar and bearing configurations. Sun et. al. [12] used a three-layer Backpropagation neural network to learn the Q values and an additional layer to perform stochastic decision making based on the Q values. Compared with backpropagation network and other function approximation methods based on gradient descent [13], FALCON has the advantages of fast incremental learning and self-organizing architecture.

The minefield pursuit task is an extension of the pursuit domain originally devised by Brenda et al [7]. Levy and Rosenschein [14] used a game theoretic function for agents to cooperate. Korf [15] introduced a policy for each predator, which greedily maximized its own utility function. Compared with prior work, FALCON has no explicit representation of goals and mechanism for collaboration. The system behaviour is purely guided by the feedback signals received from the environment. Our findings are consistent with those of Korf that explicit collaboration is not needed at least for the pursuit domain and that coordination and cooperation can emerge through the interaction of individual agents.

6. Conclusion

Two critical issues of multi-agent systems are scalability and co-adaptation dynamics. By using a self-organizing learning architecture, TD-FALCON offers an efficient solution to the scalability problem of dealing with large and continuous state-action spaces. Specifically, its online incremental learning capability enables an agent to adapt and operate in a real-time multi-agent environment. The proposed model though simple serves as a starting point and provides the foundation for building complex autonomous agents with high level cognitive capabilities.

References

[1] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, Cambridge, MA: MIT Press, 1998.

[2] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54–115, June 1987.

[3] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multi-dimensional maps," *IEEE Transactions on Neural Networks*, vol. 3, pp. 698–713, 1992.

[4] A. H. Tan, "Adaptive Resonance Associative Map," *Neural Networks*, vol. 8, no. 3, pp. 437–446, 1995.

[5] A. H. Tan, "FALCON: A fusion architecture for learning, cognition, and navigation," in *Proceedings, IJCNN, Budapest*, 2004, pp. 3297–3302.

[6] D. Gordan and D. Subramanian, "A cognitive model of learning to navigate," in *Nineteenth Annual Conference of the Cognitive Science Society*, 1997.

[7] M Brenda, V. Jagannathan, and R. Dodhiawala, "On optimal cooperation of knowledge sources - an empirical investigation," Tech. Rep., Boeing Advanced Technology Center, 1986, Technical Report BCS-G2010-28.

[8] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759–771, 1991.

[9] A. Pérez-Uribe, *Structure-Adaptable Digital Neural Networks*, Ph.D. thesis, Swiss Federal Institute of Technology-Lausanne, 2002.

[10] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," Technical Report GMU-CS-TR-2003-1. George Mason University, 2003.

[11] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, pp. 345–383, 2000.

[12] R. Sun, E. Merrill, and T. Peterson, "From implicit skills to explicit knowledge: a bottom-up model of skill learning," *Cognitive Science*, vol. 25, no. 2, pp. 203–244, 2001.

[13] R.S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems 12*. 2000, pp. 1057–1063, MIT Press.

[14] R. Levy and J.S. Rosenschein, "A game theoretic approach to the pursuit problem," in *Working papers of the 11th International Workshop on Distributed Artificial Intelligence*. 1992, pp. 195–213, San Francisco, CA. Morgan Kaufmann.

[15] R.E Korf, "A simple solution to pursuit games," in *Working papers of the 11th International Workshop on Distributed Artificial Intelligence*. 1992, pp. 183–194, San Francisco, CA. Morgan Kaufmann.