12-2015

Fast reinforcement learning under uncertainties with self-organizing neural networks

Teck-Hou TENG

Ah-hwee TAN
Singapore Management University, ahtan@smu.edu.sg

Citation

TENG, Teck-Hou and TAN, Ah-hwee. Fast reinforcement learning under uncertainties with self-organizing neural networks. (2015). *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2015)*. 2, 51-58.
Available at: https://ink.library.smu.edu.sg/sis_research/6797

2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

Fast Reinforcement Learning under Uncertainties with Self-Organizing Neural Networks

Teck-Hou Teng and Ah-Hwee Tan
School of Computer Engineering
Nanyang Technological University, Singapore
{thteng,asahtan}@ntu.edu.sg

***Abstract*—Using feedback signals from the environment, a reinforcement learning (RL) system typically discovers action policies that recommend actions effective to the states based on a Q-value function. However, uncertainties over the estimation of the Q-values can delay the convergence of RL. For fast RL convergence by accounting for such uncertainties, this paper proposes several enhancements to the estimation and learning of the Q-value using a self-organizing neural network. Specifically, a temporal difference method known as Q-learning is complemented by a Q-value Polarization procedure, which contrasts the Q-values using feedback signals on the effect of the recommended actions. The polarized Q-values are then learned by the self-organizing neural network using a Bi-directional Template Learning procedure. Furthermore, the polarized Q-values are in turn used to adapt the reward vigilance of the ART-based self-organizing neural network using a Bi-directional Adaptation procedure. The efficacy of the resultant system called Fast Learning (FL) FALCON is illustrated using two single-task problem domains with large MDPs. The experiment results from these problem domains unanimously show FL-FALCON converging faster than the compared approaches.**

***Index Terms*—Intelligent Agent, Self-Organizing Neural Network, Reinforcement Learning**

I. Introduction

Using feedback signals from the environment, reinforcement learning (RL) discovers action policies that recommend actions effective to the states. The effectiveness of the action policies is signified using the Q-value. Estimated using Q-learning [1], the Q-value can fluctuate due to uncertainties over the feedback signal [2] and the states [3]. RL cannot converge when the Q-values do not saturate.

RL problems with state uncertainties are addressed by [4], [5], [6], [3]. [2] addresses uncertainties over credit assignment. Other works include the fuzzy-based approach [7], a probabilistic-based approach [8] and kernel-based approach [9] are also known. However, most of these works are mostly seen in problems with uncertainties over the states, the feedback signals and the action policies. In contrast, this work addresses the uncertainty to the estimation and learning of the Q-values.

This work observes that the fluctuation of the Q-values can delay the convergence of RL. The hypothesis is that RL can converge faster by accounting for the uncertainties that delay the saturation of the Q-values. Therefore, this work proposes the polarization of the Q-value estimated using Q-learning using a Q-value Polarization procedure. This polarized Q-value is then learned using a new template learning procedure known as the Bi-direction Template Learning. In addition, the polarized Q-value is used to adapt the reward vigilance using the Bi-directional Adaptation procedure.

The function approximator is based on the adaptive resonance theory (ART) [10]. It is an ART-based self-organizing neural network derived from FALCON [11]. The proposed enhancements are integrated with the one-shot incremental learning FALCON to give a fast learning (FL) FALCON. The efficacy of FL-FALCON is illustrated using the Minefield Navigation Task (MNT) and the Pursuit-Evasion (PE) problem domains. The experiments based on the MNT problem domain illustrate the performance difference FL-FALCON has over the comparable approaches. The experiments based on the PE problem domain compare the efficacy of the proposed enhancements and the comparable approaches.

The presentation of this paper continues in Section II with a survey of the related works. This is followed by a summary of the self-organizing neural network in Section III. The proposed enhancements to this self-organizing neural network are introduced in Section IV. After that, the problem domains, the experiments and the results are presented in Section V. The conclusion is seen in Section VI.

II. Related Work

A large body of reinforcement learning (RL) approaches exists, which addresses uncertainties over the states, the action policies and the feedback signals. For example, a model-based policy search method known as PILCO was proposed to improve learning speed in the continuous state and the control spaces [6]. Model uncertainty is incorporated into controller learning and long-term planning to reduce model errors. This approach learns a probabilistic, non-parametric Gaussian process transition model of the system with unprecedented learning speed.

A Knowledge Evaluation-based Credit Assignment approach is proposed for addressing the uncertainty over the credit assignment problem [2]. A knowledge measure known as *certainty* derived using the Q-value estimates the role of the knowledge to the overall performance. State uncertainty due to noisy stimuli can create difficulties to the allocation of the observed rewards. To address the state uncertainty, [3] proposes the Posterior Weighted RL (PWRL) to update the estimated state probabilities using the observed rewards. The

PWRL approach can react to the switching of the reward distribution using estimations of the variance.

Large scale disassembling line balancing problems with uncertainty can be addressed using Monte-Carlo-based RL [12]. The RL approach addresses uncertainty due to demand fluctuations. It converges to optimality after training for a reasonable number of episodes. Longer training is necessary for RL to converge when the demand is stochastic. The k-Certainty Exploration method was used to track the use of the learned rules and assign a k-certainty factor to the tracked rules [13]. Using this approach, uncertainty over the learned rules is reduced and the environment can be identified gradually. An l-Certainty Exploration method that considers the state transition probabilities is also proposed to identify the environment more efficiently.

A Weighted Strategy Sharing architecture is proposed for agents to learn from each other [4]. Using different measures of expertness for agents with different experiences, the experiment results show that certain expertness measures are less sensitive when the knowledge is incorrect and data is uncertain. The uncertainty over the states can also be addressed using an automatic action-based soft partitioning of the state space [5]. This approach uses multiple agents to suggest actions based on partial observations of the state space. The suggested actions are then fused using an expertness criterion. Results from experiments based on real-world setting show the proposed learning approach can converge faster.

A Generalized Probabilistic Fuzzy RL (GPFRL) algorithm is proposed for systems with uncertain conditions [7]. GPFRL has a simple control structure, high learning efficiency and generalizes well in stochastic environments. The learned action policies are used with the weighted combination of action probabilities to avoid risky behaviours effectively. Online learning controllers can be used to address issues with unreliable teachers [8]. This approach assigns subjective probability that approaches 1.0 to the optimal action of the events. A Potential Function Method is used in cases where the features are continuous.

A Kernel-based Dual Heuristic Programming (KDHP) is proposed for learning control policies in real time [9]. KDHP is found to be better than DHP and multi-layer perceptron neural networks at approximating control policies of real-time systems with model uncertainties. The Dyna-$\mathcal{H}$ algorithm was proposed for sequential decision-making under uncertainty [14]. Dyna-$\mathcal{H}$ searches for the possible solutions by using a general function to guess the worst action a for state s. Experiment results show Dyna-$\mathcal{H}$ can find the actions effective to the states more efficiently than Q-learning and Dyna-Q.

The surveyed RL approaches are used in problems with uncertainties over the states, the action policies and the feedback signals. However, none of these works consider the uncertainty to the estimation of the value function. This work proposes novel enhancements that allow reinforcement learning to converge fast in situations with such uncertainties.

III. The TD-FALCON Neural Network

TD-FALCON is an ART-based self-organizing neural network derived from FALCON [11]. It is used as a function approximator during reinforcement learning. Learning incrementally in real time, TD-FALCON generalizes on the vector patterns without compromising on its prediction accuracy.

Algorithm 1 The Fusion ART algorithm

```
Require: Activity vectors $\mathbf{x}^{ck}$ and all weights vector $\mathbf{w}_j^{ck}$
 1: for each $F_2^c$ node $j$ do
 2:     Code Activation: Derive choice function $T_j^c$ using
            $$T_j^c = \sum_{k=1}^{3} \gamma^{ck} \frac{|\mathbf{x}^{ck} \wedge \mathbf{w}_j^{ck}|}{\alpha^{ck} + |\mathbf{w}_j^{ck}|}$$
        where the fuzzy AND operation $(\mathbf{p} \wedge \mathbf{q})_i \equiv min(p_i, q_i)$, the norm
        $\|.\|$ is defined by $|\mathbf{p}| \equiv \sum_i p_i$ for vectors $\mathbf{p}$ and $\mathbf{q}$, $\alpha^{ck} \in [0,1]$ is
        the choice parameters, $\gamma^{ck} \in [0,1]$ is the contribution parameters and
        $k = \{1,2,3\}$
 3: end for
 4: repeat
 5:     Code Competition: Index of winning cognitive node $J$ is found using
            $$J = \arg\max_j \{T_j^c : \text{for all } F_2^c \text{ node } j\}$$
 6:     Template Matching: Check whether the match functions $m_J^{ck}$ of
        cognitive node $J$ meet the vigilance criterion
            $$m_J^{ck} = \frac{\|\mathbf{x}^{\mathrm{ck}} \wedge \mathbf{w}_J^{ck}\|}{|\mathbf{x}^{\mathrm{ck}}|} \geq \rho^{ck}$$
        where $\rho^{ck} \in [0,1]$ for $k = \{1,2,3\}$ are the vigilance parameters
 7:     if vigilance criterion is satisfied then
 8:         Resonance State is attained
 9:     else
10:         Match Tracking: Modify state vigilance $\rho^{c1}$ using
                $$\rho^{c1} = \min\{m_J^{\mathrm{ck}} + \psi, 1.0\}$$
            where $\psi$ is a very small step increment to match function $m_J^{ck}$
11:         Reset: $T_j^c = 0.0$
12:     end if
13: until Resonance State is attained
14: if operating in LEARN/INSERT mode then
15:     Template Learning: modify weight vector $\mathbf{w}_J^{ck}$ using
            $$\mathbf{w}_J^{ck(\mathrm{new})} = (1-\beta^{ck})\mathbf{w}_J^{ck(\mathrm{old})} + \beta^{ck}(\mathbf{x}^{ck} \wedge \mathbf{w}_J^{ck(\mathrm{old})}) \quad (1)$$
        where $\beta^{ck} \in [0,1]$ is the learning rate
16: else if operating in PERFORM mode then
17:     Activity Readout: Read out the action vector $\mathbf{A}$ of cognitive node $J$
        using
            $$\mathbf{x}^{c2(new)} = \mathbf{x}^{c2(old)} \wedge \mathbf{w}_J^{c2}$$
        Decode $\mathbf{x}^{c2(new)}$ to derive recommended action choice $a$
18: end if
```

A. Structure and Operating Modes

Seen in Fig. 1, TD-FALCON [15] has a two-layer architecture, comprising an input/output (IO) layer and a knowledge layer. The IO layer has a sensory field F_1^{c1} for accepting state vector $\mathbf{S}$, an action field F_1^{c2} for accepting action vector $\mathbf{A}$, and a reward field F_1^{c3} for accepting reward vector $\mathbf{R}$. The category field F_2^c at the knowledge layer stores the committed and uncommitted cognitive nodes. Each cognitive node j has template weights $\mathbf{w}^{\mathrm{ck}}$ for $k = \{1,2,3\}$.

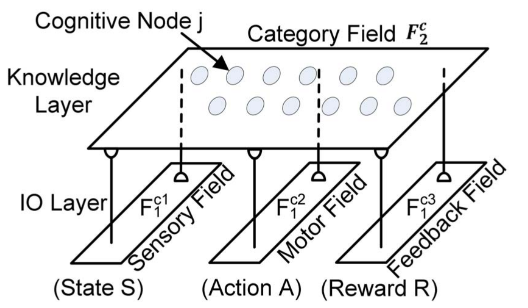


Fig. 1. The TD-FALCON architecture.

TD-FALCON has three modes of operation. In the *PERFORM* mode, Algorithm 1 is used to select cognitive node J for deriving action choice a for state s. In the *LEARN* mode, TD-FALCON learns the effect of action choice a on state s. In the *INSERT* mode, domain knowledge can be assimilated into FALCON [16].

B. Incorporating Temporal Difference Method

Seen in Algorithm 2, a temporal difference (TD) method known as the Q-Learning [1] is used to estimate the Q-value $Q(s,a)$ of action choice a in state s [17]. At state s', the estimated Q-value is used as the teaching signal to learn the association of state s and action choice a.

Algorithm 2 The TD-FALCON algorithm

```
1: Initialize FALCON
2: Sense the environment and formulate a state representation $s$
3: Choose to explore at a probability of $\epsilon$
4: if Exploration then
5:     Use Exploration Strategy [18] to select an action choice $a$
6: else if Exploitation then
7:     Use Direct Code Access [15] to select an action choice from existing knowledge
8: end if
9: Evaluate effect of action choice $a$ to derive a reward $r$ from the environment
10: Estimate the $Q$-value function $Q(s,a)$ following a TD formula given by $\Delta Q(s,a) = \alpha TD_{err}$
11: Present $\mathbf{S}$, $\mathbf{A}$ and $\mathbf{R}$ for Learning
12: Update the current state $s = s'$
13: Repeat from Step 2 until $s$ is a terminal state
```

TD-FALCON can be used for reinforcement learning by representing State s as state vector $\mathbf{S}$, action choice a as action vector $\mathbf{A}$ and the estimated Q-value as reward vector $\mathbf{R}$. Using memory-based learning, the tuple $\{\mathbf{S}, \mathbf{A}, \mathbf{R}\}$ is learned as cognitive node j. Algorithm 1 is used to select the best-matching cognitive node J during learning and action selection. Algorithm 2 outlines the reinforcement learning framework where TD-FALCON is used.

Iterative Value Estimation: The TD method incorporated into FALCON is known as the Bounded Q-Learning [17]. It estimates the value of applying action choice a to state s iteratively using

$$Q^{new}(s,a) = Q(s,a) + \alpha TD_{err}(1 - Q(s,a)), \tag{2}$$

where $\alpha \in [0,1]$ is the learning parameter and TD_{err} is the temporal error term which is derived using

$$TD_{err} = r + \gamma \max_{a'} Q(s',a') - Q(s,a), \tag{3}$$

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

C. Knowledge Pruning

Ineffective action policies are pruned to facilitate more efficient operation. A confidence-based pruning strategy similar to [11] is adapted to prune the cognitive nodes that encode the ineffective action policies.

Specifically, cognitive node j has a confidence level c_j where $c_j \in [0.0, 1.0]$and an age σ_j where $\sigma_j \in [0, \mathcal{R}]$. A newly committed cognitive node j has an initial confidence level $c_j(0)$ and an initial age $\sigma_j(0)$. The confidence level c_J of winning cognitive node J is reinforced using

$$c_J^{new} = c_J^{old} + \eta(1 - c_J^{old}),$$

where η is the reinforcement rate of the confidence level. After each training iteration, the age σ_j of cognitive node j is incremented and its confidence level c_j is decayed using

$$c_j^{new} = c_j^{old} - \zeta c_j^{old},$$

where ζ is the decay rate of the confidence level. The age attribute σ_j of cognitive node j prevents pre-mature pruning. Cognitive node j is pruned only when $c_j < c^{rec}$ where c^{rec} is the recommended confidence threshold and $\sigma_j \geq \sigma^{old}$ where σ^{old} is the old age threshold.

IV. The FL-FALCON Neural Network

The proposed enhancements for a fast learning (FL) version of FALCON known as the FL-FALCON are presented here. These enhancements are proposed to handle the uncertainties seen in the estimation of the Q-value. Integrating these enhancements with the one-shot incremental learning FALCON, reinforcement learning can converge quickly by saturating the estimated Q-value in the least amount of time.

A. Q-value Polarization

From (2) and (3), $Q(s,a)$ is estimated using the immediate reward r, $Q(s,a)$ and s' $\max_{a'} Q(s',a')$. Such an approach introduces uncertainty into the estimation of $Q^{new}(s,a)$. The immediate reward r introduces uncertainty through inaccurate user-specified reward function. Uncertainty is introduced by $Q(s,a)$ and $\max_{a'} Q(s',a')$ because both values are estimated by using Algorithm 1 to find the winning cognitive nodes. Using this approach, there is a non-zero probability of selecting cognitive nodes inappropriate for the states.

Assuming a winning cognitive node J_1 is used in the *PERFORM* mode to derive action choice a and a winning cognitive node J_2 is used in the *LEARN* mode to derive $Q(s,a)$. Using the same search criteria, the expected outcome is for $J_1 \equiv J_2$. However, due to perceptual aliasing [19] and the generalization of the learned action policies, there is a non-zero probability that $J_1 \neq J_2$. In addition, Q-learning is slow [20] in responding to changes to the effect of action choice a in state s.

Therefore, the Q-value Polarization method (QPolar) is proposed to address such issue using

$$Q^{new}(s,a) = \varepsilon(s,a)Q_g + (1-\varepsilon(s,a))Q_b, \tag{4}$$

where $\varepsilon(s,a) \in \{0.0, 1.0\}$ is the polarity of action choice a on state s and is obtained using a polarization function aligned to the reward function. $Q_g \in [0.0, 1.0]$ is the $Q(s,a)$ for when $\varepsilon(s,a) = 1.0$ and $Q_b \in [0.0, 1.0]$ is the $Q(s,a)$ for when $\varepsilon(s,a) = 0.0$ and are derived using

$$Q_g = \left\{\tfrac{Q_g + Q^{new}(s,a)}{2} \vee (Q_b + Q_m)\right\} \wedge \zeta_g$$

and

$$Q_b = \left\{\tfrac{Q_b + Q^{new}(s,a)}{2} \wedge (Q_g - Q_m)\right\} \vee \zeta_b,$$

where $(p \vee q) \equiv \max(p,q)$, $(p \wedge q) \equiv min(p,q)$, Q_m is the polarity margin, ζ_g is the upper bound of Q-value and ζ_b is the lower bound of Q-value.

For Comparison: The *certainty* measure $C(s,a)$ seen in [2] is considered as an alternative approach (certainty-Q) for handling the uncertainty seen in estimating the Q-value using

$$Q^{new}(s,a) = \{Q^{new}(s,a) + \varepsilon(s,a)C(s,a)\} \wedge \zeta_g \tag{5}$$

$C(s,a)$ is a measure of the agent's knowledge on action choice a in state s and is derived using

$$C(s,a) = \frac{e^{Q(s,a)/T}}{\sum\limits_{a' \in \mathcal{A}} e^{Q(s,a')/T}}$$

where $T \in [0.0, 1.0]$ is the scaling factor and $\mathcal{A}$ is the action space.

B. Bi-Directional Adaptation

The state and action vigilances specify the required similarity between the activity vectors $\mathbf{x}^{\mathrm{ck}}$ and the template weights $\mathbf{w}^{\mathrm{ck}}$ for $k = \{1, 2\}$. The reward vigilance ρ^{c3} is used to select cognitive nodes that recommend action choices that are effective to the states, i.e., $\varepsilon(s,a) = 1$. This means a suitable ρ^{c3} has to be used. Given that it is difficult to know *a priori* a suitable ρ^{c3}, ρ^{c3} was adapted iteratively [16] using

$$\rho^{\mathrm{c3}-} = \{\nu\rho^{\mathrm{c3}} + (1-\nu)\alpha_l Q(s,a)\} \wedge \rho^{\mathrm{c3}}, \tag{6}$$

where ν is the adaptation rate, α_l suppresses $Q(s,a)$ for the downward adaptation and $Q(s,a)$ is the estimated Q-value.

To use (6), ρ^{c3} has to be initialized using a high value. The reward vigilance ρ^{c3} is then adapted downwards. Adapting ρ^{c3} in this way improves the performance of FALCON. However, as (6) only adapts ρ^{c3} downwards, i.e., $\rho^{\mathrm{c3}} \to 0.0$, it cannot handle noisy situations where ρ^{c3} has to be adapted upwards, i.e., $\rho^{\mathrm{c3}} \to 1.0$. To adapt ρ^{c3} in either directions, the proposed Bi-directional Adaptation (BiReward) procedure adapts ρ^{c3} using

$$\rho^{\mathrm{c3(new)}} = \varepsilon(s,a)\rho^{\mathrm{c3}-} + \{1-\varepsilon(s,a)\}\rho^{\mathrm{c3}+},$$

where $\varepsilon(s,a) \in \{0.0, 1.0\}$ is the polarity of action choice a on state s, $\rho^{\mathrm{c3}-}$ is the adapted ρ^{c3} when $\varepsilon(s,a) = 1$ and $\rho^{\mathrm{c3}+}$ is the adapted ρ^{c3} when $\varepsilon(s,a) = 0$ and is derived using

$$\rho^{\mathrm{c3}+} = \{\nu\rho^{\mathrm{c3}} + (1-\nu)\alpha_u Q(s,a)\} \vee \rho^{\mathrm{c3}},$$

where α_u boosts $Q(s,a)$ for the upward adaptation. The reward vigilance ρ^{c3} can be adapted using either $Q^{new}(s,a)$ (PolarizedQ) or the learned $Q(s,a)$ (estQ). Details on the use of either $Q^{new}(s,a)$ or the learned $Q(s,a)$ are seen in the following paragraphs.

Using $Q^{new}(s,a)$ (PolarizedQ): This approach uses the Polarized Q-value derived using (4) or (5) to adapt ρ^{c3}. After that, the adapted ρ^{c3} is used when FL-FALCON operates in the *LEARN* mode.

Using $Q(s,a)$ (estQ): This approach adapts ρ^{c3} after FL-FALCON completes learning. FL-FALCON operates in the *LEARN* mode using ρ^{c3} adapted in the previous cycle. ρ^{c3} is adapted using the learned $Q(s,a)$ derived using the same approach for deriving $Q(s,a)$ used in (2) and (3).

C. Bi-directional Template Learning

The uncertainty over the estimated Q-value is seen in the form of fluctuating $Q^{new}(s,a)$. $Q^{new}(s,a)$ is encoded as a reward vector $\mathbf{R}$ and learned as $\mathbf{w}_J^{\mathrm{c3}}$. To ensure $\mathbf{w}_J^{\mathrm{c3}}$ is adapted towards the main trending direction of $Q^{new}(s,a)$, it is necessary to adapt $\mathbf{w}_J^{\mathrm{c3}}$ in either directions. Therefore, the Bi-directional Template Learning (BiLearn) approach is proposed to adapt the template weights $\mathbf{w}_J^{\mathrm{ck(new)}}$ using

$$\mathbf{w}_J^{\mathrm{ck(new)}} = (1-\beta^{\mathrm{ck}})\{\mathbf{x}^{\mathrm{ck}} \vee \mathbf{w}_J^{\mathrm{ck(old)}}\} + \beta^{\mathrm{ck}}\{\mathbf{x}^{\mathrm{ck}} \wedge \mathbf{w}_J^{\mathrm{ck(old)}}\}, \tag{7}$$

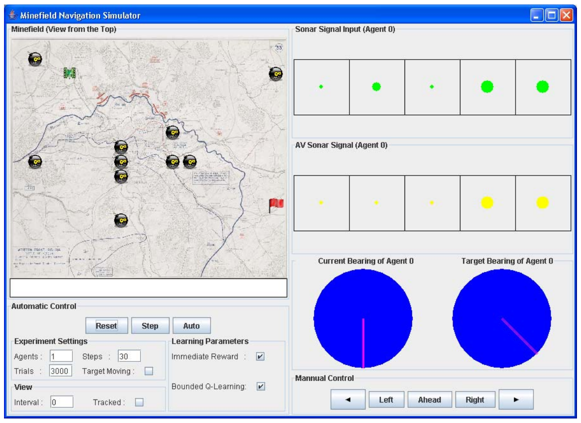
where $\mathbf{x}^{\mathrm{ck}}$ is the activity vector, $\mathbf{w}_J^{\mathrm{ck(old)}}$ is the template weights of cognitive node J and $k = \{1, 2, 3\}$. (7) is proposed to replaced (1) seen in Step 15 of Algorithm 1. When $\beta^{\mathrm{ck}} = 1$ for $k = \{1, 2, 3\}$, (7) is equivalent to (1).

V. Experiments and Results

Experiments were conducted to evaluate and compare the performance of FL-FALCON. The MNT problem domain and the experiment results are presented in Section V-A. The PE problem domain and the experiment results are presented in Section V-B. The default parameters used in these two problem domains are seen in Table I.

TABLE I
THE DEFAULT PARAMETERS USED IN THE EXPERIMENTS

FALCON for $k = \{1, 2, 3\}$	
Choice Parameters α^{ck}	$\{0.1, 0.1, 0.1\}$
Contribution Parameters γ^{ck}	$\{0.333, 0.333, 0.333\}$
Learning Rates β^{ck}	$\{1.0, 1.0, 1.0\}$
Vigilance ρ^{ck}	$\{0.95, 0.0/1.0, \rho^{c3}\}$
Essential parameters	
TD Learning $\{\alpha, \gamma$, Initial Q-value$\}$	$\{0.1, 0.5, 0.5\}$
Pruning - $\{\sigma^{old}, c^{rec}\}$	$\{50$ iterations$, 0.65\}$
Confidence - $\{c_j(0), \zeta, \eta\}$	$\{0.5, 0.0005, 0.5\}$
ρ^{c3} Adaptation - $\{\nu, \alpha_u, \alpha_l\}$	$\{0.5, 1.05, 0.95\}$
Q-value Polarization - $\{Q_b, Q_g, Q_m, \zeta_g, \zeta_b\}$	$\{0.25, 0.75, 0.5, 0.95, 0.10\}$


Minefield Navigation Simulator
Minefield (View from the Top)
Sonar Signal Input (Agent 0)
AV Sonar Signal (Agent 0)
Current Bearing of Agent 0
Target Bearing of Agent 0
Automatic Control
Reset
Step
Auto
Mannual Control
◄
Left
Ahead
Right
►

Experiment Settings
Agents : 1
Steps : 30
Trials : 3000
Target Moving : ☐

View
Interval : 0
Tracked : ☐

Learning Parameters
Immediate Reward : ☑
Bounded Q-Learning: ☑

Fig. 2. The minefield navigation simulator.

A. Minefield Navigation Task Problem Domain

Problem Description: This problem domain has an autonomous vehicle (AV) that navigates a minefield (see Fig. 2) to a random target position within a specified time frame. Ten mines are randomly distributed throughout a 16×16 minefield. The task fails when AV hits a mine or when it is unable to reach the target position using the allocated number of steps. The task succeeds when AV reaches the target position within the allocated number of steps and without moving into any mine along the way. AV is placed at a random vacant location and navigates the minefield by sensing, acting and learning iteratively. The target position and the location of the mines remain unchanged during training.

This problem domain has a large MDP because the state space comprises five continuous sonar readings and eight readings of the relative target bearing. Sonar readings of the mines are available from the left, left diagonal, front, right diagonal and right. The sonar reading at direction i is measured by $s_i = \frac{1}{d_i}$, where d_i is the distance to an obstacle which can be a mine or the boundary of the minefield. The relative target bearing comprises front, diagonal right, right, diagonal back right, back, diagonal back left, left and diagonal left.

Experiment Results: The experiments in the MNT problem domain are conducted for 20 runs and for $2,000$ training iterations per run. The mean values for one set of results are derived using a fixed size window of 100 training iterations. The success rates seen in Fig. 3 and the code population seen in Fig. 4 are the mean values based on the 20 runs of the experiments.

From Fig. 3, FL-FALCON-PolarizedQ is seen having the best success rates. TD-FALCON-Reward-Vigilance-Adaptation is seen as the next best performing approach. Using (5) with $T = 0.5$, the success rates of FL-FALCON-CertaintyQ is seen fluctuating between 80% and 92%, only to begin to saturate at around the 90% level. Q-learning is seen performing better than TD-FALCON. The success rates of TD-FALCON is seen at around the 90% level after $2,000$ training iterations. The success rates of the Monte-Carlo-Simulation is seen fluctuating between 10% and 20%.

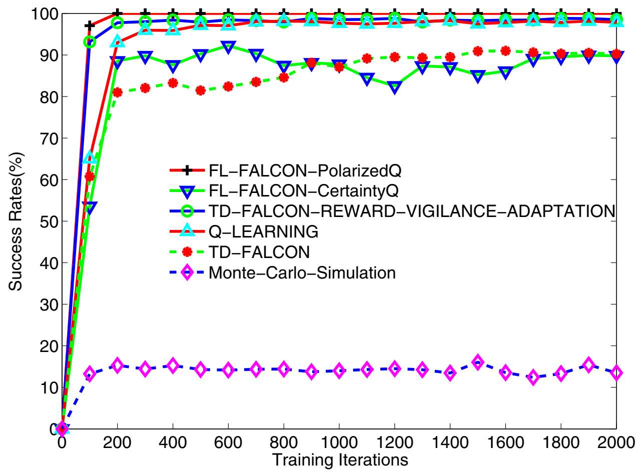


Fig. 3. Comparison of the success rates.

With pruning, Q-learning is still seen in Fig. 4 with the highest code population. TD-FALCON is seen with the next highest code population and converge to similar level of code population as TD-FALCON-Reward-Vigilance-Adaptation. The code population of TD-FALCON-Reward-Vigilance-Adaptation is seen rising gradually to around 300 nodes while that of FL-FALCON-CertaintyQ is seen fluctuating and ends with the highest code population after $2,000$ training iterations. In contrast, the code population of FL-FALCON-PolarizedQ stabilises at less than 100 nodes after around 100 training iterations, only to be pruned further from the 1400^{th} training iteration.

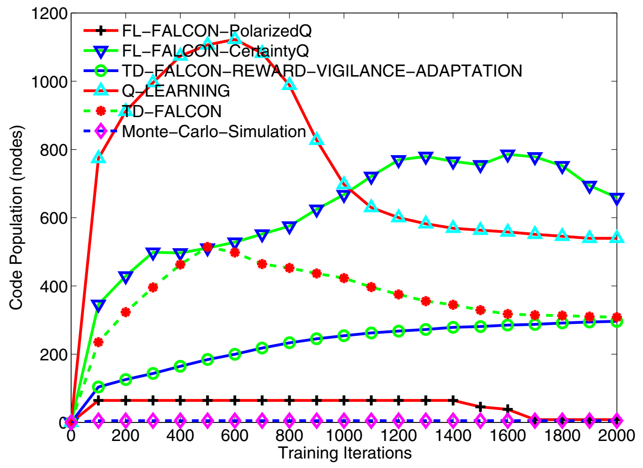


Fig. 4. Comparison of the code population.

From Fig. 3 and Fig. 4, FL-FALCON-PolarizedQ is seen having significantly better performance than the compared approaches. It is the fastest approach to reach 100% success rates and has the smallest code population.

B. Pursuit-Evasion Problem Domain

Problem Description: Illustrated in Fig. 5, this problem domain has two virtual agents known as the Blue agent and the Red agent. The Red agent is hostile towards the Blue agent. The Blue agent can only evade the Red agent to avoid elimination. The Red agent touches the Blue agent to eliminate it. There are two safe areas where the Blue agent is safe from the Red agent. Moving constantly, the Blue agent can move into the safe areas to evade the Red agent but does not remain in it. Like [18], the pursuit strategy of the Red agent is deterministic while the Blue agent learns the evasive strategies. The feedback signal to the Blue agent indicates the effectiveness of the evasive maneuvres.

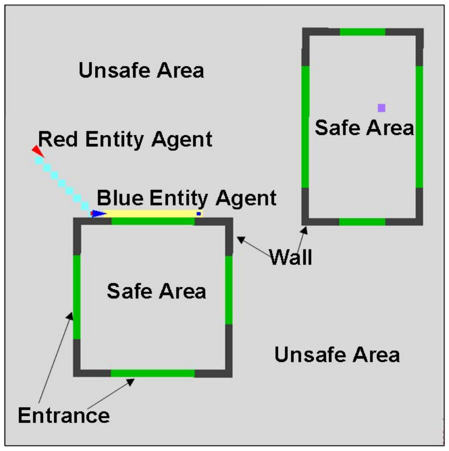


Fig. 5. The PE problem domain with safe and unsafe areas.

Based on a situation-awareness model [21], the state space of the agent comprises eight types of multi-valued attributes on the *enemy* and the *terrain*. The *perception* layer has four types of attributes, the *comprehension* layer has three types of attributes and the *projection* layer has just one type of attribute. Therefore, this problem domain has a large MDP with up to around 2.86×10^{10} possible states.

For evading the Red agent, the action space of the Blue agent comprise eight compass directions. The compass directions are north, northeast, east, southeast, south, southwest, west and northwest. The effect of choosing an evade direction as a response to the states is learned. The learned action policies may be exploited in the subsequent cycles.

Experiment Results: Experiments in the PE problem domain were conducted for 20 runs and for 500 training iterations per run. The mean values for one set of results are derived using a fixed size window of 20 training iterations. The mission completion rates seen in Fig. 6, the exploitation rates seen in Fig. 7 and the code population seen in Fig. 8 are the mean values based on the 20 runs of the experiments.

From Fig. 6, FL-FALCON-PolarizedQ has the best-performing outcome. TD-FALCON-QPolar-estQ-BiLearn has the next best-performing outcome. The mission completion rates of FL-FALCON-estQ lags in the first 80 training iteration but reaches 100% at around the same time as these two approaches. TD-FALCON-QPolar-estQ-BiReward is the last approach to attain 100% mission completion rates. The mission completion rates of TD-FALCON saturates at around 75%. Using (5) with $T = 0.5$, the mission completion rates of FL-FALCON-CertaintyQ is seen improving slowly towards 75%. The mission completion rates of the Monte-Carlo-Simulation fluctuates between 10% and 30%.

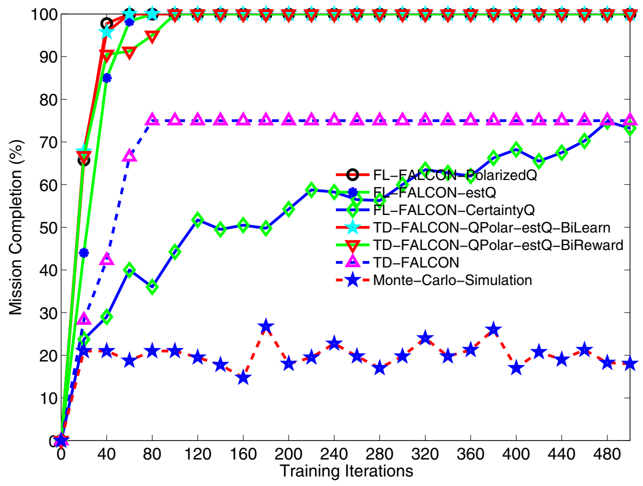


Fig. 6. Comparison of the mission completion rates.

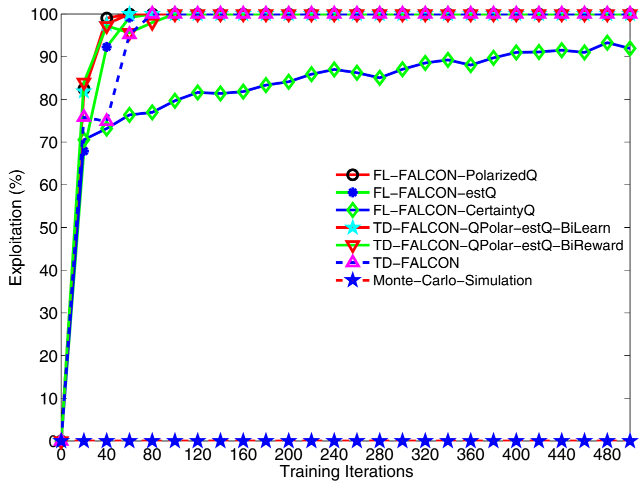


Fig. 7. Comparison of the exploitation rates.

From Fig. 7, all FALCON-based approaches are seen with 100% exploitation rates after around 100 training iterations. The order of saturation of the exploitation rates tracks the order of saturation of the mission completion rates in Fig. 6. The exploitation rates of TD-FALCON also saturates at 100% though its mission completion rates reaches only 75%. The exploitation rates of FL-FALCON-CertaintyQ is seen improving slowly. The exploitation rates of Monte-Carlo-Simulation is expectedly seen at 0%.

From Fig. 8, FL-FALCON-estQ and FL-FALCON-CertaintyQ have similar peak code population. The code population of FL-FALCON-PolarizedQ and TD-FALCON-QPolar-estQ-BiLearn peaks at similar levels. In comparison, the code population of TD-FALCON-QPolar-estQ-BiReward peaks earlier but is pruned gradually to similar levels of code population. Among the highest for almost the entire training process, the code population of FL-FALCON-CertaintyQ peaks and declines slowly. In contrast, the code population of TD-FALCON has the lowest peak and is also pruned to similar level as the other approaches. In general, the code population tracks the movement of the exploitation rates.

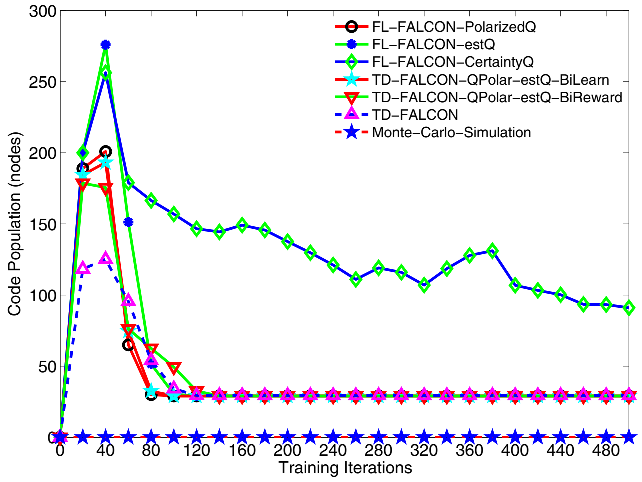


Fig. 8. Comparison of the code population.

VI. Conclusion

This work addresses the delay to the convergence of reinforcement learning due to the uncertainty on the estimation of the Q-value in problem domains with large markov decision processes. The uncertainties are attributed to perceptual aliasing and the inappropriate choice of generalized action policies. This work improves the estimation of the Q-values using a Q-value Polarization procedure. The learning of the estimated Q-values by a self-organizing neural network is improved using a Bi-directional Adaptation of the reward vigilance and a Bi-directional Template Matching. The integration of these enhancements gives a fast learning self-organizing neural network known as the FL-FALCON.

The efficacy of FL-FALCON is illustrated using the MNT and the PE problem domains. The experiments based on the MNT problem domain compare the performance of FL-FALCON with recent versions of FALCON and a few other approaches. The experiment results show FL-FALCON with Polarized Q-value (FL-FALCON-PolarizedQ) attains 100% success rates faster than the compared approaches. The experiments based on the PE problem domain compare the uses of the proposed enhancements and the other approaches. Comparing to the other approaches, the experimental results show FL-FALCON-PolarizedQ has the smallest code population and the best performance outcome.

The use of FL-FALCON in these two single-task problem domains reveals little rooms for improvement. Therefore, the plan is to move beyond single-task problem domains into multi-task problem domains. Specifically, there is plan to attempt reinforcement learning of task coordination in multi-task problem domains with multiple performance constraints. Initial results from multi-task problem domains highlight the need for better performance using the fast learning self-organizing neural network.

References

[1] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[2] A. Harati, M. N. Ahmadabadi, and B. N. Araabi, "Knowledge-based multiagent credit assignment: A study on task type and critic information," *IEEE Systems Journal*, vol. 1, no. 1, pp. 55–67, September 2007.

[3] T. Larsen, D. S. Leslie, E. J. Collins, and R. Bogacz, "Posterior weighted reinforcement learning with state uncertainty," *Neural Computation*, vol. 22, no. 5, pp. 1149–1179, May 2010.

[4] M. N. Ahmadabadi and M. Asadpour, "Expertness based cooperative q-learning," *IEEE Transactions on Systems, Man and Cybernetics - Part B*, vol. 32, no. 1, pp. 66–76, February 2002.

[5] H. Firouzi, M. N. Ahmadabadi, B. N. Araabi, S. Amizadeh, M. S. Mirian, and R. Siegwart, "Interactive learning in continuous multimodal space: A bayesian approach to action-based soft partitioning and learning," *IEEE Transactions on Autonomous Mental Development*, vol. 4, no. 2, pp. 124–138, 2012.

[6] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, pp. 408–423, February 2015.

[7] W. M. Hinojosa, S. Nefti, and U. Kaymak, "Systems control with generalized probabilistic fuzzy-reinforcement learning," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 1, pp. 51–64, 2011.

[8] Z. J. Nikolic and K.-S. Fu, "An algorithm for learning without external supervision and its application to learning control systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 3, pp. 304–312, May 1986.

[9] X. Xu, C. Lian, L. Zuo, and H. He, "Kernel-based approximate dynamic programming for real-time online learning control: An experimental study," *IEEE Transactions on Control System Technology*, vol. 22, no. 1, pp. 146–156, 2014.

[10] 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, no. 1, pp. 54–115, January 1987.

[11] A.-H. Tan, "FALCON: A Fusion Architecture for Learning, COgnition, and Navigation," in *Proceedings of the IJCNN*, 2004, pp. 3297–3302.

[12] E. Tuncel, A. Zeid, and S. Kamarthi, "Sovling large scale disassembly line balancing problem with uncertainty using reinforcement learning," *Journal of Intelligent Manufacturing*, vol. 25, pp. 647–659, 2014.

[13] K. Miyazaki, M. Yamamura, and S. Kobayashi, "k-certainty exploration method: An action selector to identify the environment in reinforcement learning," *Artificial Intelligence*, vol. 91, pp. 155–171, 1997.

[14] M. Santos, J. A. M. H., V. Lopez, and G. Botella, "Dyna-$\mathcal{H}$: A heuristic planning reinforcement learning algorithm applied to role-playing game strategy decision systems," *Knowledge-based Systems*, vol. 32, pp. 28–36, 2012.

[15] A.-H. Tan, "Direct code access in self-organizing neural networks for reinforcement learning," in *Proceedings of the IJCAI*, 2007, pp. 1071–1076.

[16] T.-H. Teng, A.-H. Tan, and J. M. Zurada, "Self-organizing neural networks integrating domain knowledge and reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 889–902, 2014.

[17] A.-H. Tan, N. Lu, and X. Dan, "Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evaluative feedback," *IEEE Transactions on Neural Networks*, vol. 19, no. 2, pp. 230–244, 2008.

[18] T.-H. Teng and A.-H. Tan, "Knowledge-based exploration for reinforcement learning in self-organizing neural networks," in *Proceedings of the IAT*, 2012, pp. 332–339.
[19] L. Chrisman, "Reinforcement learning with perceptual aliasing: the perceptual distinctions approach," in *Proceedings of the AAAI*, 1992, pp. 183–188.
[20] K. T. v. Toschanowitz, B. Hammer, and H. Ritter, "Relevance determination in reinforcement learning," in *Proceedings of the ESANN*, 2005, pp. 369–374.
[21] M. R. Endsley, *A Cognitive Approach to Situation Awareness: Theory and Application*, 2004, ch. Situation Awareness: Progress and Directions, pp. 317–341.