4-2013

Delayed insertion and rule effect moderation of domain knowledge for reinforcement learning

Teck-Hou TENG

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

Delayed Insertion and Rule Effect Moderation of Domain Knowledge for Reinforcement Learning

Teck-Hou Teng
School of Computer Engineering
Nanyang Technological University
Center for Computational Intelligence
Email: thteng@ntu.edu.sg

Ah-Hwee Tan
School of Computer Engineering
Nanyang Technological University
Email: asahtan@ntu.edu.sg

***Abstract*—Though not a fundamental pre-requisite to efficient machine learning, insertion of domain knowledge into adaptive virtual agent is nonetheless known to improve learning efficiency and reduce model complexity. Conventionally, domain knowledge is inserted prior to learning. Despite being effective, such approach may not always be feasible. Firstly, the effect of domain knowledge is assumed and can be inaccurate. Also, domain knowledge may not be available prior to learning. In addition, the insertion of domain knowledge can frame learning and hamper the discovery of more effective knowledge. Therefore, this work advances the use of domain knowledge by proposing to delay the insertion and moderate the effect of domain knowledge to reduce the framing effect while still benefiting from the use of domain knowledge. Using a non-trivial pursuit-evasion problem domain, experiments are first conducted to illustrate the impact of domain knowledge with different degrees of truth. The next set of experiments illustrates how delayed insertion of such domain knowledge can impact learning. The final set of experiments is conducted to illustrate how delaying the insertion and moderating the assumed effect of domain knowledge can ensure the robustness and versatility of reinforcement learning.**

I. Introduction

In machine learning, use of domain knowledge is not regarded as a fundamental pre-requisite [9]. It is nonetheless a known approach for improving the learning efficiency of self-organizing neural networks [13], [18]. Domain knowledge improves learning efficiency by framing the learning process. Such framing of the learning process is used to positive outcome in [13], [18]. On the other hand, framing can be detrimental when the degree of truth on its estimated effect to the situations of the domain knowledge diverges significantly from the ground truth. Therefore, taking a less näive approach as seen in [5] and similar to [3], domain knowledge is not assumed to be flawless.

Domain knowledge of human origin can have different effects on the situations and may also have different degrees of truth on the assumed effect to these situations. A *Delayed* Rule Insertion (DRIN) Algorithm is proposed to allow the insertion of domain knowledge either prior to learning or during learning. DRIN incorporates a Rule Effect Moderation (REM) technique proposed to overcome the negative impact of domain knowledge whose assumed effect diverges significantly from the ground truth.

The contributions of this work are illustrated using a non-trivial 1-v-1 Pursuit-Evasion (PE) problem domain. Pursued by a Red entity agent using deterministic strategies, a Blue entity agent uses a self-organizing neural network based on FALCON [10] to learn action policies for evading the Red entity agent. Based on the Adaptive Resonance Theory (ART) [2], FALCON learns incrementally through real-time interactions with the environment. Domain knowledge of different degrees of truth is defined for this PE problem domain. Learning efficiency is measured using the Mission Completion rates and the model complexity is determined using the Node Population.

Three sets of experiments are performed to illustrate the contributions of this work. The first set of experiments is conducted to illustrate the effect of the domain knowledge with different degrees of truth on the learning outcome. The second set of experiments is conducted to illustrate the effect of delaying the insertion of such domain knowledge on the learning outcome using the proposed DRIN Algorithm. The final set of experiments is conducted to illustrate how DRIN is used with REM to overcome the negative impact of using such domain knowledge.

The presentation of this work continues with an updated survey of recent works on the use of domain knowledge in Section II. This is followed by a brief introduction of FALCON in Section III. Details on the use of domain knowledge and the proposed solution to overcome the challenges are presented in Section IV. The Pursuit-Evasion problem domain is introduced briefly in Section V. The experiments and the results are presented in Section VI. This work concludes in Section VII with a brief write-up on the future work.

II. Related Works

Most recent works that use domain knowledge for application development and for improving the performance of plan-based models and self-organizing neural networks are surveyed here. Some of these most recent works include combining the use of ontology-based domain knowledge with context, intelligent planning and behavior informatics to create a multi-agent-based smart home healthy lifestyle assistant system (SHLAS) [21]. The authors had highlighted that the SHLAS is made intelligent using *healthy* domain knowledge.

The ACKTUS architecture is proposed to allow domain experts to incorporate domain knowledge without the need to be proficient in knowledge engineering [5]. Agents implemented using the BDI framework conduct tailored dialogues

with domain experts and other users. Similar to [21], the reliability of domain knowledge is assumed in their work.

Sharing of domain knowledge is also seen as a critical aspect of grid monitoring [8]. Use of small and incomplete amount of domain knowledge was shown leading to significant improvement of a classical planner's performance [1]. Domain knowledge represented as predicates and facts is used in answer set programming to initialize and revise the POMDP belief distributions [20].

Most similar to this work, an alternative approach of handling inaccurate domain knowledge is seen in [3] for plan-based reward shaping. In contrast, this work is based on learning of procedural knowledge. Widely used for agent-based simulations, cognitive agent implemented using FALCON is used to provide context-aware decision support [13]. Adaptive computer-generated force (CGF) is also implemented using FALCON to evolve 1-v-1 air combat maneuvering strategies against another CGF [17] and also the human pilots [16]. In addition, FALCON is known to perform better when compared to other function approximators [11], [15].

III. The Learning Model

A self-organizing neural network model based on FALCON [10] is used for an adaptive virtual agent. By learning multi-dimensional mappings across states, actions and values in an online and incremental manner, FALCON enables reinforcement learning of both value and action policies in real time.

A. Structure and Operating Modes

Seen in Fig. 1, the FALCON network [10] employs a 2-layer architecture, comprising an input-output (IO) layer and a knowledge layer. The IO layer has three input fields, namely 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 three fields of weights $\mathbf{w}_j^{ck}$ for $k = \{1, 2, 3\}$.

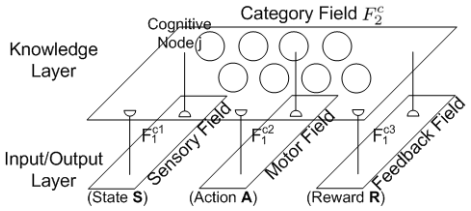

Fig. 1. The FALCON Architecture.

The FALCON network has three modes of operations - INSERT, PERFORM and LEARN. The mode of operation determines how the various parameters should be used. It is made compact using a confidence-based pruning strategy presented in Section III-C. Algorithmically, FALCON is considered to have an inner layer (outlined in Algorithm 1) and an outer layer [15](outlined in Algorithm 2).

B. Temporal Difference Method

Outlined in Algorithm 2, the outer layer of FALCON incorporates Temporal Difference (TD) methods to estimate and learn value function $Q(s, a)$ of state-action pair that indicates the goodness of taking action choice a in state s [12]. On receiving the *ground truth* on the effect of action choice a on state s, a TD formula estimates the Q-value of action choice a on state s. This estimated Q-value is then used as the teaching signal to FALCON to learn the association of state s and action choice a.

Algorithm 1 The Inner FALCON Layer

Require: Activity vectors $\mathbf{x}^{ck}$ and all weight vectors $\mathbf{w}_j^{ck}$ for $k = \{1, 2, 3\}$
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
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**: Derive m_J^{ck} to determine whether resonance is attained using

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

where $\rho^{ck} \in [0, 1]$ are the vigilance parameters
7: **if** vigilance criterion is satisfied **then**
8: *Resonance State* is attained
9: **else**
10: **Match Tracking**: Modify state vigilance ρ^{c1} using

$$\rho^{c1} = \min\{m_J^{ck} + \psi, 1.0\}$$

where ψ is a very small step increment to match function m_J^{ck}
11: **Reset**: $m_J^{ck} = 0.0$
12: **end if**
13: **until** *Resonance State* is attained
14: **if** operating in LEARN/INSERT mode **then**
15: **Template Learning**: modify $\mathbf{w}_J^{ck}$ using

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

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 action choice a
18: **end if**

Algorithm 2 The Outer FALCON Layer

1: Initialize FALCON
2: Sense the environment and formulate a state representation s
3: Apply *Direct Code Access* [11] on Inner FALCON layer to exploit existing knowledge to select action choice a
4: **if** action choice a not identified **then**
5: Explore action space $\mathcal{A}$ using Knowledge-based Exploration [14] strategy to select action choice a
6: **end if**
7: Use action choice a on state s for state s'
8: Obtain ground truth r on the effect of action choice a to state s
9: Estimate the Q-value function $Q^{new}(s, a)$ using $\Delta Q(s, a) = \alpha T D_{err}$
10: **if** action choice a is positive to state s **then**
11: Use updated $Q^{new}(s, a)$ in (2) to adapt Υ
12: **end if**
13: Present $\{\mathbf{S}, \mathbf{A}, \mathbf{R}\}$ to Inner FALCON layer for **Learning**
14: Update the current state $s = s'$
15: Repeat from Step 2 until s is a terminal state

Iterative Value Estimation: A temporal difference method known as Bounded Q-Learning [12] is used iteratively to estimate the value of applying action choice a to state s. The Q-value update function is given by

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

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

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

where $\gamma \in [0,1]$ is the discount parameter and $\max_{a'} Q(s',a')$ is the maximum estimated value of the next state s'. It is notable that in TD-FALCON, the values of $Q(s,a)$ and $\max_{a'} Q(s',a')$ are in turn estimated using the same FALCON network.

C. Knowledge Pruning

Ineffective learned knowledge needs to be pruned for more efficient operation. Therefore, a confidence-based pruning strategy similar to the one seen in [10] is adapted to prune the cognitive nodes that encode such ineffective knowledge.

Specifically, each 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 cognitive node j picked for action selection and updating is reinforced using

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

where η is the reinforcement rate of the confidence level for all cognitive nodes. After each training iteration, the confidence level of all cognitive nodes is decayed using

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

where ζ is the decay rate of the confidence level for all cognitive nodes. At the same time, the age σ_j of cognitive node j is also incremented.

The age attribute σ_j of cognitive node j prevents it from being pruned when $\sigma_j = \sigma_j(0)$, $c_j = c_j(0)$ and $c_j < c^{rec}$ where c^{rec} is the recommended confidence threshold. A cognitive node j is pruned only when $c_j < c^{rec}$ and $\sigma_j \geq \sigma^{old}$ where σ^{old} is the old age threshold.

IV. Use of Domain Knowledge

The modus ponens argument form is used by FALCON for deciding how to respond to the situations. Such argument form involves the use of domain knowledge represented using propositional rules. Specifically, a propositional rule used to represent an unit of domain knowledge is defined as follows.

$$R_j: \text{ IF } \mathbf{X}_j \text{ THEN } \mathbf{Y}_j \text{ EFFECT } p_j$$

where $\mathbf{X}_j$ is the antecedent, $\mathbf{Y}_j$ is the consequent and p_j is the assumed effect of propositional rule R_j. Details on how domain knowledge with different degrees of truth can impact learning and the proposed solution for overcoming the known challenges are presented.

A. Known Challenges

Inserting domain knowledge to initialize the learning model frames the learning process. The aim of such an approach is to promote exploitation of not easily learned knowledge and to also reduce failure rate due to exploration. However, the framing effect of using domain knowledge can hamper the discovery of more effective knowledge.

The conventional approach of inserting domain knowledge to initialize the learning model is not suitable when domain knowledge is not available prior to learning. An ability to insert domain knowledge during the learning process is required. In addition, it may also be desired to delay the insertion of domain knowledge even when it is available prior to learning. This is to allow the learning model some amount of *settling* time. Therefore, an approach for a delayed insertion of domain knowledge is presented in Section IV-C.

The ground truth on the effect of the domain knowledge may deviate significantly from its assumed effect. Such domain knowledge is considered to have different degree of truth on its assumed effect to the situations with respect to the ground truth. The Rule Effect Moderation (REM) technique proposed in Section IV-D is aimed at overcoming the negative impact of such domain knowledge.

B. Domain Knowledge with different degrees of truth

Like in [3], domain knowledge of human origin is recognised to be flawed at times. Therefore, there can be acute shortcomings using such domain knowledge. Therefore, inaccuracy in assuming the effect of the domain knowledge is inherent to such an approach. Specifically, the estimated effect of the domain knowledge to the situations can diverge significantly from the ground truth.

Given an inserted rule R_j recommending action choice a_j for situation s, R_j has an assumed effect p_j such that $p_j \in [0,1]$. The ground truth on the quantitative effect of action choice a_j on situation s is denoted using r_j such that $r_j \in [0,1]$. Using Υ such that $\Upsilon \in [0,1]$ to discriminate the positive and negative rules, the ground-truth variance $\sigma_{gt}(j)$ of rule R_j is defined as

$$\sigma_{gt}(j) = r_j - \Upsilon$$

while its assumed-effect variance $\sigma_{ae}(j)$ is defined as

$$\sigma_{ae}(j) = p_j - \Upsilon$$

Using $\sigma_{gt}(j)$ and $\sigma_{ae}(j)$, the following degrees of truth on rule R_j can be defined.

*Definition 1 (**True-Positive** Rule):* Rule R_j is a true-positive (TP) rule when $\sigma_{gt}(j) \geq 0$ and $\sigma_{ae}(j) \geq 0$

Sample R_{TP}: **IF** EnemyDirection=North
THEN EvadeDirection=South
$p_{tp} = 0.75$, $r_{tp} = 0.65$

*Definition 2 (**True-Negative** Rule):* Rule R_j is a true-negative (TN) rule when $\sigma_{gt}(j) < 0$ and $\sigma_{ae}(j) < 0$

Sample R_{TN}: **IF** EnemyDirection=North
THEN EvadeDirection=North
$p_{tn} = 0.05$, $r_{tn} = 0.15$

*Definition 3 (**False-Positive** Rule):* Rule R_j is a false-positive (FP) rule when $\sigma_{gt}(j) < 0$ and $\sigma_{ae}(j) \geq 0$.

Sample R_{FP}: **IF** EnemyDirection=North
THEN EvadeDirection=North
$p_{fp} = 0.75$, $r_{fp} = 0.65$

*Definition 4 (**False-Negative** Rule):* Rule R_j is a false-negative (FN) rule when $\sigma_{gt}(j) \geq 0$ and $\sigma_{ae}(j) < 0$.

Sample R_{FN}: **IF** EnemyDirection=North
THEN EvadeDirection=South
$p_{fn} = 0.05$, $r_{fn} = 0.65$

The degree of truth of the sample rules are based on $\Upsilon = 0.5$. The TP and TN rules are propositional rules with convergent effects while the FP and FN rules are propositional rules with divergent effects. The concepts of TP, TN, FP and FN are considered for measuring precision and recall probability in [8].

C. Delayed *Insertion of Domain Knowledge*

Conventionally, propositional rules are inserted into virtual agent prior to learning [13], [18]. Such approach is known to be effective for improving learning efficiency. However, such framing of the learning process is positive only when the domain knowledge turns out to be TP or TN. However, it will have an opposite effect when the domain knowledge turns out to be FP or FN.

Therefore, delayed insertion of domain knowledge is explored in this work. The hypothesis here is that delaying the insertion of domain knowledge of unknown degree of truth can help to minimize the framing effect while still allowing the virtual agent to benefit from the inserted domain knowledge. This hypothesis is based on the fact that FALCON begins learning using high Υ, i.e., $\Upsilon \rightarrow 1.0$. During learning, Υ is adapted to better reflect the ground truth necessary for separating the action policies with positive and negative impacts using

$$\Upsilon(t+1) = \min(\nu\Upsilon(t) + (1-\nu)Q^{new}(s,a), \Upsilon(t)) \quad (2)$$

where ν is the adaptation rate of Υ and $Q^{new}(s,a)$ is an updated estimation of the Q-value of cognitive node j with action choice a known to be effective to situation s. Delaying the insertion of domain knowledge allows the adaptation of Υ to be closer to the ground truth of the simulated scenario. Therefore, the *Delayed* Rule Insertion (DRIN) algorithm is proposed to facilitate the investigation of such hypothesis.

Unlike [13], [18] where domain knowledge is inserted prior to learning, explicit indication on when to insert the domain knowledge has to be specified using t_{insert}. Specifically, the DRIN algorithm (outlined in Algorithm 3) inserts domain knowledge $\mathbf{R}$ only when $t \equiv t_{insert}$. The insertion of domain knowledge commences with the selection of a propositional rule R_j from $\mathbf{R}$ for translation.

The translation process involves translating antecedent $\mathbf{X}_j$ into the state vector $\mathbf{S}$, the consequent $\mathbf{Y}_j$ into the action vector $\mathbf{A}_j$ and encoding the moderate effect p'_j as the reward vector $\mathbf{R}_j$. The translated rule R_j represented using the triad tuple $\{\mathbf{S}_j, \mathbf{A}_j, \mathbf{R}_j\}$ is presented to FALCON for learning using the Inner FALCON layer outlined in Algorithm 1 with $\rho^{ck} = 1.0$ for $k = \{1, 2, 3\}$.

Using $\rho^{ck} = 1.0$ for $k = \{1, 2, 3\}$ ensures that only identical set of state, action and reward vectors are grouped into the same cognitive node. Thus, each inserted rule R_j leads to a committed cognitive node encoding the $\{\mathbf{S}_j, \mathbf{A}_j, \mathbf{R}_j\}$ triad tuple as its weight templates. Hence, as many cognitive nodes as the propositional rules can be committed for learning the inserted domain knowledge.

Algorithm 3 The *Delayed* Rule Insertion Algorithm

Require: t_{insert}, domain knowledge $\mathbf{R}$ and FALCON
if $t \equiv t_{insert}$ **then**
  for each propositional rule $R_j \in \mathbf{R}$ **do**
    Translate *antecedent* $\mathbf{X}_j$ as state vector $\mathbf{S}_j$
    Translate *consequent* $\mathbf{Y}_j$ as action vector $\mathbf{A}_j$
    Encode p_j as reward vector $\mathbf{R}_j$
    Insert triad tuple $\{\mathbf{S}_j, \mathbf{A}_j, \mathbf{R}_j\}$ into FALCON using $\rho^{ck} = 1.0$
  end for
end if
return FALCON

D. The Divergent Effect of FP Rules

Delaying the insertion of domain knowledge only allows FALCON to build up its own knowledge base and to adapt Υ using (2) without any effect of framing. Domain knowledge of unknown degree of truth is still inserted into FALCON with the intention of improving the learning efficiency.

The adapted Υ is used as the reward field vigilance criterion ρ^{c3} for the selection of cognitive nodes when FALCON is operating in the *perform* and *learn* mode. Inserted rule R_j with assumed effect $p_j \geq \Upsilon$ has higher probability of being selected. In this sense, committed cognitive nodes that learn TN and FN rules will have lower probability of being selected while the committed cognitive nodes that learn the TP and FP rules have higher probability of being selected.

Selecting committed cognitive nodes with TP rules is expected to be positive to the learning process. However, selecting committed cognitive nodes that learn FP rules can be negative to the learning process. Committed cognitive node j that learns a FP rule R_{FP} can be selected because $p_{fp} \geq \Upsilon$. The assumed effect p_j is learned as the Q-value $Q(s,a)$ of cognitive node j. Using the Bounded Q-Learning method (see (1)), existing learning mechanism of FALCON is capable of correcting such distortion of the assumed effect p_{fp} of the FP rules only after several training iterations.

Algorithm 4 *Delayed* Rule Insertion with Rule Effect Moderation (DRIN-REM)

Require: t_{insert}, domain knowledge $\mathbf{R}$ and FALCON
if $t \equiv t_{insert}$ **then**
  for each propositional rule $R_j \in \mathbf{R}$ **do**
    Translate *antecedent* $\mathbf{X}_j$ as state vector $\mathbf{S}_j$
    Translate *consequent* $\mathbf{Y}_j$ as action vector $\mathbf{A}_j$
    $p'_j = \min(p_j, \Upsilon)$
    Encode p'_j as reward vector $\mathbf{R}_j$
    Insert triad tuple $\{\mathbf{S}_j, \mathbf{A}_j, \mathbf{R}_j\}$ into FALCON using $\rho^{ck} = 1.0$
  end for
end if
return FALCON

Therefore, DRIN is augmented with a Rule Effect Moderation (REM) strategy to overcome the framing effect of the FP rules. Outlined in Algorithm 4, when rule R_{j_1} is inserted at $t = t_{insert}$ such that $t_{insert} > 0$, the DRIN-REM

strategy moderates the assumed effect p_{j_1} with respect to Υ. Specifically, regardless of the underlying degree of truth of the domain knowledge which is unknown at the moment of insertion, the moderated effect of rule R_{j_1} will be the smaller value of either p_{j_1} or Υ. The following hypothesis is formed based on the use of the proposed DRIN-REM strategy.

Hypothesis 1: Applying the DRIN-REM strategy can reject the use of FP rules while accepting the use of TP rules.

Lemma 1 (Rejects Cognitive Node with FP Rule): For cognitive node j_1 with assumed effect p_{j_1} encoding a FP rule, the DRIN-REM strategy can reject the use of cognitive node j_1.

Proof 1: Applying Algorithm 4, Rule R_{j_1} with assumed effect p_{j_1} is inserted as cognitive node j_1 at $t = t_{insert}$ and using $p'_{j_1} = \min(p_{j_1}, \Upsilon)$.

At $t = t_1$ such that $t_1 > t_{insert}$, cognitive node j_1 is selected to recommend action choice a_{j_1} for situation s_1.

At $t = t_2$ such that $t_2 = t_1 + 1$, an immediate reward r_{j_1} is returned as the ground truth of the effect action choice a_{j_1} has on situation s_1 such that action a_{j_1} is *negative* to situation s_1.

Using (1), Q-value of cognitive node j_1 is revised such that $Q^{new}(s_1, a_{j_1}) < \rho^{c3}$ and $\rho^{c3} \equiv \Upsilon$

$\therefore$ at $t = t_3$ such that $t_3 > t_2$ and when $J = j_1$, cognitive node j_1 is rejected $\because m^{c3}_{J=j_1} < \rho^{c3}\square$

Lemma 2 (Accepts Cognitive Node with TP rule): For cognitive node j_2 with assumed effect p_{j_2} encoding a TP rule, the DRIN-REM strategy can accept the use of cognitive node j_2.

Proof 2: Applying Algorithm 4, Rule R_{j_2} with assumed effect p_{j_2} is inserted as cognitive node j_2 at $t = t_{insert}$ and using $p'_{j_2} = \min(p_{j_2}, \Upsilon)$.

At $t = t_1$ such that $t_1 > t_{insert}$, cognitive node j_2 is selected to recommend action choice a_{j_2} for situation s_2.

At $t = t_2$ such that $t_2 = t_1 + 1$, an immediate reward r_{j_2} is returned as the ground truth of the effect action choice a_{j_2} has on situation s_2 such that action choice a_{j_2} is *positive* to situation s_2.

Using (1), Q-value of cognitive node j_2 is revised such that $Q^{new}(s_2, a_{j_2}) \geq \Upsilon$

Knowing a_{j_2} is positive to s_2, $Q^{new}(s_2, a_{j_2})$ is used to updated Υ using (2) and $\rho^{c3} \equiv \Upsilon$

$\therefore$ at $t = t_3$ such that $t_3 > t_2$ and $J = j_2$, cognitive node j_2 is accepted $\because m^{c3}_{J=j_2} \geq \rho^{c3}\square$

Corollary 1: Given that Lemma 1 and Lemma 2 are true, Hypothesis 1 is satisfied using the DRIN-REM strategy.

V. The Pursuit-Evasion Problem Domain

The PE problem domain is a popular choice in the field of game theory [6] as well as machine learning [7], [17]. For the required level of complexity, the same non-trivial PE problem domain used in [14] is used. Therefore, only a brief introduction of the problem domain is provided.

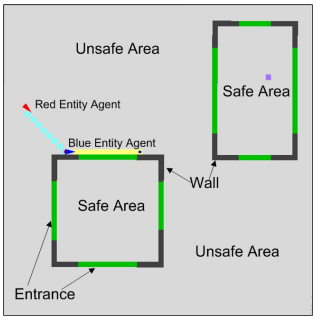

Fig. 2. The $2D$ Grid-Based Environment.

Two virtual agents known as the Blue entity agent and the Red entity agent are seen in Fig. 2. The Red entity agent is hostile towards the Blue entity agent. The two-dimensional environment has two safe areas where the Blue entity agent will be safe from the Red entity agent. Both entity agents are constantly moving in this virtual environment.

The Red entity agent is tasked to search for the Blue entity agent while the Blue entity agent is tasked with a mission of searching the areas. The Red entity agent eliminates the Blue entity agent by contacting it while the Blue entity agent cannot retaliate. In an encounter, the Red entity agent pursues the Blue entity agent using deterministic pursuits strategies while the Blue entity agent learns evasive strategies to improve on its chance of evading the Red entity agent. The state space forms the primary source of information and the action space contains the available action choices for both entity agents.

A. The State Space

A Situation-Awareness Model [4] is used for knowing the operating environment. It is comprised of the *Perception* layer, the *Comprehension* layer and the *Projection* layer. Eight types of multi-valued attributes based on the information of the *enemy* and the *terrain* giving around 3.2876×10^4 possible situations are defined.

Four types of attributes comprising of the *Enemy-Direction*, *SafeArea-Direction*, *Enemy-Orientation* and *Adjacent-Location* give a total of 90 possible combinations to make up the *Perception* layer. Three other types of attributes comprising of *Enemy-Location*, *Enemy-Proximity* and *Traversability* giving a total of 3.2778×10^4 possible combinations makes up the *Comprehension* layer. The *Projection* layer is comprised of a single type of attribute known as the *projection of threat* which gives a total of 8 possible combinations for a single adversary scenario.

B. The Action Space

The Blue entity agent evades the pursuit of the Red entity agent by moving in a particular compass direction. Therefore, the action space is comprised of the eight compass directions - north, northeast, east, southeast, south, southwest, west and northwest. Therefore, the effect of moving in a particular evade direction to the situation is learned and may be exploited for subsequent decision-making cycles.

C. The Reward Space

A number of sensory information is used to quantify the *ground truth* r on the effect of the action choices to the

situations. Agent's proximity to Adversary, agent's orientation with respect to the direction of adversary, spaciousness of selected destination, agent's proximity to safe area, presence of obstacle in selected destination and attacked by adversary are included as reward attributes in this Pursuit-Evasion problem domain. Specifically, trends of these reward attributes are used to derive the ground truth of action choice a on situation s.

VI. Experimental Results

Several experiments are conducted to gradually lead to the main contributions of this work. Experiments are presented in Section VI-A to first establish the effects of the domain knowledge with different degrees of truth on the learning outcome. This is followed by the experiments in Section VI-B to illustrate the DRIN algorithm using $t_{insert} = 90$ and $\Upsilon_0 = 0.475$. Results from the experiments conducted to illustrate the DRIN-REM strategy are then presented in Section VI-C.

TABLE I. The Controlled Parameters

FALCON Parameters for $k = \{1, 2, 3\}$	
Choice Parameters α^{ck}	$\{0.1, 0.1, 0.1\}$
Contribution Parameters γ^{ck}	$\{0.5, 0.5, 0.0\}$
Learning Rates β^{ck}	$\{1.0, 1.0, 1.0\}$
Perform/Learn Vigilance $\rho^{ck}_{p/l}$	$\{0.0, 0.0/1.0, \Upsilon\}$
TD Learning Parameters	
Learning Rate α	0.5
Initial Q-Value	0.5
Discount Factor γ	0.1
Pruning Parameters	
Confidence decay rate ζ	0.003
Confidence reinforcement rate η	0.05
Old age σ^{old}	20 iterations

Four sets of rules comprising of 128 propositional rules each were used for the experiments. Each experiment was conducted for 500 training iterations. Each set of experimental results was aggregated using 20 runs of the same configuration. By further averaging every 40 data points, plots of the experimental results were produced using just 12 data points to present the general trend of each configuration. With the exception of the naïve response (RandomResponse) configuration, all other configurations were implemented using FALCON with the controlled parameters seen in Table I. The same Pruning and Knowledge-based Exploration [14] strategies were used for all configurations.

A. Domain Knowledge with Different Degrees of Truth

Results from experiments conducted using propositional rules with four different degrees of truth (see Section IV-B) on its estimated effect are first presented here. Using Algorithm 3, domain knowledge is inserted into FALCON using $t_{insert} = 0$ (DirectInsert-*). Comparisons are made with the naïve approach (RandomResponse), FALCON without domain knowledge (LearningOnly) and the use of just the domain knowledge (*-RulesOnly).

From the Mission Completion plots seen in Fig. 3, the FN and TN rules appear to have similar effect on the Mission Completion rates while the TP rules have slightly better Mission Completion rates than the FN and TN rules. Of the four sets of domain knowledge, the FP rules have the worst effect on the Mission Completion rates. The use of the inserted

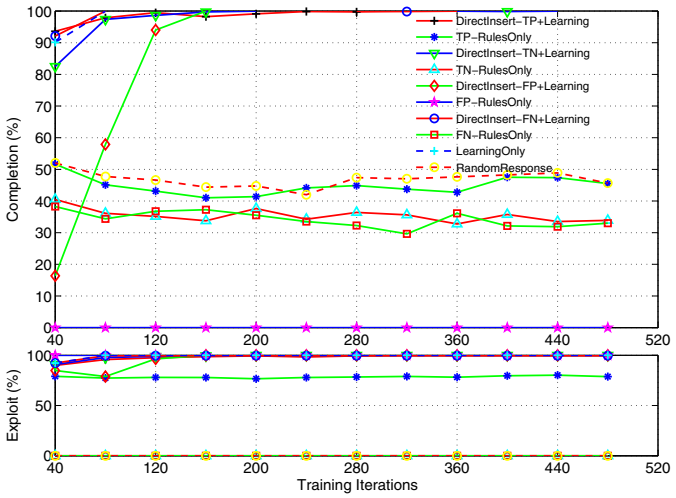

Fig. 3. Comparison of Mission Completion rates using domain knowledge with different degrees of truth

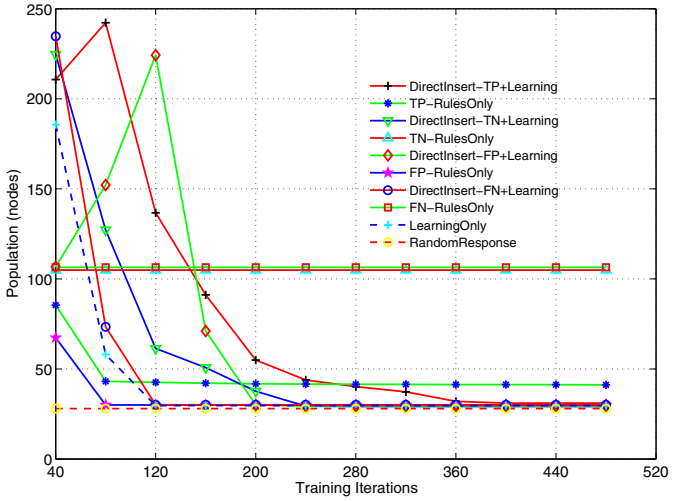

Fig. 4. Comparison of Node Population using domain knowledge with different degrees of truth

FP rules is confirmed by the 100% exploitation rates seen at the bottom plot of Fig. 3. Direct insertion of TP, FN and TN rules into FALCON are seen allowing higher Mission Completion rates than when FALCON begins learning without domain knowledge. In contrast, as seen in Fig. 3, the insertion of FP rules has significant negative impact on the Mission Completion rates.

The number of rules used for the experiments can be seen in the node population of the TN-RulesOnly and the FN-RulesOnly configurations in Fig. 4. Also seen in Fig. 4, the insertion of TP and FP rules into FALCON has led to the spiking of the node population. In contrast, the insertion of TN and FN rules is not seen having similar effect on the node population. In addition, FALCONs inserted with the TN, FN and FP rules saturate to lower node population than FALCON inserted with the TP rules.

B. Delayed Insertion of Domain Knowledge

In this section, experiments are conducted to determine whether the delayed insertion ($t_{insert} = 90$) of (TP, TN, FP and FN) domain knowledge can reduce the framing effect illustrated in Section VI-A. Comparisons are made with the direct

insertion ($t_{insert} = 0$) of domain knowledge. Specifically, this is an illustration of the DRIN strategy outlined in Algorithm 3.

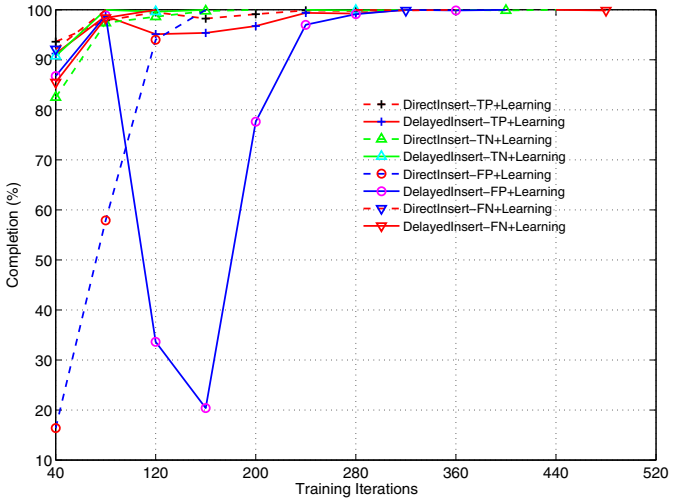

Fig. 5. Comparison of Mission Completion rates using delayed insertion of domain knowledge

From Fig. 5, compared to the direct insertion of TN rules, delaying the insertion of TN rules appears to have positive impact on the Mission Completion rates. This is due to the absence of the framing effect of the TN rules. In contrast, lower Mission Completion rates are seen for the delayed insertion of the FN rules. However, there is no lowering of the Mission Completion rates after the delayed insertion of the FN rules. Compared to the direct insertion of TP rules, a slight lowering of the Mission Completion rates is seen for the delayed insertion of the TP rules. More notably, delaying the insertion of FP rules does not reduce its negative impact at all. As seen in Fig. 5, the negative impact is seen right after the insertion of the FP rules.

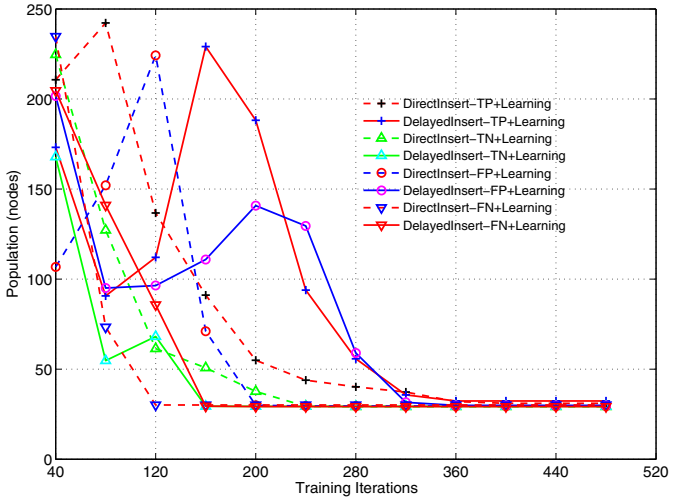

Fig. 6. Comparison of Node Population using delayed insertion of domain knowledge

Similar spiking effects of the node population from the delayed insertion of TP and FP rules are seen in Fig. 6. In contrast, the delayed insertion of FN rules leads to a slight delay in the reduction of the node population to a similar level as the DirectInsert-FN+Learning configuration. In comparison to the direct insertion of TN rules, the delayed insertion of the same rules leads to small spike in the node population. It is seen falling to similar node population earlier than when the TN rules are inserted prior to learning. Despite the insertion of domain knowledge with different degrees of truth, all versions of FALCON saturate to node population below that of the LearningOnly configuration.

C. Delayed Insertion with Rule Effect Moderation

The recovery in the Mission Completion rates from the (direct or delayed) inserted FP rules in Fig. 5 illustrates the effort required to *overcome* the framing effect. In view of such observations, the DRIN-REM strategy is proposed to speed up the recovery process while still allowing FALCON to benefit from the use of TP rules. Therefore, experiments are conducted to illustrate the impact of the DRIN-REM strategy on the use of TP and FP rules for when $t_{insert} = 0$ and $t_{insert} = 90$.

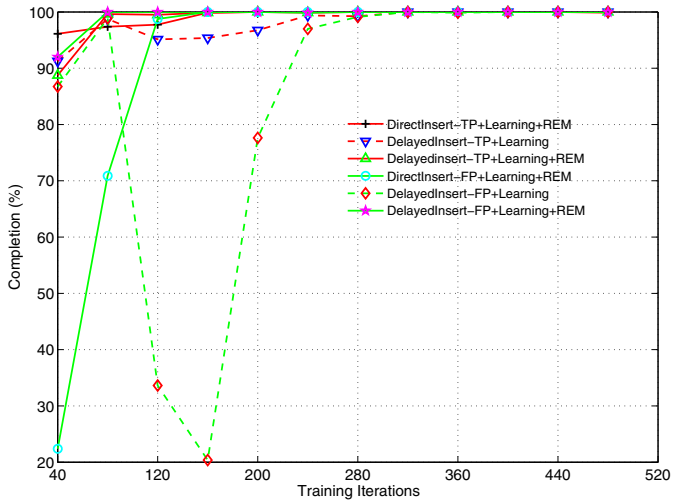

Fig. 7. Comparison of Mission Completion rates of FALCON integrated with REM technique and inserted with TP and FP rules

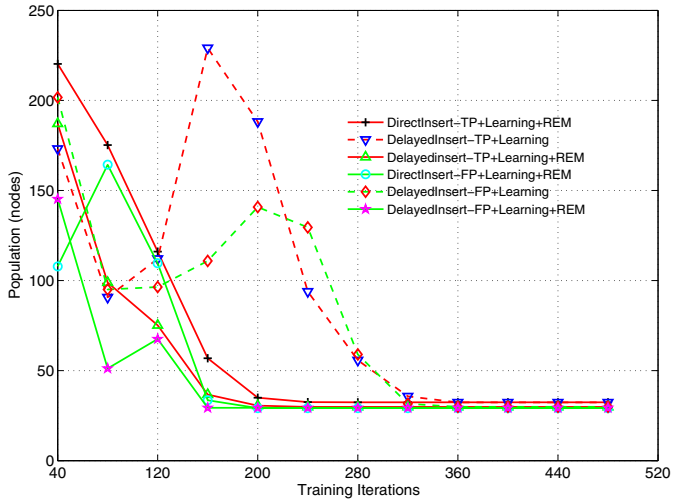

Fig. 8. Comparison of Node Population of FALCON integrated with REM technique and inserted with TP and FP rules

It is seen in Fig. 7 that using DRIN-REM with $t_{insert} = 90$ to insert the TP rules is effective in eliminating the slight dip of the Mission Completion rates seen in just delaying the insertion of the TP rules. More notably, the sharp decline in the Mission Completion rates seen for the delayed insertion of the FP rules

is no longer evident when DRIN-REM is used. As important to note is using DRIN-REM with $t_{insert} = 0$ to insert FP rules does not overcome its negative impact. Therefore, only by using the DRIN-REM strategy with $t_{insert} \gg 0$ to insert the domain knowledge can the negative impact of the FP rules be overcome while still benefiting from the use of the TP rules.

From Fig. 8, in comparison to the spikes in the node population seen for delayed insertion of the TP (DelayedInsert-TP+Learning) and FP (DelayedInsert-FP+Learning) rules, using DRIN-REM is more effective in controlling the node population of FALCON inserted with these two types of domain knowledge (see DelayedInsert-TP+Learning+REM and DelayedInsert-FP+Learning+REM). Specifically, there is only a slight increase in node population for the use of DRIN-REM on the FP rules and no observable increase in node population when DRIN-REM is used on the TP rules. Over time, all versions of FALCON saturate to similar levels of node population.

VII. Conclusion

The use of domain knowledge is advanced by including considerations for domain knowledge with different degrees of truth on its estimated effect to the situations. This is an inevitable shortcoming arising from the use of domain knowledge of human origin. In addition, domain knowledge cannot always be expected to be available prior to learning. It is also recognized that the insertion of domain knowledge into the adaptive virtual agent may frame the learning process and hampers the discovery of more effective knowledge. Therefore, this work is seen filling an important knowledge gap in the use of domain knowledge to improve learning efficiency and reduce model complexity.

Firstly, we establishes that domain knowledge can exist in one of the following degrees of truth - true-positive (TP), true-negative (TN), false-positive (FP) and false-negative (FN) - of the domain knowledge. To respond effectively to such rules and address the above-mentioned issues, we proposed the DRIN algorithm to insert domain knowledge using $t_{insert} \geq 0$. In addition, the estimated effect of the inserted domain knowledge has to be modulated using the proposed REM strategy. An analytical presentation on how the combined use of DRIN and REM strategies can handle domain knowledge with different degrees of truth in a robust manner to improve learning efficiency is also provided.

Using a non-trivial Pursuit-Evasion problem domain, the first set of experimental results illustrates the use of domain knowledge with different degrees of truth on the learning outcome. The second set of experimental results illustrates the use of the DRIN Algorithm with $t_{insert} = 90$ to insert the same sets of domain knowledge. The third set of experimental results illustrates the use of the DRIN-REM strategy with $t_{insert} = 90$ on selected sets of domain knowledge. Observations made on the third set of experimental results confirm the efficacy of the DRIN-REM strategy for addressing the stated challenges pertaining to the use of domain knowledge.

As seen in [19], machine learning is yet to be fully explored for many other forms of knowledge representation. Therefore, we aim to follow up this work with an investigation on how the proposed DRIN-REM strategies can be applied to different forms of argument and knowledge representation schemes. Given that the complexity of the knowledge learned is in direct correlation to the complexity of the problem domain, it is also necessary to identify problem domains with the right amount of complexity to explore machine learning based on more complex knowledge representation schemes.

References

[1] R. Alford, U. Kuter, and D. S. Nau. Translating HTNs to PDDL: A small amount of domain knowledge can go a long way. In *Proceedings of the IJCAI*, pages 1629–1634, 2009.

[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*, pages 54–115, 1987.

[3] K. Efthymiadis, S. Devlin, and D. Kudenko. Overcoming incorrect knowledge in plan-based reward shaping. In *Proceedings of the AAMAS*, pages 65–72, 2012.

[4] M. R. Endsley. Designing for situation awareness in complex systems. In *Proceedings of the* 2^{nd} *International Workshop on Symbiosis of Humans, Artefacts and Environment*, pages 1–14, 2001.

[5] H. Lindgren, F. Yekeh, C.-L. Yan, and J. Baskar. Agent-supported assessment for personalized ambient assisted living. In *Proceedings of the AAMAS*, pages 41–50, 2012.

[6] E. Raboin, U. Kuter, and D. S. Nau. Generating strategies for multi-agent pursuit-evasion games in partially observable euclidean space. In *Proceedings of the AAMAS*, pages 134–152, June 2012.

[7] E. Y. Rodin. A Pursuit-Evasion Bibliography. *Computers & Mathematics with Applications*, 13(1-3):275–340, 1987.

[8] A. J. R. Stoter, S. Dalmolen, and W. Mulder. Agent-mining of grid log-files: A case study. In *Proceedings of the AAMAS*, pages 107–118, 2012.

[9] R. Sun and C. Sessions. Learning plans without a priori knowledge. *Adaptive Behavior*, 8(3/4):225–254, 2000.

[10] A.-H. Tan. FALCON: A Fusion Architecture for Learning, Cognition, and Navigation. In *Proceedings of the IJCNN*, pages 3297–3302, 2004.

[11] A.-H. Tan. Direct Code Access in Self-Organizing Neural Networks for Reinforcement Learning. In *Proceedings of the IJCAI*, pages 1071–1076, January 2007.

[12] 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*, 19(2):230–244, February 2008.

[13] T.-H. Teng and A.-H. Tan. Cognitive agents integrating rules and reinforcement learning for context-aware decision support. In *Proceedings of the IAT*, pages 318–321, December 2008.

[14] T.-H. Teng and A.-H. Tan. Knowledge-based exploration for reinforcement learning in self-organizing neural networks. In *Proceedings of the IAT*, pages 332–339, December 2012.

[15] T.-H. Teng and A.-H. Tan. Integrating domain knowledge and reinforcement learning: A self-organizing neural network approach. *IEEE Transaction on Systems, Man and Cybernetics - Part B*, 2013. under review.

[16] T.-H. Teng, A.-H. Tan, W.-S. Ong, and K.-L. Lip. Adaptive CGF for Pilots training in Air Combat Simulation. In *Proceedings of the* 15^{th} *International Conference on Information Fusion*, pages 2263–2270, July 2012.

[17] T.-H. Teng, A.-H. Tan, Y.-S. Tan, and A. Yeo. Self-organizing Neural Networks for Learning Air Combat Maneuvers. In *Proceedings of the IJCNN*, pages 2859–2866, June 2012.

[18] T.-H. Teng, Z.-M. Tan, and A.-H. Tan. Self-organizing neural models integrating rules and reinforcement learning. In *Proceedings of the IJCNN*, pages 3770–3777, June 2008.

[19] F. van Harmelen, V. Lifschitz, and B. Porter, editors. *Handbook of Knowledge Representation*. Elsevier, 2008.

[20] S.-Q. Zhang, S. Bao, and M. Sridharan. ASP-POMDP: Integrating non-monotonic logical reasoning and probabilistic planning on mobile robots. In *Proceedings of the AAMAS*, pages 185–201, 2012.

[21] X.-H. Zhu, Y.-X. Yu, Y.-M. Ou, D. Luo, C.-Q. Zhang, and J.-H. Chen. System modeling of a smart-home healthly lifestyle assistant. In *Proceedings of the AAMAS*, pages 67–80, 2012.