Integrating Domain Familiarity and Reinforcement Learning For Artificial Neural Network

*1Ms. Nisha S., *2 Mrs. Sangeetha Lakshmi G.., *3 Ms. Raja Lakshmi N.S.,

*1 M.Phil Research Scholar, Department of Computer Science DKM College for Women (Autonomous), Vellore, TamilNadu, India.

*2Assisant Professor, Department of Computer Science DKM College for Women (Autonomous), Vellore, TamilNadu, India.

*3Assisant Professor, Department of Computer Science DKM College for women (Autonomous), Vellore TamilNadu, India.

---***---

Abstract - ***Knowledge integration and learning are two key issues in designing knowledge-based intelligent systems. This talk will present a family of self-organizing neural networks, collectively known as fusion Adaptive Resonance Theory (fusion ART), for building knowledge-based intelligent systems with real-time learning capabilities. By extending the original Adaptive Resonance Theory (ART) models consisting of a single pattern field into a multi-channel architecture, fusion ART unifies a number of important neural network designs developed over the past decades. Based on a universal set of neural encoding and adaptation principles, fusion AT supports a myriad of learning paradigms, notably unsupervised learning, supervised learning, and reinforcement learning. TD-FALCON is a self-organizing neural network that incorporates Temporal Difference (TD) methods for reinforcement learning. Despite the advantages of fast and stable learning, TD-FALCON still relies on an iterative process to evaluate each available action in a decision cycle. To remove this deficiency, this paper presents a direct code access procedure whereby TD-FALCON conducts instantaneous searches for cognitive nodes that match with the current states and at the same time provide maximal reward values. Our comparative experiments show that TD-FALCON with direct code access produces comparable performance with the original TD-FALCON while improving significantly in computation efficiency and network complexity.***

Key Words: Adaptive Resonance Theory, Temporal Difference, FALCON, ANN, intelligent system

I. INTRODUCTION

Reinforcement learning [Sutton and Barto, 1998] is an interaction based paradigm wherein an autonomous agent learns to adjust its behavior according to feedback from the environment. Classical solutions to the reinforcement learning problem generally involve learning one or more of the following mappings, the first linking a given state to a desired action (action policy), and the second associating a pair of state and action to a utility value (value function), using temporal difference methods, such as SARSA [Rummery and Niran-jan, 1994] and Q-learning [Watkins and Dayan, 1992].The problem of the original formulation is that mappings must be learned for each and every possible state or each and every possible pair of state and action. This causes a scalability issue for continuous and/or very large state and action spaces.

Neural networks and reinforcement learning have had an intertwining relationship [Kaelbling et al., 1996]. In particular, multi-layer feed-forward neural networks, also known as multi-layer perceptron (MLP), have been used extensively in many reinforcement learning system and applications [Ack-ley and Littman, 1990; Sutton, 1984]. Under the recent thread of research in Approximate Dynamic Programming (ADP) [Si et al., 2004], MLP and gradient descent back propagation (BP) learning algorithms are commonly used to learn an approximation of the value function from the state and action spaces (value policy) and/or an approximation of the action function from the state space (action policy). MLP and BP however are not designed for online incremental learning as they typically require an iterative learning process. In addition, there is an issue of instability in the sense that learning of new patterns may erode the previously learned knowledge. Consequently, the resultant reinforcement learning systems may not be able to learn and operate in real time.

Instead of learning value functions and action policies, self-organizing neural networks, such as Self-Organizing Map (SOM), are typically used for the representation and generalization of continuous state and action spaces [Smith, 2002]. The state and action clusters are then used as the entries in a Q-value table implemented separately. Using a localized representation, SOM has the advantage of more stable learning, compared with back propagation networks. However, SOM remains an iterative learning system, requiring many rounds to learn the compressed representation of the state and action patterns. In addition, as presents a neural architecture called FALCON (Fusion Architecture for Learning, Cognition, and Navigation), that learns multi-channel mappings simultaneously across multi-modal input patterns, involving states, actions, and rewards, in an online and incremental manner. For handling problems with delayed evaluative feedback (reward signal), a variant of FALCON, known as TD-FALCON [Tan and Xiao, 2005], learns the value functions of the state-action space estimated through Temporal Difference (TD) algorithms. Compared with other ART-based systems described by Ueda et. al and Ninomiya, TD-FALCON presents a truly integrated solution in the sense that there is no implementation of a separate reinforcement learning module or Q-value table.

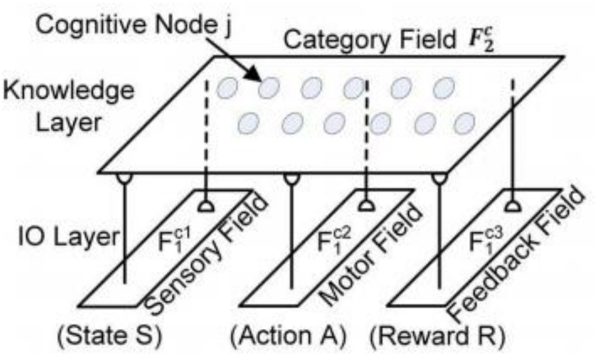

Fig1 : FALCON Architecture

Although TD-FALCON provides a promising approach, its action selection procedure contains an inherent limitation. Specifically, TD-FALCON selects an action by weighting the consequence of performing each and every possible action in a given state. Besides that the action selection process is inefficient with a large number of actions, the numerating step also assumes a finite set of actions, rendering it inapplicable to continuous action space. In view of this deficiency, this paper presents a direct code access procedure by which TD-FALCON can perform instantaneous searches for cognitive nodes that match with the current states and at the same time provide the highest reward values. Besides that the algorithm is much more natural and efficient, TD-FALCON can now operate with both continuous state and action spaces. Our comparative experiments based on a minefield navigation task show that TD-FALCON with direct code access produces comparable performance with the original TD-FALCON system while improving vastly in terms of computation efficiency as well as network complexity.

The rest of the paper is organized as follows. For completeness, section 2 presents a summary of the FALCON architecture and the associated learning and prediction algorithms. Section 3 presents the new TD-FALCON algorithm with the direct code access procedure. Section 4 introduces the minefield navigation simulation task and presents the experimental results. The final section concludes and provides a brief discussion of future work.

II. RELATED WORK

FALCON employs a 3-channel architecture comprising a cognitive field F2c and three input fields, namely a sensory field Ff1 for representing current states, an action field F_{f2} for representing actions, and a reward field Ff3 for representing reinforcement values. The generic network dynamics of FALCON, based on fuzzy ART operations [Carpenter et al., 1991], is described below. (*a1,a2,...,am*) denote the action vector, where a% G [0,1] indicates a possible action i.Let R = (r, f) denote the reward vector, where r G [0,1] is the reward signal value and f (the complement of r)is given by f =1 — r. Complement coding serves to normalize the magnitude of the input vectors and has been found effective in ART systems in preventing the code proliferation problem. As all input values of FALCON are assumed to be bounded between 0 and 1, normalization is necessary if the original values are not in the range of [0, 1].

Activity vectors: Let xck denote the Ff* activity vector for k = 1,..., 3.Let yc denote the F^2_c activity vector. Weight vectors: Let wck denote the weight vector associated with the jth node in F^2_c for learning the input patterns in Fffc for k = 1,..., 3. Initially, F^2_c contains only one uncommitted node and its weight vectors contain all 1's. When an uncommitted node is selected to learn an association, it becomes committed.

2.1 Parameters: The FALCON's dynamics is determined by choice parameters ack > 0 for k = 1,..., 3; learning rate parameters /3_{ck} G [0,1] for k = 1,..., 3; contribution parameters Y_{ck} G [0,1] for k = 1,..., 3 where k=1 Yck = 1; and vigilance parameters p_{ck} G [0,1] for k = 1,..., 3. 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. Specifically, given the activity vectors x_{c1}, x_{c2} and x_{c3} (in the input fields F_{f1}, F_{fs} and F_{f3} respectively), for each F_{2c} node j, the choice function T_j is computed as follows:

Where the fuzzy AND operation A is defined by (p $_A$ q)% = min(pj,qj), and the norm |.| is defined by |p|= J2% P% for vectors p and q. In essence, the choice function Tj computes the similarity of the activity vectors with their respective weight vectors of the F2c node j with respect to the norm of individual weight vectors.

2.2 Code competition: A code competition process follows under which the F2c node with the highest choice function value is identified. The winner is indexed at J where

$$T_J = max\{T_j^c : for\ all\ F^2{}_c\ node\ j\}.$$

(3) When a category choice is made at node J, yJ = 1;and yc = 0 for all *j = J*. This indicates a winner-take-all strategy. 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 activity patterns. Specifically, resonance occurs if for each channel k,the match function mJk of the chosen code J meets its vigilance criterion:

2.3 Input vectors: Let S = *(s1,s2,...,sn)* denote the state vector, where s% G [0,1] indicates the sensory input i.*Let A* = The match function computes the similarity of the activity and weight vectors with respect to the norm of the activity vectors. Together, the choice and match functions work cooperatively to achieve stable coding and maximize code compression.

III. PREVIOUS METHOD

The research described in several of the elements proposed as part of this thesis: use of RL, learning plans on BDI systems, and extraction of BDI plans from MDPs and POMDPs. A more detailed overview of previous research whose elements come closest to the work done as part of this paper

3.1 Research on BDI and Learning

An extensive review of the existing literature in RL and BDI did not uncover any research that made use of RL to learn BDI plans without relying on a-priori knowledge. As discussed in the Introduction on Chapter 1, the lack of learning capabilities for BDI systems was recognized as far back as Researchers tackled this by augmenting the BDI framework with various learning frameworks including decision trees, self-organizing neural networks, hybrid-architectures using low level learners, and met plans for plan hypothesis abduction and plan medications. Other relevant research tackled the use of a-priori knowledge, previously learned knowledge and the learning of plans without apriority knowledge on planning systems, and the integration of learning, planning and execution. These studies were, however, not investigated in relation to BDI systems.

Target: plan learning and plan improvement

Model: hybrid, inductive

Learning Element: self-organizing neural network (FALCON), hypothesis abduction

Goal: plan learning via plan extraction using PGS and plan improvement using hypothesis abduction

The goal of this thesis is to use reinforcement learning to generate plans without a-priori knowledge on BDI agent systems. The key idea is that the result of reinforcement learning is a policy or policies in the general case. Since policies map states to actions, the policies can then be used as input to generate plans in BDI agents systems. The approach can then be summarized as a two step process:

1. Use reinforcement learning as the learning module.

2. Use policies learned as input to generate BDI plans.

None of the previous work combines the elements of RL for plan generation on BDI agent systems. The problem selected for study in this thesis is justice by this lack of research exploring the generation of plans in BDI systems using reinforcement learning that does not rely on a priori knowledge.

IV. PROPOSED METHOD

4.1 3TD-FALCON

TD-FALCON incorporates Temporal Difference (TD) methods to estimate and learn value functions of action-state 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 are then used in the action selection mechanism, also known as the policy, to select an action with the maximal payoff. The original TD-FALCON algorithm proposed by Tan and Xiao (2005) selects an action with the maximal Q-value in a state s by enumerating and evaluating each available action a by presenting the corresponding state and action vectors S and A to FALCON. The TD-FALCON presented in this paper re-places the action enumeration step with a direct code access procedure, Given the current state s, TD-FALCON first decides between exploration and exploitation by following an action selection policy. For exploration, a random action is picked. For exploitation, TD-FALCON searches for optimal action through a direct code access procedure. Upon receiving a feedback from the environment after performing the action, a TD formula is used to compute a new estimate of the Q value of performing the chosen action in the current state. The new Q value is then used as the teaching signal for TD-FALCON to learn the association of the current state and the chosen action to the estimated Q value. The details of the action selection policy, the direct code access procedure, and the Temporal Difference equation are elaborated below.

4.2 RL Training

A detailed survey of RL and automated planning system is presented by Partalas. Partalas argues that \there is a close relationship between those two areas as they both deal with the process of guiding an agent, situated in a dynamic environment, in order to achieve a set of predeened goals." Because RL combines planning and learning the distinctions above blur in practice if not in theory. Since a relevant part of the research for this study will be the selection of feasible RL approaches to generate plans for BDI systems, only a quick summary of the research detailed by Partalas will be given here except in cases where research is directly related to the problem for this study. Partalas describes the possible approaches to combining planning with RL as:

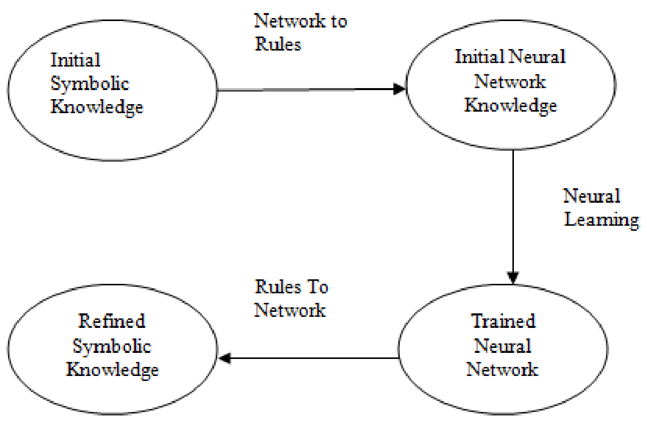
Initial Symbolic Knowledge
Network to Rules
Initial Neural Network Knowledge
Neural Learning
Rules To Network
Refined Symbolic Knowledge
Trained Neural Network

Fig2 : Proposed Method

Our approach to understanding trained networks uses the three-link chain illustrated by the first link inserts domain knowledge, which need be neither complete nor correct, into a neural network using KBANN. The second link trains the KNN using a set of classified training examples and standard neural learning methods. The final link extracts rules from trained KNNs. Rule extraction is an extremely difficult task for arbitrarily-configured Networks, but is somewhat less daunting for KNNs due to their initial comprehensibility. Our method takes advantage of this property to efficiently extract rules from trained KNNs.

Significantly, when evaluated in terms of the ability to correctly classify examples not seen during training, our method produces rules that are equal or superior to the networks from which they came Moreover, the extracted rules are superior to the rules resulting from methods that act directly on the rules (rather than their re-representation as a neural network). Also, our method is superior to the most widely-published algorithm for the extraction of rules from general neural networks.

V. SYSTEM IMPLEMENTATION

5.1 Rule Extraction

This section presents a set of experiments designed to determine the relative strengths and weaknesses of the two rule-extraction methods described above. Rule-extraction techniques are compared using two measures: quality, which is measured both by the accuracy of the rules; and comprehensibility which is approximated by analysis of extracted rule sets.

5.2 Testing Methodology

We use repeated 10-fold cross-validation for testing learning on two tasks from molecular biology: promoter recognition and splice-junction determination. Networks are trained using the cross-entropy. Following Hinton's suggestion for improved network interpretability, all weights "decay" gently during training.

5.3 Accuracy of Extracted Rules

It plots percentage of errors on the testing and training sets, averaged over eleven repetitions of 10-fold cross-validation, for both the promoter and splice-junction tasks. For comparison, Figure 4 includes the accuracy of the trained KNNs prior to rule extraction (the bars labeled Network"). Also included in is the accuracy of the EITHER system, an "all symbolic" method for the empirical adaptation of rules

5.4 Incorporating TD Method

For learning from delayed evaluative feedback signals, the value function Q(s, a) of state-action pairs is estimated using TD method outlined in Algorithm2. At time t, lines 1–9 of Algorithm 2 show FALCON operating in PERFORM mode to select action choice a either by exploration or by exploitation. At time t + 1, lines 10–13 of Algorithm 2 show that FALCON operating in LEARN mode uses reward r from the environment on action choice a to estimate the value function Q(s, a).

: A TD method known as bounde**1) Iterative Value Estimation**d Q-learning is iteratively used to estimate the value of applying action choice a to state s. The Q-value update function is given by

$$Q_{new}(s, a) = Q(s, a) + \alpha T_{Derr}(1 - Q(s, a))$$

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

$$T_{Derr} = r + \gamma \max a\ Q(s',a') - Q(s, a)$$

Where $\gamma \in [0, 1]$ is the discount parameter and the maxa Q(s', a') is the maximum estimated value of the next state s'. The estimated Q-value $Q_{new}(s, a)$ is used as a teaching signal to learn the association of state s and action choice a. It is notable that in TD-FALCON, the values of

$$Q(s, a) \text{ and } \max_a Q(s', a')$$

are in turn estimated using the same FALCON network.

5.5 ALGORITHM IMPLEMENTATION

Algorithm 1

TD-FALCON Algorithm

```
1: Initialize the FALCON network.
2: Sense the environment and formulate a state vector S based on the current state s.
3: Following an action selection policy, first make a choice between exploration and exploitation.
4: if Exploration then
5: Choose action choice a using an exploration strategy.
6: else if Exploitation then
7: Identify action choice a with the maximal $Q(s, a)$ value by presenting the state vector S, the action vector $A = \{1, . . . , 1\}$, and the reward vector $R = \{1, 0\}$ to FALCON.
8: end if
9: Perform the action choice a, observe the next state s', and receive a reward
r (if any) from the environment.
10: Estimate the revised value function $Q(s, a)$ following a TD formula, such
as $\_Q(s, a) = \alpha(r + \gamma \max_a Q(s', a') - Q(s, a))$.
11: Formulate action vector A based on action choice a and reward vector R based on Q(s, a).
12: Present the corresponding state S, action A, and reward R vectors to FALCON for learning.
13: Update the current state by s = s'.
14: Repeat from Step 2 until s is a terminal state.
```

5.2 Algorithm 2

Translation of Propositional Rules

```
Ensure: Initialize FALCON with an uncommitted cognitive node.
1: for each propositional rule $r_j$ do
2: for each attribute $a_p \in X^{r_j}$ do
3: for each attribute-value binding $b_{pq} \in V(a_p)$ do
4: Translate bpq into vector vpq using (5).
5: end for
6: Translate ap into attribute vector Sp using (6).
7: end for
8: Translate antecedent $X^{r_j}$ into state vector $S^{r_j}$ using (7).
9: Repeat steps 3–7 for translation of each attribute $a_p \in Y^{r_j}$ .
10:Translate consequent $Y^{r_j}$ into action vector $A^{r_j}$ using (8).
11: Set reward prj into reward vector $R^{r_j}$ using (9).
12: Operate FALCON in INSERT mode to insert translated propositional rule $r_j$ as $\{S^{r_j},A^{r_j},R^{r_j}\}$.
13: end for
14: return FALCON with inserted domain knowledge.
```

VI. EVALUATION RESULT

The objective of the experiments is to evaluate the suitability of TD-FALCON as an inference engine for the entity agent operating in a complex decision-making domain. It is required to provide accurate response to scenarios using a priori knowledge as well as learned knowledge. The experiments compare the prediction accuracy of TD-FALCON trained using two learning paradigms - RL and supervised learning (SL), each with or without a priori knowledge and also against another rule inference engine known as DROOLS. SL and DROOLS are included for baseline comparison against the RL approach.

TD-FALCON operates in the PERFORM mode to derive a response from its knowledge base while it gets into the LEARN mode to update this knowledge base. TD-FALCON operates with baseline vigilance $\rho_{ck} = \{0.2, 0.8, 0.5\}$ for the state, action and reward fields respectively in the LEARN mode while it has ρ_{ck} = {0.0} in the PERFORM mode. Both modes use a common set of values for the following sets of parameter: choice parameters α_{ck} = {0.1, 0.001, 0.001}, learning rate βck = 1.0 for fast learning and contribution parameter γ_{ck} = 1 3 for k = 1, 2, 3.

DROOLS yield a consistent prediction accuracy of 97.15% while TD-FALCONs trained using SL achieve 100% prediction accuracy earlier than those trained using RL. This is expected as SL teaches the expected responses to TD-FALCON while RL requires more iteration to explore the solution space for suitable responses.

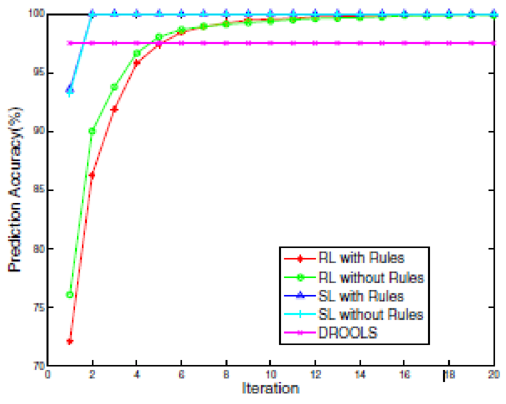

Fig 3: Comparison of action choice prediction accuracy

The profile of the prediction accuracy plots of the TDFALCON trained with and without rules in is quite closely matched. The inherent inadequacy of the inserted rules is also highlighted by the lower initial prediction accuracy of the configurations with rule insertion over those without rule insertion. This indicates the reduced role of the a priori knowledge after the acquisition of more sophisticated knowledge. The learning mechanism of TDFALCON is able to supersede the less adequate rules with those that are able to provide more accurate responses.

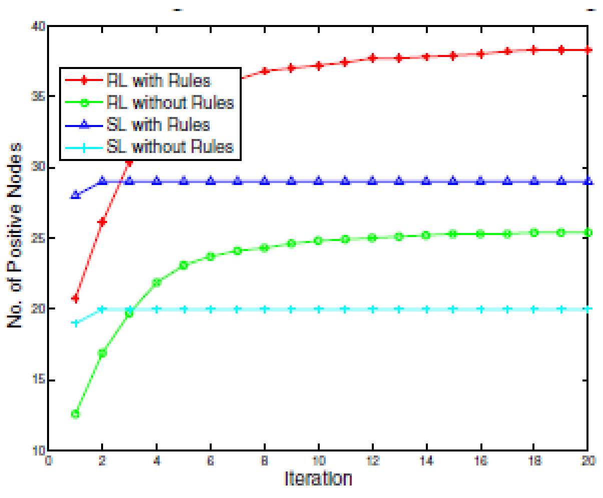

Fig 4: Comparison of the number of cognitive nodes created

Plots the creation of the cognitive nodes from each of the experiment configurations. The production of cognitive nodes plateaus as the prediction accuracy approaches 100% accuracy. This indicates that TD-FALCON has acquired sufficient knowledge to provide the appropriate responses to all the situations. Generalization is observed as the number of positive nodes created is significantly lesser than the situations that it has to respond to.

VII. CONCLUSION

This paper has shown how domain knowledge can be integrated with RL using a self-organizing neural network known as TD-FALCON. We have analytically shown how the inserted domain knowledge is utilized for action selection and learning. In addition, we proposed the greedy exploitation and reward vigilance adaptation strategies to make better use of domain Knowledge to improve learning efficiency. Using such an approach, exploration is triggered only when no effective cognitive node can be exploited for the states. It is shown that the appropriate cognitive nodes can be selected as the reward vigilance is adapted during RL.

To illustrate the efficacy of the proposed strategies for integrating domain knowledge with RL, experiments were conducted using the PE and MNT problem domains. Comparing with the selected models, the experiment results show that inserting domain knowledge directly into TD-FALCON using the proposed strategies improves success rates and reduces code population in these two distinct problem domains. Comparison of timing information from these two problem domains also shows the proposed strategies to be more efficient than the compared models.

This work of integrating domain knowledge and RL using a self-organizing neural network sets the framework for developing more efficient autonomous knowledge-based systems capable of continuously expanding its knowledge through real time interaction with the environment. In our future work, we shall embark on the application of the proposed strategies in more challenging and complex real-world problem domains. Beyond the type of logical structure, domain knowledge in these problem domains is likely to be more complex and heterogeneous. By drawing inspirations from the fields of cognitive psychology and neuroscience, we aim to build self organizing knowledge systems for addressing the issues of acquiring, managing, and retrieving such rich and diverse knowledge, possibly through the use of different types of memory representations and models

REFERENCES

[1] A. M. Shapiro, "How including prior knowledge as a subject variable may change outcomes of learning research," Amer. Educ. Res. J., vol. 40, no. 1, pp. 159–189, 2004.

[2] A.-H. Tan, "Cascade ARTMAP: Integrating neural computation and symbolic knowledge processing," IEEE Trans. Neural Netw., vol. 8, no. 2, pp. 237–250, Mar. 1997.

[3] Y. S. Abu-Mostafa, "Learning from hints in neural networks," J. Complex., vol. 6, no. 2, pp. 192–198, 1990.

[4] M. Pazzani and D. Kibler, "The utility of knowledge in inductive learning," Mach. Learn., vol. 9, no. 1, pp. 57–94, 1992.

[5] T. M. Mitchell and S. Thrun, "Explanation-based neural network learning for robot control," in Advances in Neural Information Processing Systems 5. San Mateo, CA, USA: Morgan Kaufmann, 1993, pp. 287–294.

[6] G. G. Towell and J. W. Shavlik, "Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules," in Advances in Neural Information Processing Systems 4. San Mateo, CA, USA: Morgan Kaufmann, 1992, pp. 977–984.

[7] L.-M. Fu, "Knowledge-based connectionism for revising domain theories," IEEE Trans. Syst., Man, Cybern., vol. 23, no. 1, pp. 173–182, Jan./Feb. 1993.

[8] C. H. C. Ribeiro, "Embedding a priori knowledge in reinforcement learning," J. Intell. Robot. Syst., vol. 21, no. 1, pp. 51–71, 1998.

[9] R. Schoknecht, M. Spott, and M. Riedmiller, "Fynesse: An architecture for integrating prior knowledge in autonomously learning agents," Soft Comput., vol. 8, no. 6, pp. 397–408, 2004.

[10] G. Hailu and G. Sommer, "Integrating symbolic knowledge in reinforcement learning," in Proc. Int. Conf. Syst., Man, Cybern., vol. 2. Oct. 1998, pp. 1491–1496.

[11] D. Shapiro, P. Langley, and R. Shachter, "Using background knowledge to speed reinforcement learning in physical agents," in Proc. Int. Conf. Auto. Agents, May 2001, pp. 254–261.

[12] A.-H. Tan, "FALCON: A fusion architecture for learning, cognition, and navigation," in Proc. IJCNN, Budapest, Hungary, Jul. 2004, pp. 3297–3302.

[13] 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 Trans. Neural Netw., vol. 19, no. 2, pp. 230–244, Feb. 2008.

[14] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," Comput. Vis., Graph., Image Process., vol. 37, no. 1, pp. 54–115, 1987.

[15] T.-H. Teng, Z.-M. Tan, and A.-H. Tan, "Self-organizing neural models integrating rules and reinforcement learning," in Proc. IEEE IJCNN, Jun. 2008, pp. 3770–3777.

BIOGRAPHIES

1. Ms. Nisha S., MCA., M.Phil Research Scholar, Department of Computer Science DKM College for Women (Autonomous), Vellore. TamilNadu. India.

2. Mrs. Sangeetha Lakshmi G., Asst. Prof Department of Computer Science DKM College for Women (Autonomous), Vellore, TamilNadu, India.

3. Ms. Raja Lakshmi N.S., Asst. Prof Department of Computer Science DKM College for Women (Autonomous), Vellore, TamilNadu, India.