

Self-Organizing Neural Architecture for Reinforcement Learning

Ah-Hwee Tan

Intelligent Systems Centre and School of Computer Engineering
Nanyang Technological University, Nanyang Avenue, Singapore 639798
E-mail: asahtan@ntu.edu.sg

Abstract. Self-organizing neural networks are typically associated with unsupervised learning. This paper presents a self-organizing neural architecture, known as TD-FALCON, that learns cognitive codes across multi-modal pattern spaces, involving sensory input, actions, and rewards, and is capable of adapting and functioning in a dynamic environment with external evaluative feedback signals. We present a case study of TD-FALCON on a mine avoidance and navigation cognitive task, and illustrate its performance by comparing with a state-of-the-art reinforcement learning approach based on gradient descent backpropagation algorithm.

1 Introduction

Reinforcement learning [9] is a learning paradigm wherein an autonomous system learns to adjust its behaviour 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 state to an action, and the second associating a pair of state and action to a utility value, using strategies in reinforcement learning [13]. The problem of the original approach 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.

Typically associated with unsupervised learning, self-organizing neural networks can also be used for reinforcement learning. For example, self-organizing map (SOM) has been used for the representation and generalization of continuous state and action spaces [8, 7]. 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 gradient descent backpropagation neural networks. Following a similar approach, Ueda *et. al* [12] use Adaptive Resonance Theory (ART) models to learn the clusters of state and action patterns. The clusters are in turns used as the compressed states and actions by a Q-learning module.

Supervised learning version of self-organizing networks can also be used for reinforcement learning. For instance, Ninomiya [5] couples a supervised ART

system with a temporal difference reinforcement learning module in a hybrid architecture. Whereas the states and actions in the reinforcement module are exported from the supervised ART system, the two learning systems operate independently. The redundancy in representation unfortunately leads to instability and unnecessarily long processing time in action selection as well as learning of value functions.

This paper describes a natural extension of Adaptive Resonance Theory (ART) [3] for developing an integrated reinforcement learner that adapts its behaviour through interacting with the environment. Whereas predictive ART performs supervised learning through the pairing of teaching signals and the input patterns [1, 10], the proposed neural architecture, known as FALCON (Fusion Architecture for Learning, COgnition, and Navigation), learns multi-channel mappings simultaneously across multi-modal input patterns, involving states, actions, and rewards, in an online and incremental manner. Compared with the system presented in Ninomiya (2002), FALCON presents a truly integrated solution in the sense that there is no implementation of a separate reinforcement learning module or Q-value table. Using competitive coding as the underlying principle of computation, the network dynamics encompasses a myriad of learning paradigms, including unsupervised learning, supervised learning, as well as reinforcement learning. In this paper, we focus on a class of FALCON networks, known as TD-FALCON, that learns the value functions of the state-action space estimated through temporal difference (TD) algorithms.

To evaluate the proposed system, experiments have been conducted based on a minefield navigation task similar to the one developed at the Naval Research Laboratory (NRL) [4]. The task involves an autonomous vehicle (AV) learning to navigate through obstacles to reach a stationary target (goal) within a specified number of steps. Experimental results have shown that TD-FALCON produces a much more stable performance and yet learns faster, by roughly two orders of magnitude, than a state-of-the-art reinforcement learning system using the gradient based backpropagation (BP) network.

The rest of the paper is organized as follows. Section 2 presents the FALCON architecture and the associated learning and prediction algorithms. Section 3 presents the TD-FALCON algorithm, specifically, the action selection policy and the value function estimation mechanism. The final section presents the case study on the minefield navigation simulation domain and discusses the simulation results.

2 FALCON ARCHITECTURE

The proposed FALCON architecture is based on multi-channel Adaptive Resonance Associative Map (multi-channel ARAM) [11], an extension of predictive Adaptive Resonance Theory (ART) networks. For reinforcement learning, FALCON makes use of a 3-channel architecture (Figure 1), comprising a cognitive field F_2^c and three input fields, namely a sensory field F_1^{c1} for representing current states, an action field F_1^{c2} for representing actions, and a reward field F_1^{c3} for

representing reinforcement values. The generic network dynamics of FALCON, based on fuzzy ART operations [2], is described below.

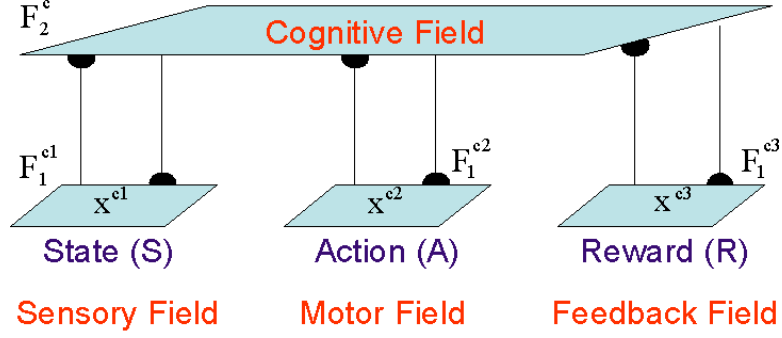


Fig. 1. The FALCON architecture.

Input vectors: Let $\mathbf{S} = (s_1, s_2, \dots, s_n)$ denote the state vector, where $s_i \in [0, 1]$ indicates the sensory input i . Let $\mathbf{A} = (a_1, a_2, \dots, a_m)$ denote the action vector, where $a_i \in [0, 1]$ indicates a possible action i . Let $\mathbf{R} = (r, \bar{r})$ denote the reward vector, where $r \in [0, 1]$ is the reward signal value and \bar{r} (the complement of r) is given by $\bar{r} = 1 - r$. 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.

Activity vectors: Let \mathbf{x}^{ck} denote the F_1^{ck} activity vector for $k = 1, \dots, 3$. Let \mathbf{y}^c denote the F_2^c activity vector.

Weight vectors: Let \mathbf{w}_j^{ck} denote the weight vector associated with the j th node in F_2^c for learning the input patterns in F_1^{ck} for $k = 1, \dots, 3$. 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*.

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

To emulate the activities of sense, act, and learn, FALCON network operates in one of the two modes, namely prediction and learning. The detailed algorithm is presented below.

2.1 Predicting

In a predicting mode, FALCON receives input patterns in one or more fields and predicts the patterns of the remaining fields. Upon input presentation, the input fields receiving values are initialized to their respectively input vectors. Input fields not receiving values are initialized to \mathbf{N} , where $N_i = 1$ for all i .

The prediction process in FALCON takes place in three key steps, namely code activation, code competition, and activity readout, described as follows.

Code activation: A bottom-up propagation process first takes place in which the activities (known as choice function values) of the cognitive nodes in the F_2^c field are computed. Specifically, Given the activity vectors \mathbf{x}^{c1} , \mathbf{x}^{c2} and \mathbf{x}^{c3} , for each F_2^c node j , the choice function T_j^c is computed as follows:

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

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

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

$$T_j^c = \max\{T_j^c : \text{for all } F_2^c \text{ node } j\}. \quad (2)$$

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

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

$$\mathbf{x}^{c2(\text{new})} = \mathbf{x}^{c2(\text{old})} \wedge \mathbf{w}_J^{c2}. \quad (3)$$

The resultant F_1^{c2} activity vectors are thus the fuzzy AND of their original values and their corresponding weight vectors.

2.2 Learning

In a learning mode, FALCON performs code activation and code competition (as described in the previous section) to select a winner J based on the activity vectors \mathbf{x}^{c1} , \mathbf{x}^{c2} , and \mathbf{x}^{c3} . To complete the learning process, template matching and template learning are performed as described below.

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

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

When resonance occurs, learning then ensues, as defined below. If any of the vigilance constraints is violated, mismatch reset occurs in which the value of the choice function T_j^c is set to 0 for the duration of the input presentation.

With a *match tracking* process, at the beginning of each input presentation, the vigilance parameter ρ_{c1} equals a baseline vigilance $\bar{\rho}_{c1}$. If a mismatch reset occurs, ρ_{c1} is increased until it is slightly larger than the match function m_J^{c1} . The search process then selects another F_2^c node J under the revised vigilance criterion until a resonance is achieved.

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

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

For an uncommitted node J , the learning rates β^{ck} are typically set to 1. For committed nodes, β^{ck} can remain as 1 for fast learning or below 1 for slow learning in a noisy environment. When an uncommitted node is selecting for learning, it becomes *committed* and a new uncommitted node is added to the F_2^c field. FALCON thus expands its network architecture dynamically in response to the input patterns.

The network dynamics of the FALCON architecture described above can be used to support a myriad of cognitive operations. We describe how FALCON can be used to learn action policies and value functions for reinforcement learning in the following section.

3 TD-FALCON

TD-FALCON incorporates Temporal Difference (TD) methods to estimate and learn value functions, specifically, 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 used in the action selection mechanism, *the policy*, that strives to achieve a balance between exploration and exploitation so as to maximize the total reward over time. A key advantage of using TD methods is that they can be used for multiple-step prediction problems, in which the merit of an action can only be known after several steps into the future (delayed reward).

The general sense-act-learn algorithm for TD-FALCON is summarized in Table 1. Given the current state s , the FALCON network is used to predict the value of performing each available action. The value functions are then processed by an action selection strategy (also known as policy) to select an action. Upon receiving a feedback (if any) from the environment after performing the action, a TD formula is used to compute a new estimate of the Q value for 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 value. The details of the action selection strategy and the temporal difference equation are described in the following subsections.

3.1 Action Selection Policy

Action selection policy refers to the strategy for selecting an action from the set of the available actions for an agent to take in a given state. It is the policy

Table 1. Generic flow of the TD–FALCON algorithm.

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

referred to in step 3 of the TD-FALCON algorithm described in Table 1. The simplest action selection policy is to pick the action with the highest value predicted by the TD-FALCON network. However, if an agent keeps selecting the optimal action that it believes, it will not be able to explore and discover better alternative actions. There is thus a fundamental tradeoff between *exploitation*, i.e., sticking to the best actions believed, and *exploration*, i.e., trying out other seemingly inferior and less familiar actions.

The ϵ -greedy policy selects the action with the highest value with a probability of $1 - \epsilon$ and takes a random action with probability ϵ [6]. With a constant ϵ value, the agent will always explore the environment with a fixed level of randomness. In practice, it may be beneficial to have a higher ϵ value to encourage the exploration of paths in the initial stage and a lower ϵ value to optimize the performance by exploiting familiar paths in the later stage. A decay ϵ -greedy policy is thus adopted to gradually reduce the value of ϵ over time.

3.2 Value Function Estimation

One key component of the TD-FALCON (Step 5) is the iterative estimation of value functions $Q(s,a)$ using a temporal difference equation $\Delta Q(s, a) = \alpha TD_{err}$, where $\alpha \in [0, 1]$ is the learning parameter and TD_{err} is the temporal error term. Using Q-learning, TD_{err} is computed by

$$TD_{err} = r + \gamma \max_{a'} Q(s', a') - Q(s, a) \tag{6}$$

where r is the immediate reward value, $\gamma \in [0, 1]$ is the discount parameter, and $\max_{a'} Q(s', a')$ denotes the maximum estimated value of the next state s' . It is important to note that the Q values involved in estimating $\max_{a'} Q(s', a')$ are computed by the same TD-FALCON network itself and not by a separate reinforcement learning system. The Q-learning update rule is applied to all states

that the agent traverses. With value iteration, the value function $Q(s, a)$ is expected to converge to $r + \gamma \max_{a'} Q(s', a')$ over time.

Whereas many reinforcement learning systems have no restriction on the values of rewards r and thus the value function $Q(s, a)$, TD-FALCON and ART systems typically assume that all input values are bounded between 0 and 1. A solution to this problem is by incorporating a scaling term into the Q-learning updating equation directly. The Bounded Q-Learning rule is given by

$$\Delta Q(s, a) = \alpha TD_{err} (1 - Q(s, a)). \quad (7)$$

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

4 The Minefield Navigation Task

The minefield simulation task studied in this paper is similar to the underwater navigation and mine avoidance domain developed by Naval Research Lab (NRL) [4]. The objective is to navigate through a minefield to a randomly selected target position in a specified time frame without hitting a mine. Our experimentation is based on a minefield navigation simulator developed in-house (Figure 2). In each trial, the autonomous vehicle (AV) starts from a randomly chosen position in the field, and repeats the cycles of sense, act, and learn. A trial ends when the system reaches the target (success), hits a mine (failure), or exceeds 30 sense-act-learn cycles (out of time). The target and the mines remain stationary during the trial.

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

We conduct experiments with both immediate and delayed evaluative feedback. For both reward schemes, at the end of a trial, a reward of 1 is given when the AV reaches the target. A reward of 0 is given when the AV hits a mine. For the immediate reward scheme, a reward is estimated at each step of the trial by computing a utility function $utility = \frac{1}{1+rd}$, where rd is the remaining distance between the current position and the target position. When the AV runs out of time, the reward is also computed using the utility function based on the

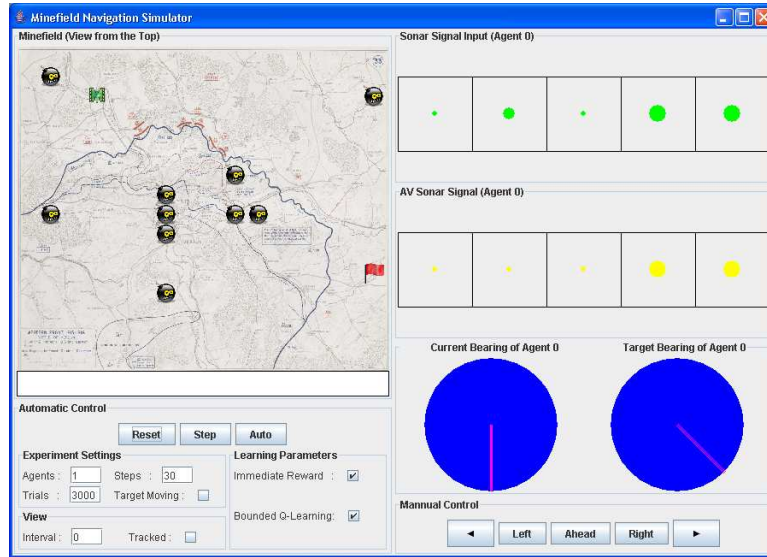


Fig. 2. The minefield navigation simulator.

remaining distance to the target. For the delayed reward scheme, no immediate reward is given at each step of the trial. A reward of 0 is given when the system runs out of time.

The FALCON system consists of 18 nodes in the sensory fields (representing 5x2 complement-coded sonar signals and 8 target bearing values), 5 nodes in the action field, and 2 nodes in the reward field (representing the complement-coded function value). All FALCON experiments use a standard set of parameter values: choice parameters $\alpha^{c^k} = 0.1$ for $k = 1, 2, 3$; learning rate $\beta^{c^k} = 1.0$ for $k = 1, 2, 3$ for fast learning; contribution parameters $\gamma^{c^1} = \gamma^{c^2} = 0.5$; baseline vigilance parameters $\bar{\rho}^{c^1} = \bar{\rho}^{c^2} = 0.2$ for a marginal level of match criterion on the state and action spaces, and $\bar{\rho}^{c^3} = 0.5$ for a stricter match criterion. For temporal difference learning, the learning rate α is fixed at 0.5. The discount factor γ is set to 0.1 for experiments with immediate reward and 0.9 for experiments with delayed reward. The initial Q value is set to 0.5. For action selection policy, ϵ is initialized to 0.5 and decayed at a rate of 0.0005 until it drops to 0.005.

To put the performance of TD-FALCON in perspective, we further conduct experiments to evaluate the performance of a reinforcement learning system, using the standard Q-learning rule and a gradient descent backpropagation (BP) algorithm as the function approximator. We have chosen the BP algorithm as the benchmark of comparison as it has been shown to produce superior approximation accuracy and successfully applied to many applications [14]. The BP learner employs a standard three-layer Perceptron architecture to learn the value function with a learning rate of 0.25 and a momentum term of 0.5. The input layer consists of 18 nodes representing the 5 sonar signal values, 8 possible

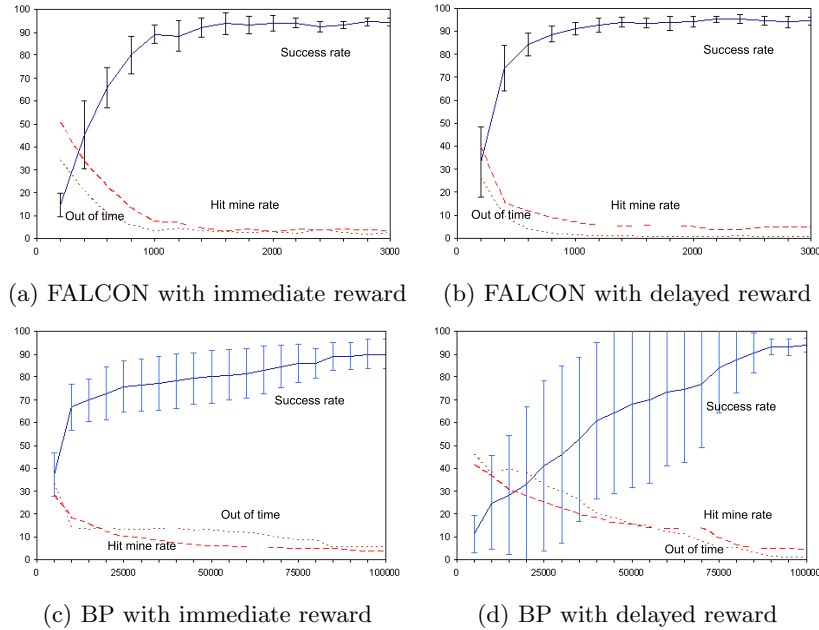


Fig. 3. The performance of TD-FALCON and BP reinforcement learner over 3000 trials across ten experiments.

target bearings, and 5 selectable actions. The output layer consists of only one node representing the value of performing an action in a particular state. A key issue in using a BP network is the determination of the number of hidden nodes. We experiment with a varying number of nodes empirically and obtain the best results with 36 nodes. To enable a fair comparison, the BP learner also makes use of the same action selection module based on the decay ϵ -greedy policy.

Figure 3 summarizes the performance of TD-FALCON and the BP reinforcement learner in terms of success rate, hit mine rate, and out-of-time rate, averaged at 200-trial intervals over 3000 trials across ten sets of experiments. The results are based on a 16 by 16 minefield containing 10 mines. We can see that the success rate of TD-FALCON increases steadily right from the beginning. Beyond 1000 trials, the TD-FALCON system can achieve over 90 percent success rates. As for the BP experiments, we observe a large performance variance, even towards the end of the 100,000 trials. In some cases, the BP learner can achieve up to 98% success rates as early as after 60,000 trials. In other cases, however, the system is stuck in a plateau with less than 10% success rates.

For both types of experiments, the BP reinforcement learner generally takes a very large number of trials (around 90,000 trials) to reach around 90% success rates on average. In contrast, TD-FALCON consistently achieves the same level of performance within the first 1000 trials. This indicates that TD-FALCON is roughly 90 times (about two orders of magnitude) faster than the BP learner in

terms of learning efficiency. Considering network complexity, the BP learner has the advantage of a highly compact network architecture. When trained properly, a BP network consisting of only 36 hidden nodes can produce very good performance. In terms of stability, however, TD-FALCON is clearly a more reliable learner by consistently producing superior performance.

References

1. G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3:698–713, 1992.
2. G. A. Carpenter, S. Grossberg, and D. B. Rosen. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759–771, 1991.
3. G.A. Carpenter and S. Grossberg. Adaptive Resonance Theory. In M.A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 87–90. Cambridge, MA: MIT Press, 2003.
4. D. Gordan and D. Subramanian. A cognitive model of learning to navigate. In *Nineteenth Annual Conference of the Cognitive Science Society*, 1997.
5. S. Ninomiya. A hybrid learning approach integrating adaptive resonance theory and reinforcement learning for computer generated agents. Technical report, Ph.D. Dissertation, University of Central Florida, 2002.
6. A. Pérez-Uribe. *Structure-Adaptable Digital Neural Networks*. PhD thesis, Swiss Federal Institute of Technology-Lausanne, 2002.
7. J. Provost, B.J. Kuipers, and R. Miikkulainen. Self-organizing perceptual and temporal abstraction for robotic reinforcement learning. In *Proceedings, AAAI Workshop on Learning and Planning in Markov Processes*, 2004.
8. A. J. Smith. Applications of the self-organising map to reinforcement learning. *Neural Networks*, 15(8-9):1107–1124, 2002.
9. R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
10. A. H. Tan. Adaptive Resonance Associative Map. *Neural Networks*, 8(3):437–446, 1995.
11. A. H. Tan and H. S. Soon. Concept hierarchy memory model: A neural architecture for conceptual knowledge representation, learning, and commonsense reasoning. *International Journal of Neural Systems*, 4(2):93–124, 1996.
12. H. Ueda, N. Hanada, H. Kimoto, and T. Naraki. Fuzzy Q-learning with the modified fuzzy ART neural network. In *Proceedings, IEEE/WIC/ACM International Conference on Intelligent Agent Technologies*, pages 308–315. IEEE Computer Society press, 2005.
13. C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3/4):279–292, 1992.
14. P. Werbos. ADP: Goals, opportunities and principles. In Jennie Si, Andrew G. Barto, Warren Buckler Powell, and Don Wunsch, editors, *Handbook of Learning and Approximate Dynamic Programming*. Wiley-IEEE Press, July 2005.