# Dynamic and Incremental Exploration Strategy in Fusion Adaptive Resonance Theory for Online Reinforcement Learning

Budhitama Subagdja
*Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY)*
*Nanyang Technological University, Singapore*
*Email: budhitama@ntu.edu.sg*

**Abstract**

One of the fundamental challenges in reinforcement learning is to setup a proper balance between exploration and exploitation to obtain the maximum cummulative reward in the long run. Most protocols for exploration bound the overall values to a convergent level of performance. If new knowledge is inserted or the environment is suddenly changed, the issue becomes more intricate as the exploration must compromise the pre-existing knowledge. This paper presents a type of multi-channel adaptive resonance theory (ART) neural network model called fusion ART which serves as a fuzzy approximator for reinforcement learning with inherent features that can regulate the exploration strategy. This intrinsic regulation is driven by the condition of the knowledge learnt so far by the agent. The model offers a stable but incremental reinforcement learning that can involve prior rules as bootstrap knowledge for guiding the agent to select the right action. Experiments in obstacle avoidance and navigation tasks demonstrate that in the configuration of learning wherein the agent learns from scratch, the inherent exploration model in fusion ART model is comparable to the basic $\epsilon$-greedy policy. On the other hand, the model is demonstrated to deal with prior knowledge and strike a balance between exploration and exploitation.

**Keywords**: *Reinforcement Learning, Exploration strategy, Adaptive Resonance Theory*

## 1. Introduction

Reinforcement learning studies the techniques of how autonomous agents learn a task by directly interacting with the environment. The learning is often formalized as Markov decision process (MDP) in which the agent learns through cycles of sense, act, and learn [13]. Classical solutions to the reinforcement learning problem generally involve learning *policy function* which maps each state to a desired action and *value function* which associates each pair of state and action to a utility value (Q-values) [22].

A fundamental issue in the original formulation of reinforcement learning is how to setup a proper balance between exploration and exploitation so that the agent can maximally obtain its cummulative reward in the long run [18]. A common approach to address this issue is to apply a stochastic algorithm which selects between a random choice and a learnt action based on a probability value ($\epsilon$-greedy). In practice, the rate of exploration changes in an ad hoc manner so that the agent tends to explore the environment at the beginning and gradually make use of its learnt knowledge. Another approach uses *softmax* policy that makes all actions to have an equal probability to be chosen at the beginning, and gradually tend to choose more on the action with the best (maximum) value estimate. This kind of approach assumes that the agent at the start of its activity explores the environment in a random fashion and gradually makes use of its own learnt knowledge.

The stochastic approach including the ones that use the gradual change is an effective way for reinforcement learning when the agent learns offline from scratch (learning is conducted in a simulation and separated from the actual performance or usage in the environment). However, in online learning (the learning is conducted while performing actions in the actual environment and tasks), it is difficult if not impossible to determine the right probability for action selection and/or the proper rate of its change so that the overall performance can be optimal.

Another pertinent issue in reinforcement learning is how to conduct learning when a prior or additional knowledge is given on the run. The stochastic approach of action selection policy may not be suitable as the algorithm makes the agent explores the environment at the beginning regardless the imposition of pre-existing knowledge. Recently, many approaches are starting to address this issue by incorporating *transfer learning* to reinforcement learning [17]. Most works focus on how the knowledge learnt from one task can be transferred to another reinforcement learner to boost its performance in learning the new task. Only few approaches like bayesian reinforcement learning [5] are considering to solve it through the internal action selection mechanism of reinforcement learning.

This paper presents a self-organizing neural network model called fusion ART (Adaptive Resonance Theory) that can be used as a function approximator for reinforcement learning. The neural network model has an inherent prop-

erty to grow its nodes (neurons) and connections on-the-fly which can be useful to address the exploration-exploitation dilemma as proposed in this paper. Another intrinsic feature of the neural model is the direct correspondence between a sub-network in the model and a rule representation that maps a state to an action and the reward. This feature enables the neural network to be inserted with pre-given rules on-the-fly which can tackle the issues of transfer learning. In this paper I suggest that the growing neural network provides a self-regulating mechanism to control the rate of exploration. As a part of the inherent feature of the ART network, Experiments in a navigation task domain show that the agent's performance while learning using the inherent selection mechanism is as optimal as learning with the $\epsilon$-greedy policy while facilitating the use of prior knowledge.

The rest of the paper is organized as follows. Section 2 discusses the issues and challenges in implementing the right action selection policy and its influence to the use of prior knowledge. Section 3 introduces the self-organizing neural network model as a function approximator for reinforcement learning. The section also explains how the neural network can support prior and inserted knowledge and tackle the action selection issues. Section 4 introduces the navigation simulation task and presents the experimental results. The final section concludes and discusses future work.

## 2. Action Selection and Prior Knowledge in Reinforcement Learning

Reinforcement learning is typically formulated as learning to optimize a policy function $\pi$ or a function that determines the action $a$ to take by an agent (that is being controlled) given the current state $s$ (see Figure 1). The optimization criteria is based on how much the agent is expected to receive rewards in the long run. When the model that provides good or optimal solution for policy $\pi$ is not available, the agent must explore the environment to learn the policy based on experience. In this case, the policy $\pi$ can be considered as the target knowledge to learn by the agent. To acquire the knowledge, the agent needs to select which action either to test (exploit) or to explore.

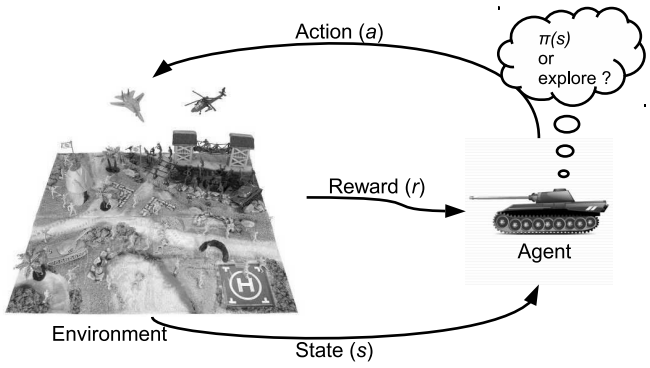


Figure 1. Exploration vs Exploitation in Reinforcement Learning.

The main purpose of action selection in reinforcement learning is to trade off exploration and exploitation in such a way that the expected reward into the future can be maximized. Standard selection policies such as $\epsilon$-greedy or those that based on a simulated annealing procedure (e.g softmax policy) consider the whole learning phase as a search process towards the maximum rewarding solution. Although it is unlikely that those action selection techniques can find the optimum solution, they offer practical approaches to find a very good solution in a fast manner over some noisy data.

When prior knowledge (e.g as an initial policy consisting of mappings of states, actions, and rewards) are inserted before learning, intuitively some conditions are expected. The expected characteristics are as follows:

1) the learner tries to exploit the pre-inserted knowledge as much as possible to behave in the environment;
2) if no prior knowledge is suitable (e.g the state is not much explored) for the current state, explore the environment (e.g random selection);
3) if prior knowledge is wrong or inaccurate, adjust the knowledge and gradually shift to full exploitation.

However, the expectations above contradict the application of the standard action-selection policy (e.g $\epsilon$-greedy) in reinforcement learning. The gradual change of exploration procedure always makes the agent to explore (random selection) at the beginning regardless the status of the agent's prior knowledge. Although it will shift eventually to its exploitation mode, the set of acquired knowledge during exploration makes the pre-given knowledge insignificant. The stochastic approach is suitable only for learning from scratch, but the inclusion of pre-inserted knowledge becomes useless.

Some non-greedy (non-stochastic) approaches of exploration strategy have been devised to be based on the context of explored states or the knowledge of the agent [18]. Methods such as *counter-based exploration* [18] and *interval estimation algorithm* [8] make use of additional information to direct the selection of exploration behavior during online learning. For each state-action pair $(s, a)$, a certain value $v$ is maintained as a visiting counter or a special utility value. A more recent version of the kind is R-Max, which also makes use of visiting counters incorporated with *known/unknown* flag on each state-action pair [1]. The algorithm has also been extended to deal with large and continuous environment (R-Max-Lspi) [9]. Another approach uses Bayesian methods to address the exploration-exploitation trade off while facilitating transfer learning to incorporate prior knowledge [5]. The bayesian approach encodes value function (e.g transitions, rewards) as a probability distribution. The action is selected based on this probability distribution. Prior knowledge can be provided as a collection of transition and reward mappings incorporating a probability distribution.

Although the above exploration strategies can speed up the search towards the optimum solution while taking the

state of learnt knowledge into consideration, they are still not addressing the issue of prior knowledge effectively. It is impractical to set up a map of states, actions, and rewards together with entries like visited numbers or context-dependent values prior to learning as those values should be obtained after the agent interacts with the environment. In the bayesian approach of reinforcement learning, providing a prior probability distribution may be impractical in some cases especially those with large state spaces. Moreover, computing the posterior probability for a large state space may be too complex and intractable.

The next section shows another alternative of balancing exploration and exploitation in reinforcement learning with the inclusion of prior knowledge. The approach uses a neural network architecture supporting a direct mapping between the neural connections and a symbolic rule representation. Based on the existing neural connections, the learner chooses either the learnt action or explores the environment.

## 3. Multi-Channel Adaptive Resonance Theory

When the task of reinforcement learning is large and involving continuous values, neural networks can be used as *function approximators* that can generalize the state space reducing computational effort and substantially increasing the speed of learning. The neural network substitutes the lookup table for representing the value function. Our approach in this paper uses a multi-channel adaptive resonance theory neural network called fusion ART as a function approximator for reinforcement learning.

### 3.1. Principles of Adaptive Resonance Theory

Adaptive Resonance Theory (ART) is a theory about how the brain autonomously learns to categorize, recognize, and predict objects and events in a dynamic environment [7]. It explains how a human brain acts as a self-organizing system that can rapidly learn huge amounts of data in real time from a changing world but still conduct it in a stable manner without catastrophically forgetting previously learnt knowledge. As a neural network architecture [3], ART solves the *stability-plasticity dilemma* by employing two complementary iterating processes: *bottom-u*p activation and *top-down* matching.

In Figure 2, the bottom-up process categorizes the input pattern by activation and selection of representative neurons (node $j$) in category field ($F2$). The selected category primes and focuses the input pattern through the *attentional* system. the top-down matching thus judges the degree it fits with the input pattern ($F2$) using the *orienting* system. A resonant state occurs when there is a good enough match between the bottom-up and top-down patterns. The two processes resonate with each other so that one process reinforces and is reinforced by the other. At this state, learning is initiated to update weights $\mathbf{w}_j$.

When the input pattern does not meet the top-down match according to the orienting system criteria, the current selected category is reset and suppressed until a match is found. However, if no match can be found possibly because the input pattern represents a truly novel experience, the search process recruits uncommitted neurons to learn the pattern as a novel information.

In this case, stable learning is achieved by allowing memories (or weights) to change when the input pattern is close enough to the expectation from top-down matching. The stability is also achieved by allowing the formation of a new category when the input is totally new.

Unlike other types of neural networks, in this bi-directional complementary process, there is no separation between phases of learning and activation. The categorization and learning are integrated parts of the network activity. Another characteristic of ART is that the recruitment of uncommitted neurons for representing novel patterns allows the network to grow so that network can learn incrementally in the long run in fast but stable manner.

### 3.2. Fusion ART dynamics

A multi-channel ART is a variant of ART network that incorporate multiple input (output) neural fields. Each input (output) field is associated with its own input (or output) vector and some adjustable parameters. The multi-channel ART network is a flexible architecture that serves a wide variety of purposes. The neural network can learn and categorize inputs and can be made to map a category to some predefined fields by a readout process to produce the output. Similar to ART, learning can be conducted by adjusting the weighted connections while the network searches and selects the best matching node in the category field. When no existing node can be matched, a new uncommitted node is allocated to represent the new pattern. The network thus grows dynamically in response to incoming patterns.

Depending on the task domain, the neural network may also apply different types of vector encoding on its different input fields. To handle continuous values, it is possible to employ vector calculation to process the activation of neurons which has been used in ART 2 architecture [4]. A simpler but more effective approach is to use Fuzzy operation to process inputs and categorization [2]. In this paper, we focus on the use of fuzzy operation (Fuzzy ART) rather than vector calculation (ART 2) as the former offers dynamic generalization mechanism and more expressive vector representation for prior knowledge. Multi-channel ART employing Fuzzy operations is also called fusion ART which is used throughout this paper.

In fusion ART, certain fields may apply *complement coding* to prevent *category proliferation* and enable generalization of input (output) attributes [2].

The generic network dynamics based on fuzzy ART operations, can be described as follows:

**Input vectors:** Let $\mathbf{I}^k = (I_1^k, I_2^k, \ldots, I_n^k)$ denote the input vector, where $I_i^k \in [0, 1]$ indicates the input $i$ to channel $k$. With *complement coding*, the input vector $\mathbf{I}^k$ is augmented with a complement vector $\bar{\mathbf{I}}^k$ such that

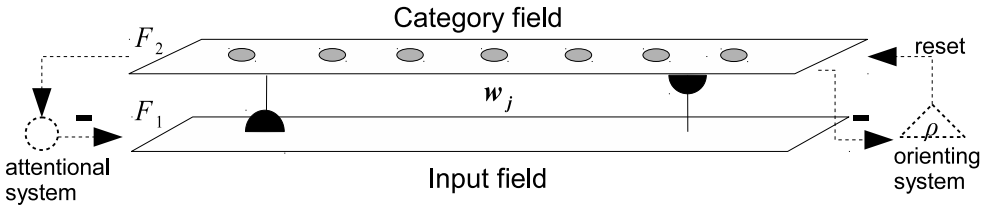


Figure 2. Simplified diagram of ART (Adaptive Resonance Theory) neural network architecture.

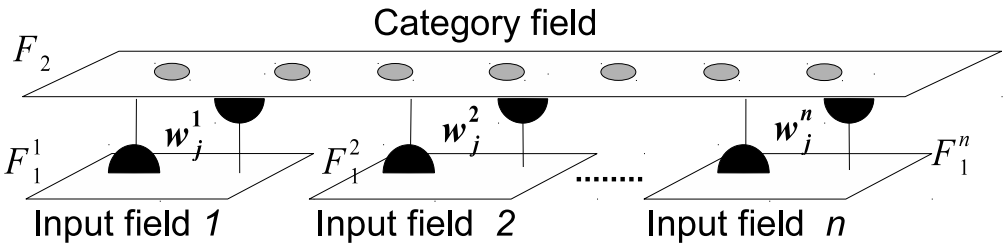


Figure 3. Fusion ART Architecture.

$\bar{I}_i^k = 1 - I_i^k$.

**Activity vectors:** Let $\mathbf{x}^k = (x_1^k, x_2^k, \ldots, x_n^k)$ denote the $F_1^k$ activity vector, for $k = 1, \ldots, n$. Initially, $\mathbf{x}^k = \mathbf{I}^k$. Let $\mathbf{y}$ denote the $F_2$ activity vector.

**Weight vectors:** Let $\mathbf{w}_j^k$ denote the weight vector associated with $j$th node in $F_2$ for learning the input patterns in $F_1^k$. Initially, $F_2$ contains only one *uncommitted* node and its weight vectors contain all 1's.

**Parameters:** The network dynamics is determined by learning rate parameters $\beta^k \in [0, 1]$ that sets how much the update is applied to the weight vector of the corresponding $k$-channel, contribution parameters $\gamma^k \in [0, 1]$ that corresponds to the importance of field $k$ during bottom-up activation, and vigilance parameters $\rho^k \in [0, 1]$ which indicates how sensitive field $k$ towards differences during tempate matching operation. In this case $k = 1, \ldots, n$. choice parameter $\alpha^k \geq 0$ indicates the importance of field $k$ among the other fields. It also used to avoid division by zero.

**Code activation:** A bottom-up activation firstly takes place in which the activities (known as choice function values) of the nodes in the $F_2$ field are computed. Given the activity vectors $\mathbf{x}^1, \ldots, \mathbf{x}^3$, for each $F_2$ node $j$, the choice function $T_j$ is computed as follows:

$$T_j = \sum_{k=1}^{K} \gamma^k \frac{|\mathbf{x}^k \wedge \mathbf{w}_j^k|}{\alpha^k + |\mathbf{w}_j^k|}, \tag{1}$$

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

**Code competition:** A code competition process follows under which the $F_2$ node with the highest choice function value is identified. The winner is indexed at $J$ where

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

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

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

$$m_J^k = \frac{|\mathbf{x}^k \wedge \mathbf{w}_J^k|}{|\mathbf{x}^k|} \geq \rho^k. \tag{3}$$

If any of the vigilance constraints is violated, mismatch reset occurs in which the value of the choice function $T_J$ is set to 0 for the duration of the input presentation. The search process then selects another $F_2$ node $J$. This search and test process is guaranteed to end as it will either find a *committed* node that satisfies the vigilance criterion or activate an *uncommitted* node which would definitely satisfy the criterion due to its initial weight values of 1s.

**Template learning:** Once a resonance occurs, for each channel $k$, the weight vector $\mathbf{w}_J^k$ is modified by the following learning rule:

$$\mathbf{w}_J^{k(\text{new})} = (1 - \beta^k)\mathbf{w}_J^{k(\text{old})} + \beta^k(\mathbf{x}^k \wedge \mathbf{w}_J^{k(\text{old})}). \tag{4}$$

The learning rule adjusts the weight values towards the fuzzy AND of their original values and the respective weight values. The rationale is to learn by encoding the common attribute values of the input vectors and the weight vectors. For an uncommitted node $J$, the learning rates $\beta^k$ are typically set to 1. For committed nodes, $\beta^k$ 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 $F_2$ field. The network thus expands its nodes and connections dynamically in response to the input patterns.

Fusion ART can be used by itself as a stand-alone neural architecture. It has been applied as a reinforcement learning architecture for autonomous agents called FALCON [14], [16]. The reinforcement learner has been applied to control a non-player character (NPC) in a competition for autonomous bots in realtime first-person-shooter video game [19]. A fusion ART network can also be considered as a building block for more complex memory or cognitive architecture. In [20], it is used as the building block for modeling episodic memory comprising memory formation, forgetting, and consolidation processes. This model of episodic memory has been demonstrated to handle memory tasks in a fast-paced realtime first-person shooter video game environment [12]. Recently, the memory model has been demonstrated to exhibit transitive inference [11] and has been integrated with a larger more complex multi-memory system [21]. It is also worth mentioning that the principles of attention, expectation, and resonance search as mentioned in [7] have been successfully applied to demonstrate the transient formation of goal hierarchy, the implementation of planning, and the online acquisition of hierarchical procedural knowledge in [10] using fusion ARTs as the building blocks.

### 3.3. Reinforcement learning in fusion ART

For reinforcement learning, the architecture makes use of 3-channel fusion ART architecture (Figure 4) comprising a categorization field $F_2$ and three input fields, namely a sensory field $F_1^1$ for representing the current state, an action field $F_1^2$ for representing the action to take, and Q (reward) field $F_1^3$ for representing the reinforcement value. Each input field employs independent parameters for code activation and pattern matching implying that different input (output) fields may be processed differently. It also implies that the neural network serves as a fuzzy approximator for reinforcement learning.

The basic reinforcement learning algorithm with fusion ART (also known as FALCON [14]) acquires an action policy by learning the mapping of the current state to the corresponding desirable action from experience. The system adjusts its internal representation upon receiving a reward feedback after performing each action. In a realistic environment, it may take a long sequence of actions before a reward or penalty is finally given. This is known as a temporal credit assignment problem in which we need to estimate the credit of an action based on what it will lead to a rewarding state eventually.

**Temporal Difference Learning.** Instead of learning a function that maps states to actions directly from immediate rewards, the FALCON incorporates Temporal Difference (TD) methods to estimate and learn the value function of state-action pairs $Q(s, a)$ that indicates the goodness for a learning system to take a certain action $a$ in a given state $s$. In this way, TD methods 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).

Given the current state $s$, the neural network is used to predict the value of performing each available action. Upon input presentation, the activity vectors are initialized as $\mathbf{x}^1 = \mathbf{S} = (s_1, s_2, \ldots, s_n)$ where $s_i \in [0, 1]$ indicates the value of sensory input $i$, $\mathbf{x}^2 = \mathbf{A} = (a_1, a_2, \ldots, a_n)$ where $a_I = 1$ if $a_I$ corresponds to the action $a$, $a_i = 0$ for $i \neq I$, and $\mathbf{x}^3 = \mathbf{Q} = (1, 1)$. The Q values are estimated through resonance search processes for every possible action $a_I$. The value functions are then processed to select an action based on the action selection policy. 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 neural network to learn the association of the current state and the chosen action to the estimated value.

One key component of the FALCON as a reinforcement learner is the iterative estimation of value functions $Q(s, a)$ using a temporal difference equation $\Delta Q(s, a) = \alpha TDerr$, where $\alpha \in [0, 1]$ is the learning parameter and $TDerr$ is the temporal error term. Using Q-learning, $TDerr$ is computed by

$$TD_{err} = r + \gamma \max_{a'} Q(s', a') - Q(s, a). \tag{5}$$

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 fusion ART 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 $Q(s, a)$, 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

$$\Delta Q(s, a) = \alpha TD_{err}(1 - Q(s, a)). \tag{6}$$

By introducing the scaling term $1 - Q(s, a)$, the adjustment of Q-values will be self-scaling so that they will not

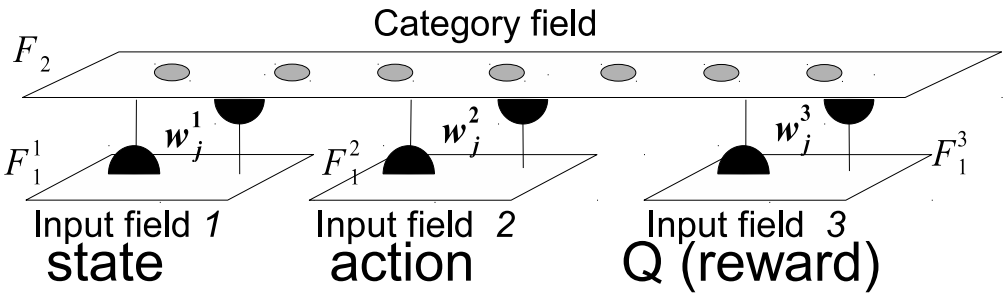


Figure 4. Fusion ART for Reinforcement Learning.

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.

The standard action selection policy used in FALCON, which is also applied in [16], is $\epsilon$-greedy policy which selects the action with the highest value with a probability of $1-\epsilon$ and takes a random action with probability $\epsilon$. A decay $\epsilon$-greedy policy is thus adopted to gradually reduce the value of $\epsilon$ over time so that the exploration is encouraged in the initial stage and gradually optimize the performance by exploiting familiar paths in the later stage. The standard TD learning algorithm using fusion ART is shown in Algorithm 1

**Algorithm 1:** TD Learning with fusion ART

1 initialize the neural network;
2 **repeat**
3  sense the environment and formulate a state $s$;
4  **for** *each action* $a \in \mathcal{A}$ **do**
5  predict the value of each $Q(s,a)$ by presenting the corresponding state and action vectors **S** and **A** to the corresponding input (output) fields;
6  **end**
7  based on the computed value functions, select an action $a$ based on the action selection policy;
8  perform action $a$;
9  observe the next state $s'$, receive a reward $r$ if any;
10  estimate $Q(s,a)$ following $\Delta Q(s,a) = \alpha TDerr$;
11  present the corresponding $s$,$a$, and $Q$ to the corresponding vectors **S**, **A**, and **Q** input (output) fields for learning;
12  update $s \leftarrow s'$;
13 **until** *$s$ is a terminal state*;

In [15], it has been proven that selecting the best action from the list of value of every possible action to take in the current state (as shown in line 4 to 7 in Algorithm 1) is equivalent to the retrieval of the action with the highest or maximal Q (reward) value given the current state. The search for the best action through the loop of value checking over all possible actions in the current state can be replaced by simply presenting the state **S**, the action **A**, and the maximum value vector $\mathbf{Q}^*$, in which $\mathbf{Q}^* = (1, 0)$, to the corresponding fields in fusion ART to retrieve the code with the best action to take.

Algorithm 1 can be replaced with Algorithm 2 which simplifies the selection of the best action by directly retrieving the code with maximum Q value. This simplified algorithm also implies that the implementation of policy function is equivalent to the retrieval of the best matching code in the network.

**Algorithm 2:** Direct Code Retrieval in TD Learning with fusion ART

1 initialize the neural network;
2 **repeat**
3  sense the environment and formulate a state $s$;
4  present vector **S**, vector $\mathbf{A}^* = (1, .., 1)$, and vector $\mathbf{Q}^* = (1, 0)$ to corresponding fields to retrieve the best code and read it out to the corresponding vector **S**, **A**, and **Q**;
5  **if** *according to the action policy, it is for exploration* **then**
6  select a random action and update the corresponding vector **A** accordingly;
7  perform action $a$;
8  observe the next state $s'$, receive a reward $r$ if any;
9  estimate $Q(s,a)$ following $\Delta Q(s,a) = \alpha TDerr$;
10  present the corresponding $s$,$a$, and $Q$ to the corresponding vectors **S**, **A**, and **Q** input (output) fields for learning;
11  update $s \leftarrow s'$;
12 **until** *$s$ is a terminal state*;

### 3.4. Rules and Adaptive Exploration

The network dynamics of fusion ART can be regarded as a myriad of learning operations namely similarity matching, associative learning, learning by instruction, and reinforcement learning. Through learning by similarity matching, the network identifies key situations in its environment that are of significance to the task at hand. Through learning by association (or directly as instructions), it learns the association between typical situations and their corresponding desired actions. Finally, through the reinforcement signals given by the environment, it learns the value of performing a specific action in a given situation.

**3.4.1. Rule insertion.** The recognition categories (codes) learnt in $F_2$ layer are compatible with a class of IF-THEN rules that maps a set of input attributes (antecedents) in one pattern channel (field) to a disjoint set of output attributes (consequents) and the estimated reward value in the other channel. In this way, instructions in the form of IF-THEN rules (accompanied by reward values) can be readily translated into the recognition categories at any stage of the learning process. Particularly, each corresponding rule can have the following format:

$$\textbf{IF } c_1 \wedge c_2 \wedge \cdots \wedge c_n \textbf{ THEN } a_1 \wedge a_2 \wedge \cdots \wedge a_m \ (\mathbf{Q} = r)$$

where $\wedge$ indicates a logical AND operator. Hence, the rules are conjunctive in the sense that the attributes in the IF clause and in the THEN clause have AND relationships. Each conditional attribute $c_i$ and action attribute $a_j$ correspond to each element of the state and action vector respectively.

**Algorithm 3:** Rule Insertion Algorithm

```
set all ρ^k to 1;
for each rule r_i do
    translate antecedent, consequent, and reward of r_i into
    vector S, A, and Q respectively;
    present S, A, and Q to the corresponding input fields;
    invoke the network dynamics procedure (Section 3.2)
    to insert the translated rule;
end
```

Imposing fusion ART with domain knowledge through explicit instructions may serve to improve learning efficiency and predictive accuracy. To insert rules into the network, the IF-THEN clauses and reward values of each instruction (rule) can be translated into corresponding input vectors. Let **S**, **A**, and **Q** are the state vector, the action vector, and the reward vector respectively. After the translation, the state-action-reward $(\mathbf{S}, \mathbf{A}, \mathbf{Q})$ triad of vectors are inserted into the network through the iterative performance of the code activation, code competition, template matching and template learning procedure (Section 3.2). During rule insertion, the vigilance parameters ($\rho^k$) are set to 1s to ensure that each distinct rule is encoded by one category node. The procedure for inserting rules is shown in Algorithm 3.

**3.4.2. Adaptive Exploration.** It is suggested in this paper that the feature of fusion ART that grows nodes and connections during learning can actually be used to regulate exploration and exploitation. The regulation can also be inherently conducted according to the status of the agent's existing knowledge. The intuition is simply that when the neural network fails to find any existing matching category for the current state, an uncommitted node is allocated for a new rule. The allocation implies that the agent needs to explore the environment as the action for that particular state is unknwon or novel. Otherwise, it just follows the original step to select the best category (rule) for further execution.

Consequently, the action selection process involving $\epsilon$-greedy can be replaced with a simpler model that adaptively turns to exploration by selecting a random action only if no matching category can be found. Algorithm 4 shows the action selection procedure using the adaptive exploration strategy of fusion ART.

**Algorithm 4:** Action Selection Policy with Adaptive Exploration Strategy

```
given state s and the best code retrieved using direct code
method (Algorithm 2);
if the code is newly allocated or just committed node in F_2
then
    set the policy to exploration;
else
    set the policy to exploitation;
```

The next section shows some experiments results confirming that the characteristics of the proposed action selection policy is comparable to the standard $\epsilon$-greedy policy even when prior rules are provided. The performance improvement rate is not influenced by any constant value or parameters.

This simplification of selection policy can be achieved thanks to the principle of overcoming the *stability-plasticity dilemma* in ART neural networks. The exploration strategy can lead to convergent if the trajectory towards the rewarding state is known or has been familiarized. However, the policy will switch back to exploration mode if novel situation is detected.

## 4. Experimental Results

To test the new exploration policy against the standard $\epsilon$-greedy, an experiment is conducted using a simulation. The simulation described in this paper is similar to the underwater navigation and mine avoidance domain developed by Naval Research Lab (NRL) [6]. The objective is to navigate through a minefield to a randomly selected target position in a specified time frame without hitting a mine. In each trial, an 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).

As the configuration of the minefield is generated randomly and changes every time a trial is started, the system needs to learn strategies that can be carried trials. 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. Figure 5 shows the screenshot of the simulator used in the experiment.

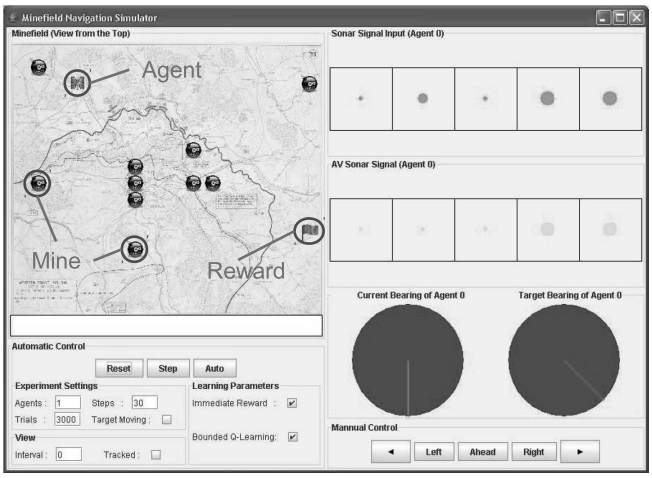


Figure 5. Screenshot of the minefield simulator

We conduct experiments with the delayed evaluative feedback (TD). 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 or runs out of time. The neural network 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). The TD fusion ART model employs a set of parameter values as follows: $\alpha = 0.1$ (all fields), $\beta = 1.0$, and $\gamma = 1.0$ (all fields). Baseline vigilances $\bar{\rho}$ for all fields are set to 0. $\rho^k$ value increases by a small factor (0.001) during the resonance search until a resonant state is achieved (in which $\rho^k$ is reset to 0 after that). This approach of changing the vigilance parameter during resonance search is also known as *match tracking* [16]. For $\epsilon$-greedy action selection policy, the decaying rate is $\epsilon = 0.001$.

The experiment is conducted to see whether the existence of prior knowledge influences the performance improvement in the neural network both with $\epsilon$-greedy and the adaptive exploration policy. At first, each reinforcement learning configuration (TD with $\epsilon$-greedy and TD with the adaptive action selection) runs for 1000 trials to let it learns some rules (Figure 6(i)). After that, the simulation parameters are reset their baseline level and a refreshed run continues until 2000 trials (Figure 6(ii)) which make 3000 trials in total. The baseline reset is made to investigate the effect of prior knowledge in TD learning using the $\epsilon$-greedy policy considering the knowledge has been captured in the first 1000 trials. Figure 6 shows the effect of the $\epsilon$ reset on success rates of TD learning with multi-channel ART for both the standard $\epsilon$-greedy policy and the inherent policy. The results are averaged over 30 independent experiment runs. After 1000 trials, the performance of TD with $\epsilon$-greedy is dropped to about half of its maximum performance. This is due to the set back of $\epsilon$ value after the reset that makes the agent to do the exploration once more. On the other hand, the one with the adaptive exploration policy keeps steady on its optimal level regardless the $\epsilon$ reset. The figure indicates that the new policy can still continue its improvement based on its current state of knowledge (rules).

The next stage of the experiment is conducted to observe whether the adaptive exploration strategy can still make the performance converges just as the $\epsilon$-greedy policy when no prior rules are given. In this stage, each success rate is complemented with its standard deviation. Figure 7 shows the performance rate and the standard deviation of both $\epsilon$-greedy and the adaptive policy in TD learning averaged over 30 independent experiment runs. It is shown that in both $\epsilon$-greedy and the adaptive exploration policy, the performance converges. In fact, it is shown that using the adaptive policy, the success rate converges faster than the other one using $\epsilon$-greedy although the difference is marginal.

The results of the experiment indicate that the adaptive exploration strategy is comparable to the standard strategy using $\epsilon$-greedy policy. However, the proposed strategy can overcome the discontinuity in learning despite the change in the environment configuration since the learnt knowledge is maintained. On the other hand, the interrupt during the trial sets back the performance gained to the initial state since the current $\epsilon$ level is not kept in the learnt knowledge although the last set of learnt knowledge is maintained over trials.

## 5. Conclusion

We have presented a multi-channel neural network framework for realizing a reinforcement learning. The neural network serves as fuzzy approximator to handle tasks and situations involving multiple channels, multimodality, large state space and continuous values. It is also possible to incorporate prior knowledge into the network by mapping rules into neural nodes and connections. Besides learning from scratch, the agent can be provided with a set of useful knowledge as bootstraps so that it can behave more effectively in a given environment.

A new action-selection policy is also introduced which also reveals that the multi-channel ART neural network has inherent features supporting online reinforcement learning. The inherent policy for adaptive exploration can replace the $\epsilon$-greedy policy so that prior knowledge can be exploited more effectively by putting the context of the decision on the status of the agent's knowledge. Our comparative experiments show that the inherent policy is comparable with the $\epsilon$-greedy.

In any case, the advantage of using the fusion ART's inherent dynamic exploration is that the performance rate becomes dependent to the availability of prior knowledge rather than some external parameters or heuristics. It is demonstrated in this paper that any useful pre-existing rules will be exploited instead of being ignored by the new policy. Consequently, the policy would instantly select the exploration mode whenever a lack of knowledge is identified. In that case, fusion ART offers an inherent self-regulating control for exploration and exploitation in reinforcement learning.

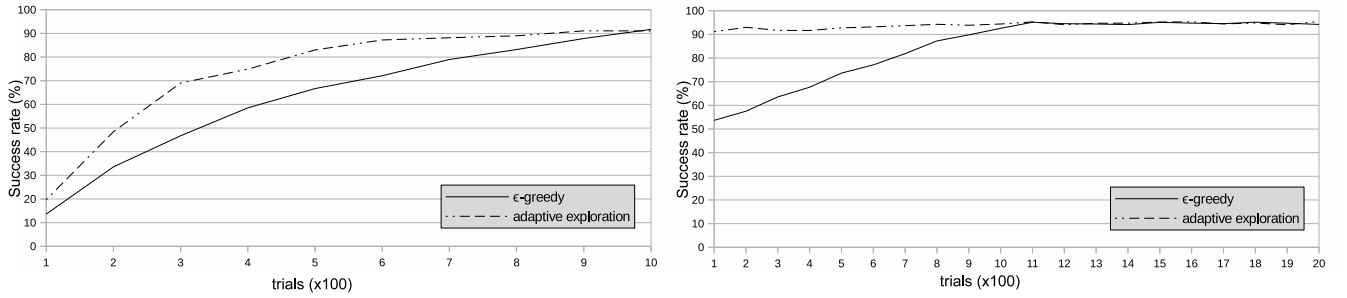


Figure 6. The effect of $\epsilon$ reset. (i) Before the reset; (ii) After the reset

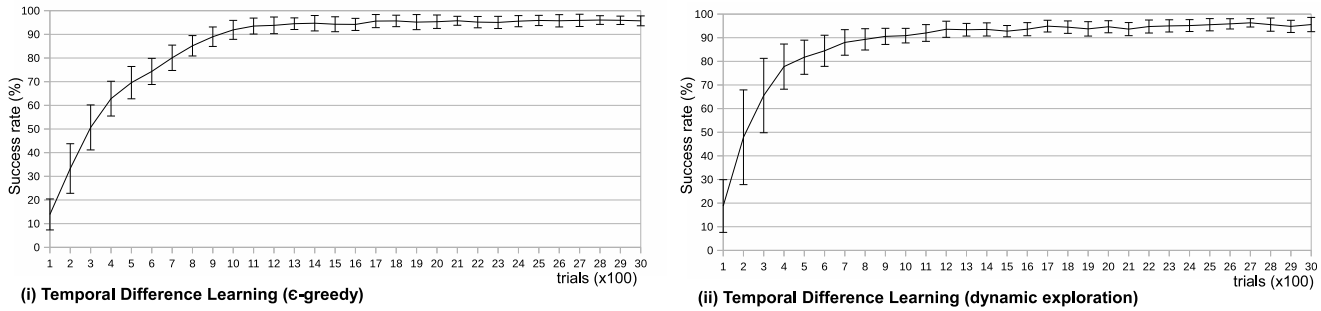


Figure 7. The learning performance of TD learning with $\epsilon$-greedy and inherent exploration policy.

Our future work will involve applying the inherent action selection strategy to more complex and challenging domains. Another possible extension is to study the relationships between the neural network parameters (e.g vigilance, contribution, choice, and learning rate parameters) and the effectiveness of reinforcement learning. Moreover, the adaptive exploration policy described in this paper is still quite simple that decision is limited to either to explore or to exploit the knowledge. Another more sophisticated strategy deserves to be investigated which may combine the policy with other strategies while exploiting the inherent features in the ART neural network as much as possible.

## Acknowledgement

This research is supported by the National Research Foundation, Prime Ministers Office, Singapore under its IDM Futures Funding Initiative.

## References

[1] R. I. Brafman and M. Tennenholtz, "R-max a general polynomial time algorithm for near-optimal reinforcement learning," *Journal of Machine Learning Research*, vol. 3, pp. 213–231, 2002.

[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*, vol. 4, pp. 759–771, 1991.

[3] G. Carpenter and S. Grossberg, "Adaptive Rresonance Theory," in *The Handbook of Brain Theory and Neural Networks*, M. Arbib, Ed. Cambridge, MA: MIT Press, 2003, pp. 87–90.

[4] G. A. Carpenter and S. Grossberg, "ART 2: Stable self-organization of pattern recognition codes for analog input patterns," *Applied Optics*, vol. 26, pp. 4919–4930, 1987.

[5] R. Dearden, N. Friedman, and S. Russell, "Bayesian q-learning," in *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*. Menlo Park: American Association for Artificial Intelligence, 1998, pp. 761–768.

[6] D. Gordan and D. Subramanian, "A cognitive model of learning to navigate," in *Proceedings, Nineteenth Annual Conference of the Cognitive Science Society*, 1997, pp. 271–276.

[7] S. Grossberg, "Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world," *Neural Networks*, vol. 37, no. 2013, pp. 1–47, 2013.

[8] L. P. Kaelbling, *Learning in Embedded Systems*. Cambridge: MIT Press, 1993.

[9] L. Li, M. L. Littman, and C. R. Mansley, "Online exploration in least-squares policy iteration," in *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, 2009, pp. 733–739.

[10] B. Subagdja and A.-H. Tan, "iFALCON: A neural archiecture for hierarchical planning," *Neurocomputing*, vol. 86, pp. 124–139, 2012.

[11] ——, "Neural modeling of sequential inferences and learning over episodic memory," *Neurocomputing*, vol. 161, no. 2015, pp. 229–242, 2015.

[12] B. Subagdja, W. Wang, A.-H. Tan, Y.-S. Tan, and L.-N. Teow, "Memory formation, consolidation, and forgetting in learning agents," in *Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, 2012.

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

[14] A.-H. Tan, "Falcon: A fusion architecture for learning, cognition, and navigation," in *Proceedings of 2004 IEEE International Joint Conference on Neural Networks (IJCNN'04)*, 2004, pp. 3297–3302.

[15] ——, "Direct code access in self-organizing neural architectures for reinforcement learning," in *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 2007, pp. 1071–1076.

[16] A.-H. Tan, N. Lu, and D. Xiao, "Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evaluative feedback," *IEEE Transactions on Neural Networks*, vol. 19, no. 2, pp. 230–244, 2012.

[17] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[18] S. B. Thrun, "The role of exploration in learning control," in *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, D. A. White and D. A. Sofge, Eds. Florence: Van Nostrand Reinhold, 1992, pp. 527–599.

[19] D. Wang, B. Subagdja, A.-H. Tan, and G.-W. Ng, "Creating human-like autonomous players in real-time first person shooter computer games," in *Proceedings of the Twenty-First Annual Conference on Innovative Applications of Artificial Intelligence (IAAI'09)*, 2009, pp. 173–178.

[20] W. Wang, B. Subagdja, A.-H. Tan, and J. A. Starzyk, "Neural modeling of episodic memory: Encoding, retrieval, and forgetting," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 10, pp. 1574–1586, 2012.

[21] W. Wang, B. Subagdja, A.-H. Tan, and Y.-S. Tan, "A self-organizing multi-memory system for autonomous agents," in *Proceedings of 2012 IEEE International Joint Conference on Neural Networks (IJCNN'12)*, 2012, pp. 480–487.

[22] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3/4, pp. 279–292, 1992.