7-2011

A hybrid agent architecture integrating desire, intention and reinforcement learning

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

Yew-Soon ONG

Akejariyawong TAPANUJ

Citation

TAN, Ah-hwee; ONG, Yew-Soon; and TAPANUJ, Akejariyawong. A hybrid agent architecture integrating desire, intention and reinforcement learning. (2011). *Expert Systems with Applications*. 38, (7), 8477-8487.
Available at: https://ink.library.smu.edu.sg/sis_research/5244

Expert Systems with Applications 38 (2011) 8477–8487

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A hybrid agent architecture integrating desire, intention and reinforcement learning

Ah-Hwee Tan *, Yew-Soon Ong, Akejariyawong Tapanuj

School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore

ARTICLE INFO

Keywords:
BDI architecture
Reinforcement learning
Plan learning
Self-organizing neural networks
Minefield navigation

ABSTRACT

This paper presents a hybrid agent architecture that integrates the behaviours of BDI agents, specifically desire and intention, with a neural network based reinforcement learner known as Temporal Difference-Fusion Architecture for Learning and COgNition (TD-FALCON). With the explicit maintenance of goals, the agent performs reinforcement learning with the awareness of its objectives instead of relying on external reinforcement signals. More importantly, the intention module equips the hybrid architecture with deliberative planning capabilities, enabling the agent to purposefully maintain an agenda of actions to perform and reducing the need of constantly sensing the environment. Through reinforcement learning, plans can also be learned and evaluated without the rigidity of user-defined plans as used in traditional BDI systems. For intention and reinforcement learning to work cooperatively, two strategies are presented for combining the intention module and the reactive learning module for decision making in a real time environment. Our case study based on a minefield navigation domain investigates how the desire and intention modules may cooperatively enhance the capability of a pure reinforcement learner. The empirical results show that the hybrid architecture is able to learn plans efficiently and tap both intentional and reactive action execution to yield a robust performance.

1. Introduction

Derived from folk psychology, belief, desire, and intention (BDI) is a popularly used framework for human modelling and logical reasoning (Bratman, Israel, & Pollack, 1988). BDI systems encode goal directed behaviours by using plans derived from expert knowledge about the task domain. An abstract plan consists of an ordered set of subgoals or actions that an agent may execute with little sensory feedback from its environment, together with an overall goal which that plan achieves if execution is completed successfully. Following a plan also serves to frame and thus constrain the subsequent reasoning and actions of agents (Pollack, 1992). The use of plans is thus useful for resource-bounded agents, which have limited computing power in sensing and/or deliberation. A traditional limitation of BDI architecture, however, is the lack of learning ability. In most cases, plans and capabilities are predefined by developers or captured from human experts.

In modern cognitive science, many have held the view that cognition is a process deeply rooted in the body's interaction with the world (Anderson, 2003; Brooks, 1999). In other words, autonomous systems acquire intelligence through their interaction with the environment. Often formalized as a Markov Decision Process (MDP), an autonomous agent performs reinforcement learning (Kaelbling, Littman, & Moore, 1996; Sutton & Barto, 1998) through a sense, act, and learn cycle in the motivation of receiving positive rewards in the future.

In view of their complementary strengths, there has been great interest in hybrid architecture that integrates high level symbolic systems, such as BDI, with low level reinforcement learning algorithms. Some examples of hybrid systems include CLARION (Sun, 2000), BDI with standard Q-learning (Norling, 2004), BDI with decision tree induction (Guerra-Hernandez, Fallah-Seghrouchni, & Soldano, 2004), and ACT-R with sequence learning (Lebiere & Wallach, 2000). Among these hybrid systems, temporal difference methods, such as Q-learning, coupled with gradient descent neural network based function approximators, have been the most effective and commonly used (Si, Barto, Powell, & Wunsch, 2004; Sun, 2000). However, gradient descent methods are well known to learn from the differences between prediction and target patterns by making small error correction steps iteratively. In addition, there is the issue of instability as learning of new patterns may erode the previously learned knowledge. Consequently, the resultant systems may not be able to learn and operate in real time.

In this paper, we present a hybrid architecture that integrates the features of BDI, namely desire and intention, and a reinforcement learning system known as Temporal Difference-Fusion Architecture for Learning and COgNition (TD-FALCON) (Tan, Lu, & Xiao, 2008; Xiao & Tan, 2007). TD-FALCON is an extension of a class of self-organizing neural networks, known as Adaptive Resonance Theory (ART) (Carpenter & Grossberg, 1987) that integrates temporal difference

* Corresponding author.
E-mail addresses: asahtan@ntu.edu.sg (A.-H. Tan), asysong@ntu.edu.sg (Y.-S. Ong), tapa0001@ntu.edu.sg (A. Tapanuj).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.01.045

methods (Sutton & Barto, 1998; Watkins & Dayan, 1992) for reinforcement learning. By inheriting the ART code stabilizing and dynamic network expansion mechanism, FALCON is capable of learning cognitive nodes encoding multi-dimensional mappings across multi-modal input patterns, involving states, actions, and rewards, in an online and incremental manner.

Using belief–desire–intention as the system's framework, the proposed BDI-FALCON architecture extends a low level reactive reinforcement learner TD-FALCON into a deliberative reasoner. With the explicit maintenance of the agent's goals in the desire module, the agent now performs reinforcement learning with the awareness of its objectives instead of relying purely on external reinforcement signals. The desire module also allows the flexibility of defining and refining the agent's goals. With the goal attainment evaluation capability, the agent can compute and generate reinforcement signals internally (implicit rewards). The desire module thus equips the agent with a higher level of self-awareness, in comparison with a pure reinforcement learning agent.

Working in parallel with the desire and reactive learning modules, the intention module equips the hybrid architecture with deliberative planning capabilities. It enables the agent to focus on its current course of action by purposefully maintain an agenda of actions to perform. This is useful in a complex environment wherein individual reactive responses are not adequate. Performing with a plan also reduces the need for an agent to repeatedly sense the environment and may therefore improve efficiency of resource-limited agents. In addition, through reinforcement learning, plans can likewise be learned and evaluated without the rigidity of user-defined plans as used in traditional BDI systems.

Compared with the other hybrid systems, the BDI-FALCON architecture described in this paper has made a number of new attempts. Firstly, TD-FALCON is a relatively new reinforcement learning system that has shown superior learning capabilities, especially in terms of real-time learning efficiency, compared with gradient descent based reinforcement learning systems (Tan et al., 2008; Xiao & Tan, 2007).

Secondly, our intention module has made use of a set of plan learning, selection, and evaluation methods, in the same vein as those used in TD-FALCON for cognitive node learning and selection. As a consequence, a plan can be selected for execution even it does not have an exact match with the current context or goal, as required by traditional BDI systems.

Thirdly, the BDI-FALCON architecture has incorporated a desire module that constantly maintains an explicit representation of the system's goals. Although its current design is relatively simple, the desire module fulfils the function of generating internal reinforcement feedback for both plan learning and reactive learning, thus closing the loop of the processes among the three components within the agent.

In addition, our work shows that both high level and low level cognitive functionalities, including desire, intention, and reactive learning, can all be realized through a unified set of mathematical models and algorithms. This approach is motivated by our hypothesis that human cognition is an emergent property arising from the interaction among low level neural circuits and pathways, which may operate according to a unified set of neural activation and learning processes (Tan, Carpenter, & Grossberg, 2007).

Last but not the least, we have conducted extensive experiments and analyzed the behaviour of the hybrid architecture with two plan and action integration strategies, in terms of the plan adoption ratios and the overall success rates. To investigate the system behaviour and capabilities, we have chosen a minefield navigation task, similar to the one developed at the Naval Research Laboratory (NRL) (Gordan & Subramanian, 1997). The task involves an autonomous vehicle (AV) learning to navigate through obstacles to reach a stationary target (goal) within a specified number of steps. Our experimental results show that hybrid architecture is able to combine intentional and reactive action execution, leading to improvement both in terms of task completion performance and efficiency.

The rest of the paper is organized as follows. Section 2 presents background and related work. Section 3 presents in details the BDI-FALCON architecture and its three main components, namely the desire module, the intention module, and the reactive learning module. Section 4 reports our case study on the minefield navigation problem. The final section provides a discussion of results and highlights more general issues.

2. Related work

Hybrid systems integrating high level capabilities, such as planning, and low level reactive modules involving learning has been an active research area. Sun (2000) described a two-level model, known as CLARION, for learning reactive plans and extracting plans from reinforcement learners. The first three layers of the bottom level form a backpropagation network learning and computing Q-values. The fourth layer (the top level with only one node) determines probabilistically the action to be performed based on a Boltzmann distribution. Given a specific problem scenario, plans can be generated on the spot using a beam search strategy that chains up actions with optimal Q-values at each step (Sun & Sessions, 2000). The plan extraction process however assumes that the next state after performing each and every action can always be determined beforehand. In addition, Sun et al. did not made explicit connection to the BDI framework, in particular, desire and intention.

Using a goal-directed (top down) approach, Wallis (2004) discussed the notion of *goal-tagged activities*, that achieved planning by low-level adherence to high-level goals without the need for explicit symbolic representation. This was achieved by the chaining of implicitly goal-encoded *activities* (which are reactive modules similar to plans in the BDI sense). Similarly, the ACT-R architecture (Anderson, Bothell, & Byrne, 2004) uses a simple recurrent network (SRN) for encoding fragments of plans and utilizes the persistence property of working memory for "reading out" the action sequences (Lebiere & Wallach, 2000). As their focus is on achieving goal-directed plan based behaviour through low level mechanism, they do not consider the issue of how plans can be learned.

Working from the BDI perspective, Norling (2004) integrated BDI with a standard table lookup version of Q-learning to learn reactive rules for path finding in a grid world. As the Q-value table stores each and every incident as an entry, the system is not able to scale up to complex and continuous domains. Subagdja and Sonenberg (2005) further extended the BDI architecture to incorporate learning, through the generation and testing of hypothesis for formulating plans. However, the learning is restricted to evaluating specific types of plans.

With the intention of learning knowledge through interacting with the environment, Karim, Sonenberg, and Tan (2006) proposed a hybrid system consisting of a high level BDI system and a low level reactive FALCON (Tan, 2004) model, in which BDI-styled plans were learned out of FALCON's reactive action execution. The Plan Generation System (PGS) used a strategy to build plans by appending actions as the system performs. Hybrid architecture was originally illustrated on a minefield navigation domain and was subsequently expanded and applied to a multi-agent predator–prey domain (Karim, Subagdja, & Sonenberg, 2006). The approach and architecture presented in this paper follow those of PGS presented by Karim and Sonenberg et al. (2006) and Karim et al. (2006). However, we have designed an entirely different set of algorithms, following the fuzzy ART operators (Carpenter, Grossberg, & Rosen, 1991), for goal

matching, plan selection, plan evaluation, and plan learning. These algorithms enable a plausible plan to be selected for execution even without an exact match in the state and goal representation. In addition, we have expanded the analysis significantly by experimenting with different strategies for integrating the intentional and reactive learning modules.

3. The integrated architecture

Following the belief–desire–intention (BDI) framework, the proposed BDI-FALCON architecture consists of three main modules, namely desire, intention, and the reactive learner (Fig. 1). The low level reactive learning module is a TD-FALCON network interacting with the environment through the three sensory, motor, and feedback channels. Based on the goals defined in the desire module and the sensory inputs received from the environment, TD-FALCON performs reinforcement learning to acquire a set of action and value policies that enables the agent to achieve its goals.

The desire module maintains an explicit representation of the agent's goals. Active goals are those that give direction to the agent's activities for it to achieve its objectives. The idea of active goals is similar to that presented in PGS (Karim & Sonenberg et al., 2006; Karim et al., 2006). By matching the defined goals with the corresponding sensory attributes, the desire module computes how well the agent has progressed towards the desired goals. The degree of goal attainment can then be used as an implicit reward signal for reinforcement learning in the reactive and intention modules. The desire module thus renders the hybrid architecture a higher level of awareness than a pure reinforcement learner that relies purely on external reward signals.

The intention module maintains the plan repository and supports the key processes of plan learning, plan selection, plan execution and plan evaluation. Each plan p in the repository comprises the start state acting as a context in which the plan is applicable, the target state of which the plan will lead to, the sequence of actions to be performed under the plan, and the estimated payoff (or confidence) of using the plan.

Given a set of active goals and the current sensory inputs as the context, the plan selection process identifies the most applicable plan to perform through a code competitive mechanism similar to that used in FALCON. During plan execution, the action sequence of the adopted plan is extracted and performed through the motor channel. Execution of plans thus enables an agent to perform a series of actions without the need of going through the typical sense–act–learn cycle for each action. This could potentially lead to saving in computation cost and enables the system to be more resilient in a challenging environment, wherein external signals may not be available all the time. Through a simple form of reinforcement learning, the plan evaluation process adjusts the confidence value of each adopted plan according to the outcome that it leads to. Plans with low confidence can then be pruned from the repository.

3.1. Reactive learning module

The reactive learning module is a TD-FALCON model, that incorporates temporal difference (TD) learning into a self-organizing

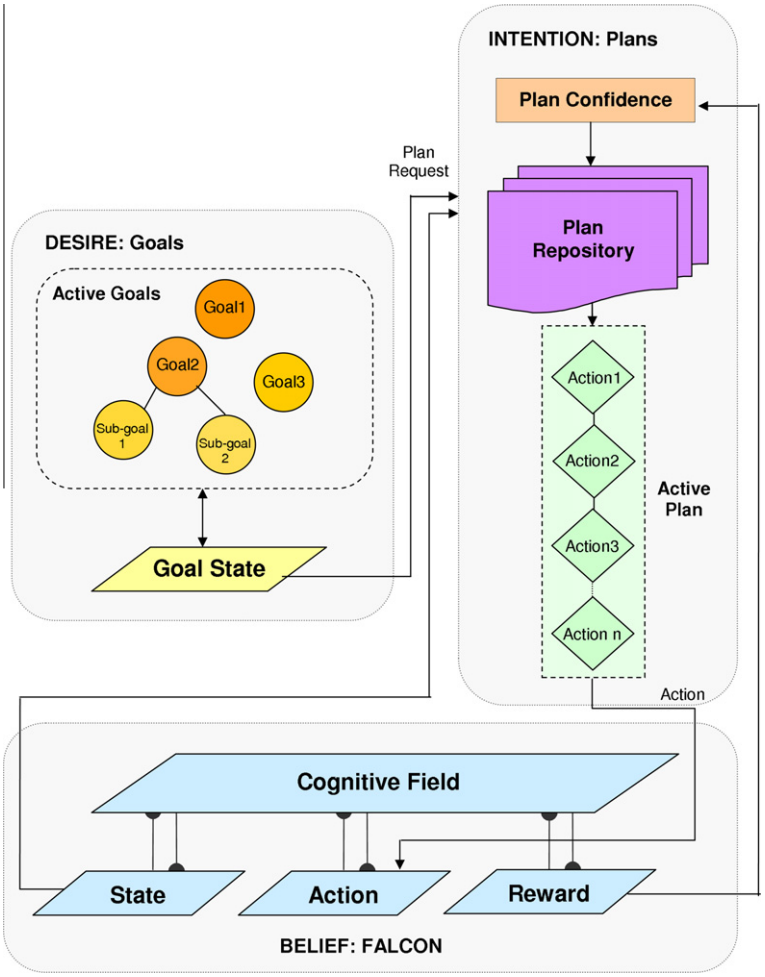


Fig. 1. The BDI-FALCON architecture.

neural model known as Fusion Architecture for Learning and COgNition (FALCON). For completeness, we present a summary of the FALCON prediction and learning dynamics as well as the overall TD-FALCON algorithm in the following sections. For a more detailed description of TD-FALCON, please refer to Tan et al. (2008) and Tan (2007).

3.1.1. FALCON dynamics

FALCON employs a 3-channel architecture (Fig. 2), consisting of a sensory field F_1^{c1} for representing the current state, a motor field F_1^{c2} for representing the available actions, a feedback field F_1^{c3} for representing the reward signals received from the environment, as well as a cognitive field F_2^c for encoding the relations among the activity patterns across the three input channels. The generic network dynamics of FALCON, based on fuzzy ART operations (Carpenter et al., 1991), is summarized below.

Input vectors: Let $\mathbf{S} = (s_1, s_2, \ldots, s_n)$ denote the state vector, where $s_i \in [0,1]$ indicates the sensory input i. Let $\mathbf{A} = (a_1, a_2, \ldots, a_m)$ 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 $k = 1, \ldots, 3$. 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 jth node in F_2^c for learning the input patterns in F_1^{ck} for $k = 1, \ldots, 3$. 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 $k = 1, \ldots, 3$; learning rate parameters $\beta^{ck} \in [0,1]$ for $k = 1, \ldots, 3$; contribution parameters $\gamma^{ck} \in [0,1]$ for $k = 1, \ldots, 3$ where $\sum_{k=1}^{3} \gamma^{ck} = 1$; and vigilance parameters $\rho^{ck} \in [0,1]$ for $k = 1, \ldots, 3$.

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 $\mathbf{x}^{c1}$, $\mathbf{x}^{c2}$ and $\mathbf{x}^{c3}$ (in the input fields F_1^{c1}, F_1^{cs} and F_1^{c3}, respectively), for each F_2^c node j, the choice function T_j^c is computed as follows:

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

where the fuzzy AND operation $\wedge$ is defined by $(\mathbf{p} \wedge \mathbf{q})_i \equiv min(p_i, q_i)$, and the norm $|\cdot|$ is defined by $|\mathbf{p}| \equiv \sum_i p_i$ for vectors $\mathbf{p}$ and $\mathbf{q}$. In essence, the choice function T_j computes the similarity of the activity vectors with their respective weight vectors of the F_2^c node j with respect to the norm of individual weight vectors.

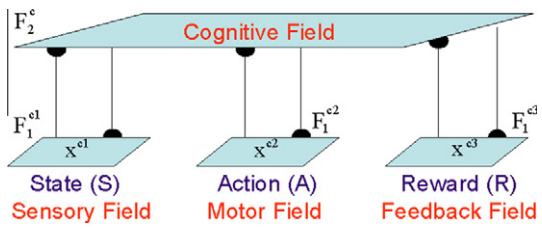


Fig. 2. The FALCON architecture.

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

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

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.

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* m_J^{ck} of the chosen code J meets its vigilance criterion:

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

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 co-operatively to achieve stable coding and maximize code compression.

When resonance occurs, learning 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. The search process then selects another F_2^c node J until a resonance is achieved. This search and test process is guaranteed to end as FALCON 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 1*s*.

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:

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

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 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.

3.1.2. TD-FALCON algorithm

TD-FALCON (Tan & Xiao, 2005; Tan et al., 2008; Xiao & Tan, 2007) 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 TD-FALCON model used in this study employs a direct code access procedure (Tan, 2007) as shown in Table 1. 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 an 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.

Table 1
TD-FALCON algorithm with direct code access.

1. Initialize the FALCON network
2. Sense the environment and formulate a state representation s
3. Following an action selection policy, choose between exploration and exploitation
   If exploring, take a random action.
   If exploiting, identify the action 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
4. Perform the action a, observe the next state s', and receive a reward r (if any) from the environment
5. Estimate the revised value function $Q(s,a)$ following a Temporal Difference (TD) formula $\Delta Q(s,a) = \alpha TD_{err}(1 - Q(s,a))$, where $TD_{err} = r + \gamma \max_{a'} Q(s',a') - Q(s,a)$, of which 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'
6. Present the state, action, and reward (Q-value) vectors (**S**, **A**, and **R**) to FALCON for learning.
7. Update the current state by $s = s'$
8. Repeat from Step 2 until s is a terminal state

3.2. The desire module

The desire module maintains and manages the agent's goals. It also handles the function of goal attainment evaluation. The dynamics of the desire module is summarized as follows.

Goal target vectors: Assuming that there is a total of M goals in the system, let $\mathscr{G} = \{\mathbf{w}^{g1}, \mathbf{w}^{g2}, \ldots, \mathbf{w}^{gM}\}$ denote the set of the goal target vectors, where $\mathbf{w}^{gj}$ indicates the target state vector of the jth goal and its element $w_i^{gj} \in [0,1]$ indicates the target value of the attribute i. When a new agent is created, its designer needs to specific the goal target vectors to reflect the goals of the agents appropriately.

Goal state vector: Let $\mathbf{x}^g = (x_1^g, x_2^g, \ldots, x_N^g)$ denote the goal state vector, where $\mathbf{x}_i^g \in [0,1]$ indicates the current value of a goal-related state attribute i. The goal state vector is typically a part of or can be derived from the state vector **S**.

Goal attainment function: After sensing the environment and extracting the goal state vector $\mathbf{x}^g$ from the state vector **S**, the goal attainment function A^g computes the match value between the goal state vector $\mathbf{x}^g$ and the goal target vectors $\mathbf{w}^g$, with respect to the norm of each individual goal target vector. Specifically, given the current goal state vector x^g, the overall goal attainment can be calculated as

$$A^g = \prod_{j=1}^{M} \frac{|\mathbf{x}^g \wedge \mathbf{w}^{gj}|}{|\mathbf{w}^{gj}|}, \tag{5}$$

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 **p** and **q**. The goal attainment value can then used as the internal reward signal (r) to the reactive module for reinforcement learning and to the intention module for plan evaluation.

3.3. The intention module

The intention module maintains a repository of plans. Each plan p_j in the repository is represented by the quad-tuple $\langle \mathbf{w}_j^{ps}, \mathbf{w}_j^{pg}, \mathbf{a}_j^p, c_j^p \rangle$, where.

- $\mathbf{w}_j^{ps}$ represents the start state acting as a context in which the plan p is applicable,
- $\mathbf{w}_j^{pg}$ represents the target state of which the plan will lead to,
- $\mathbf{a}_j^p = (a_1, a_2, \ldots, a_n)$ denotes the sequence of actions to be performed under the plan, and
- $c_j^p \in [0,1]$ denotes an estimated payoff (or confidence) of using the plan.

The key processes in the intention module include plan learning, plan selection, plan execution and plan evaluation, described as follows.

Plan learning: A plan is created each time a successful sequence of actions is discovered. A buffer is used to record the start state and the sequence of actions carried out during the trial. At the end of a successful trial, a new plan p_J is learned, such that the start state, the goal state and the action sequence are recorded into the corresponding components, namely $\mathbf{w}_J^{ps}, \mathbf{w}_J^{pg}$ and $\mathbf{a}_J^p$, of the newly created plan, respectively. The confidence value c_J^p of the new plan is initialized to a default value.

Input vectors: Let $\mathbf{x}^s$ denote the current state vector and $\mathbf{x}^g$ indicate the current goal state vector consisting of goal-related attributes.

Plan selection: The plan selection process follows the code selection strategy as used in the family of fusion ART models (Tan, 1995; Tan & Pan, 2005; Tan et al., 2007; Tan et al., 2008), wherein the choice function values of the state and goal fields are combined using a linear summation operator.[1] Specifically, given the state vector $\mathbf{x}^s$ and the goal state vector $\mathbf{x}^g$, for each plan j in the plan repository, a similarity-based choice function value (y_j) is calculated by

$$y_j = \frac{|\mathbf{w}_j^{ps} \wedge \mathbf{x}^s|}{|\mathbf{w}_j^{ps}|} + \frac{|\mathbf{w}_j^{pg} \wedge \mathbf{x}^g|}{|\mathbf{w}_j^{pg}|}, \tag{6}$$

where $|p| \equiv \sum_{i=1}^{n} p_i$ and $(p \wedge q)_i = \min(p_i, q_i)$.

Upon computing the choice function value of each plan, a competition process selects the plan p_J with the maximum choice value as the winner. If there are more than one plans with the same maximum value, the winner plan is the one with the highest confidence.

Plan matching: Before the winning plan p_J can be adopted, a matching process takes place to ensure that the selected plan is a good match. The plan matching process computes the match value with respect to the current system states $\mathbf{x}^s$ by

$$m_J^p = \frac{|\mathbf{w}_J^{ps} \wedge \mathbf{x}^s|}{|\mathbf{x}^s|} + \frac{|\mathbf{w}_J^{pg} \wedge \mathbf{x}^g|}{|\mathbf{x}^g|}. \tag{7}$$

If the match value falls below the value of the vigilance parameter ρ^p, a mismatch reset occurs and the system returns to the normal sense-act-learn cycle to identify a suitable action for that particular circumstance. Otherwise, the system adopts the selected plan and interprets it for execution. The plan vigilance ρ^p is thus an important parameter which influences the likelihood for a plan to be selected for execution.

Plan execution: Plan execution reads out an action at a time from the sequence of actions $\mathbf{a}_J^p$ encoded in the adopted plan p_J and executes it through the action field of FALCON.

[1] We had also experimented with a product operator and found no significant differences in terms of performance.

Plan evaluation: When an adopted plan leads to a outcome with a positive reward value, the confidence of the plan is increased by a small value δ^p. On the other hand, if the adopted plan leads to a negative outcome, the confidence is decreased accordingly by δ^p. Specifically, the confidence of plan $J(c_J^p)$ is updated by

$$c_J^p = \begin{cases} \min(1, c_J^p + \delta^p) & \text{if reward} = 1, \\ \max(0, c_J^p - \delta^p) & \text{if reward} = 0, \end{cases}$$

where $\delta^p \in [0,1]$ is the reinforcement rate for plan confidence. When the confidence value of a plan drops below a certain threshold, the plan is removed from the repository.

3.4. Integrating plan and reactive execution

With the presence of both plan and reactive execution capabilities, a strategy is needed for combining the plan execution of the intention module and the reactive responses of the reactive learner module. We experiment with two strategies, namely the "follow-through strategy" and the "re-evaluation strategy" as follows.

Follow-through strategy: Table 2 shows the system dynamics with the follow-through strategy. After a plan is selected for execution, the action sequence of the plan is executed from the beginning to the end. In other words, the agent follows the entire sequence of actions before it performs sensing again. An agent using the follow-through strategy is called a bold agent by Kinny and Georgeff (1991), that never reconsiders its options during the execution of a plan. This is in contrast to a cautious agent that reconsiders every new options in every step. Although it incurs the minimal reasoning cost, the follow-through strategy may not be applicable in a dynamically changing environment.

Re-evaluation strategy: To strike a balance between a bold agent and a cautious agent, a re-evaluation strategy is developed which has a moderated degree of commitment to plans. This strategy is similar to the follow-through strategy, except that an extra sensing is performed half-way through the execution of the plan. In other words, the agent performs another round of sensing after executing the first half of the actions specified in the selected plan. This is to enable the agent to evaluate the applicability of the current plan by comparing the next action (as specified by the action sequence) with the action selected by FALCON's direct access method. The adopted plan continues if the next action of the plan coincides with the action selected by the reactive module.

The overall behaviour of the agent is similar for the two plan adoption strategies. In contrast to the traditional sense–act–learn cycle, the agent now performs a sensing and follows either a plan selected from the intention module or an action selected by the direct code access procedure of the FALCON module. The preference of following a plan over executing an action selected by the low level FALCON is consistent with the behaviour of the subsumption architecture (Brooks, 1999), wherein the outputs of high level modules subsume those of low level modules. After performing a plan or an action through the motor channel, evaluation and learning proceed within the module used in selecting the plan or action.

4. Case study on minefield navigation

The minefield navigation task (Fig. 3) requires an autonomous vehicle (AV) starting at a randomly chosen position in the field to navigate through the minefield to a randomly selected target position in a specified time frame without hitting a mine. A trial ends when the system reaches the target (success), hits a mine (failure), or runs out of time.

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 $s_i = \frac{1}{d_i}$, 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 range and the bearing of the target from the current position. In each step, the system can choose one out of the five possible actions, namely MoveLeft, MoveFrontLeft, MoveFront, MoveFrontRight, and MoveRight.

Although the minefield navigation task seems similar to the obstacle avoidance task in grid world, which is typically used in evaluating AI and reinforcement learning systems, we note that the minefield navigation domain is different from the grid world in important aspects. Firstly, whereas the configuration of the gird world is typically fixed within an experiment, the minefield configuration is generated randomly for each learning trials in the experiment and thus an agent needs to learn and bring over knowledge

Table 2
Dynamics of the BDI-FALCON agent with the follow-through strategy.

1. Initialize the agent's system state
2. Obtain the current state by sensing the environment
3. Record the current state as the start state. Initialize the action memory array to NULL
4. Execute a plan or an action.
   - 4.1. Plan Execution: If a qualified plan is found, the start state and the action memory are cleared and the action sequence encoded by the selected plan is adopted for execution
     - 4.1.1. When the action sequence is completed, the agent senses the environment and computes the goal attainment function for plan evaluation
     - 4.1.2 If end of trial and a positive outcome is reached, the confidence of the adopted plan is increased
     - 4.1.3. If end of trial and a negative outcome is received, the confidence of the adopted plan is decreased accordingly. A plan that has a lower confidence value than the threshold is removed from the repository
     - 4.1.4. If not end of trial, go to step 2
   - 4.2. Reactive Action Execution: If no plan is found, FALCON is used to select an action
     - 4.2.1. Append the selected action into the action memory and perform the selected action
     - 4.2.2. After performing the action, compute the goal attainment function for FALCON to perform reactive learning
     - 4.2.3. If end of trial and a goal state is reached, perform plan learning by recording the start state, goal state, and action memory into a new plan and add it into the repository
     - 4.2.4. If not end of trial, obtain the current state by sensing the environment. Go back to step 4

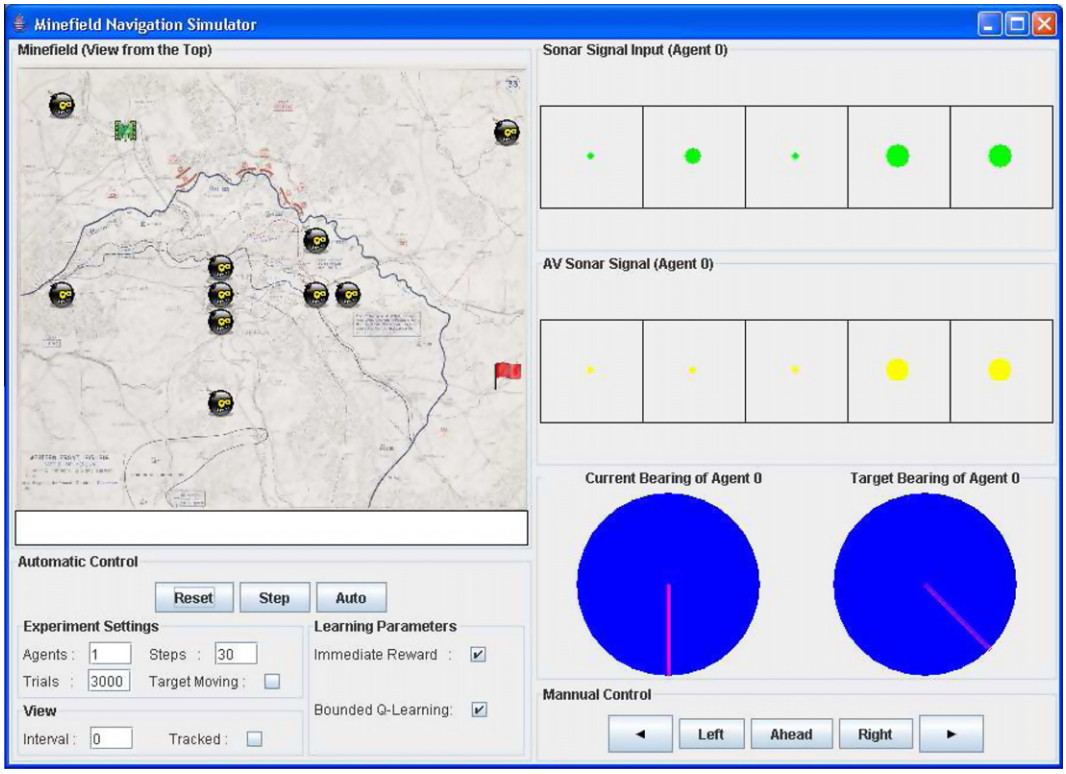


Fig. 3. The minefield navigation simulator.

that is applicable in a new minefield configuration. Secondly, in the minefield domain, the agent neither represents nor possesses the knowledge of its position in terms of x and y coordinates. Instead, it relies on a rather coarse sensing capability based on five sonar sensors in five directions to determine its immediate environment. This is a more realistic setting for real world embodiment. Thirdly, the agent here is not provided with any knowledge or plans, but is tasked to learn the knowledge through reinforcement signals received. Due to the limited sensory capability, the domain, though seemingly simple, is not easy to learn. Our prior experiments have shown that alternative gradient-descent based reinforcement learning systems took more than 30,000 trials to attain reasonable performance (Tan et al., 2008).

4.1. Goal representation and matching

In contrast to traditional reinforcement learning agents which have no explicit representation of goals, the BDI-FALCON architecture provides an explicit implementation of goals and goal attainment evaluation. For the minefield problem, we define two main goals as follows.

- Goal 1: Agent reaches the target.
- Goal 2: Agent maintains its life value at the maximum.

Based on the two goals defined, the goal target vectors should consist of two key attributes, namely *distance* indicating the remaining distance towards the target and *life* indicating the energy level of the agent. A goal target vector can thus be defined as

$$\mathbf{w}^{gj} = \langle D, \overline{D}, L, \overline{L} \rangle,$$

where D and L are the normalized values of the remaining distance and the life value, with $\overline{D}$ and $\overline{L}$ as their complements, respectively.

For goal 1, the corresponding goal target vector will have a value of 0 for the first element, indicating that the desired remaining distance to the target is zero. The second element, as the complement of the first value, is set to 1. The rest of the elements not relevant to the goal are set to 0. Similarly for goal 2, the goal target vector assigns a value of 1 for the third element, indicating a maximum life value. The other elements are set to 0. The two goal target vectors are thus given by

$$\begin{aligned} &\mathbf{w}^{g1} = (0.0, 1.0, 0.0, 0.0) \quad \text{and} \\ &\mathbf{w}^{g2} = (0.0, 0.0, 1.0, 0.0). \end{aligned} \tag{8}$$

In each reaction cycle, the states of the goal attributes can be obtained as follows. The distance to the target can be retrieved through the sensory input signals supplied by the maze. The life value is initialized to the user defined value and decremented by a certain value each time the agent runs into a mine. The raw values of distance and life are then normalized to the range of [0,1] before assigning to the goal state vector.

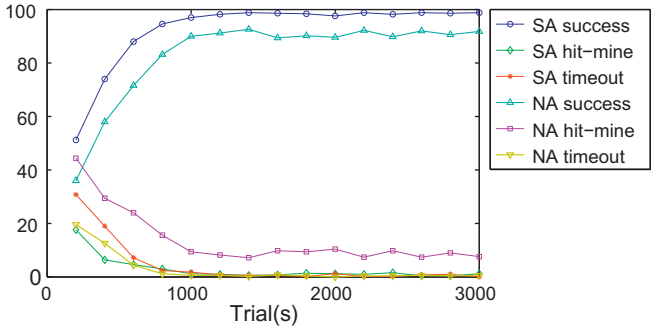


Fig. 4. The performance of the normal agents (NA) and the super agents (SA) in terms of success rates, hit-mine rates and out-of-time rates.

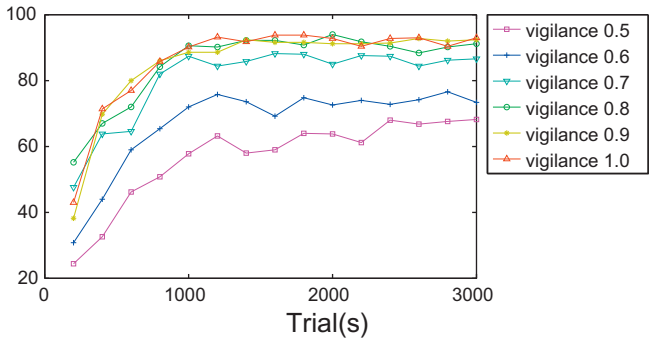


Fig. 5. The success rates of BDI-FALCON over 3000 trials using different plan vigilance values.

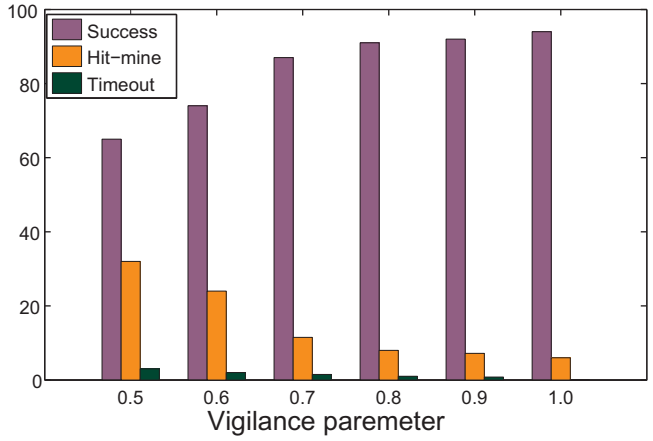


Fig. 6. The overall performance of BDI-FALCON at 3000 trials using different plan vigilance values.

With the goal state vector and the goal target vectors, the degree of goal attainment can be computed using the goal attainment function (Eq. (5)). This function calculates the matching value of the current goal state with the goal target vector of each goal and returns the overall goal attainment as the product of the individual match values of all the active goals.

Based on the above goals, we experimented with two sets of agents with different surviving capability. The first type of normal agents has a fragile life. Each time it runs into a mine, it gets blown up and the life value drops immediately to zero. The second type of super agents has a higher tolerance. Each time it hits a mine, the life value is reduced by 50%. Thus the agent gets a second chance after hitting a mine.

Referring to Fig. 4, we can see that there are significant performance differences between the two classes of agents. The normal agents produce performance at a level which is consistent with our prior experiments with FALCON models learned using external reinforcement signals (Tan, 2007). Our experiments thus verify that using the desire module, the agents can generate reinforcement signals internally according to how they are affected by the environment and are capable of learning effectively based on these internally generated reward signals.

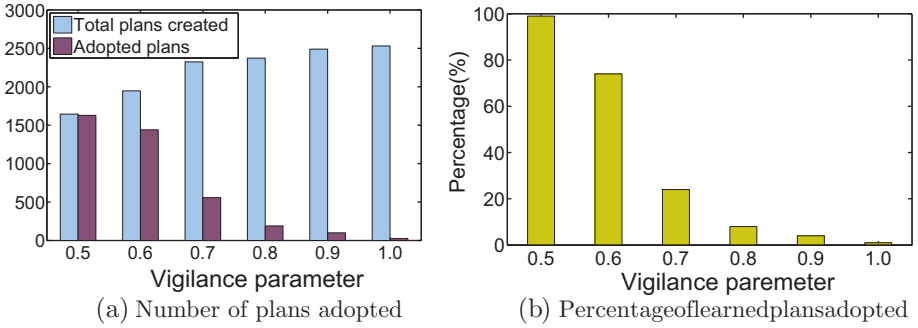
On the other hand, the super agents require a significantly smaller number of trials to achieve 90% success rate and achieve a high success rate of 98% at 3000th trial. The results are intuitive as the super agents still have a second chance of success after hitting a mine. More importantly, we note that the life value of a super agent drops to 0.5 instead of 0 upon hitting a mine and this causes its goal attainment function to be computed differently compared to that of a normal agent. Despite this, our experimental results of normal and super agents show that we are able to build various agents with different levels of capabilities and behaviour using the same set of goal encoding scheme and goal attainment function as defined in the desire module.

4.2. Plan learning and adoption

We set out to study how plans are learnt and used in the minefield navigation task. To this end, we experiment with various values of the plan vigilance parameter ρ^p and evaluate the system in terms of success rates and the number of plans created and adopted. The experimental results are obtained by averaging over five sets of 3000-trial runs with the plan vigilance value varying from 0.5 to 1.0.

As shown in Fig. 5, the success rates of BDI-FALCON generally drop as the plan vigilance decreases. With a plan vigilance of 0.8 and above, the system performance is roughly comparable to that of the original TD-FALCON system. However, much poorer success rates are observed with a plan vigilance value of 0.7 and below.

As presented in the previous section, a normal agent using TD-FALCON (with no plans) achieves a success rate of 91.2% after 3000 trials. Referring to the detailed performance of BDI-FALCON (Fig. 6), we note that BDI-FALCON with a plan vigilance of 1 achieve a better success rate of 93% after 3000 trials. Although we expect the system performance to drop with the plan vigilance value decreases, BDI-FALCON with a plan vigilance of 0.8, still achieves a success rates of 91.2%, the same as that of TD-FALCON. These results are encouraging considering that prior experiments on the same minefield domain using the PGS system (Karim & Sonenberg et al., 2006) actually found a slight degradation in performance, comparing with the original reactive FALCON system. Specifically, the PGS system (Karim & Sonenberg et al., 2006) reported a 89.4% success rate using FALCON and 88.6% using their BDI-FAL-



(a) Number of plans adopted (b) Percentageoflearnedplansadopted

Fig. 7. Number and percentage of plans learned and adopted.

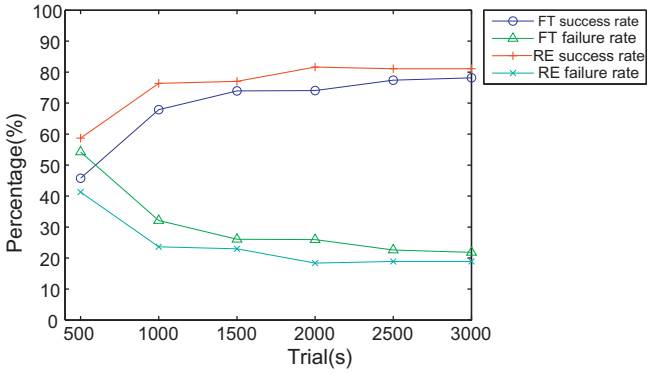


Fig. 8. The success and failure rates of the plans as used in the follow-through (FT) and re-evaluation (RE) strategies.

CON hybrid after 1000 trials. As another point of comparison, BDI-FALCON achieves success rates of 90.6% and 90.2% with plan vigilance values of 0.8 and 1.0, respectively, comparing with a success rate of 90.6 by TD-FALCON, recorded after the first 1000 trials. The results thus indicate that the fuzzy match-based plan selection and learning algorithms as used in the current system enables rules to be learned and used more effectively, comparing with a conventional BDI plan selection and execution module.

In Fig. 7, we note that the number of plans adopted is inversely proportional to the plan vigilance value. This is expected as the plan vigilance parameter determines the level of similarity a selected plan needs to be applicable. A higher vigilance would mean a lower chance for the plan to be adopted. On the other hand, the size of the plan repository is proportional to the plan vigilance parameter value. This is also not surprising as a high vigilance value will cause more existing plans to be rejected, raising the chance of creating new plans.

Note that it is not our objective to seek specific parameter values for producing optimal performance. Instead, we prefer that the system is able to produce robust performance over a wide range of parameter settings. Based on our experimental results, a plan vigilance value of 0.8 appears to produce a balanced set of results as it yields a comparable performance with the original system in terms of success rates and at the same time, provides a reasonable level (around 10% to 15%) of plan adoption. The vigilance level is thus used in the subsequent experiments as reported in the next sections.

4.3. Plan analysis

Besides the overall success rates, we are interested in how the adopted plans have actually contributed to the outcomes. In Fig. 8, we compare the percentages of successful and failure trials, in which plans are used, for the follow-through and re-evaluation strategies. The objective is to examine the quality of plans, in terms of the outcomes following their use. As a relatively small number of plans are invoked over the 3000 trials, the results are averaged at 500-trial interval across ten simulation runs.

At a micro-level, within the set of those plans that lead to successful trials, we identify how many of them directly lead to the target and how many only contribute partially to the succeeded path. The types of contribution are categorized into four main categories as follows.

1. Plans that directly lead to the target.
2. Plans that contribute partially to the successful path.
3. Plans that lead to hit-mine failure.
4. Plans that lead to timeout failure.

As the number of plans adopted in the two strategies are roughly the same, we can use the percentage to compare the plan contribution in the following experiment. Referring to Fig. 9, within the set of adopted plans, the two strategies have very similar percentage of plans that directly lead to the target but the re-evaluation strategy has a noticeable higher percentage for those plans with partial contribution towards the target. Correspondingly, among the adopted plans, the re-evaluation strategy has a lower percentage of time-out and hit-mine failures. Therefore, the success rates using the re-evaluation strategy are generally better than those using the follow-through strategy. The results are not surprising as the follow-through strategy, which is rather simplistic, is unlikely to work well in a complex and dynamics environment. On the other hand, re-evaluation is considered as more robust than follow-through, as it enables the agent to reassess the applicability of its currently executing plan and to generate a more accurate response to the situation, at a small cost of additional sensing.

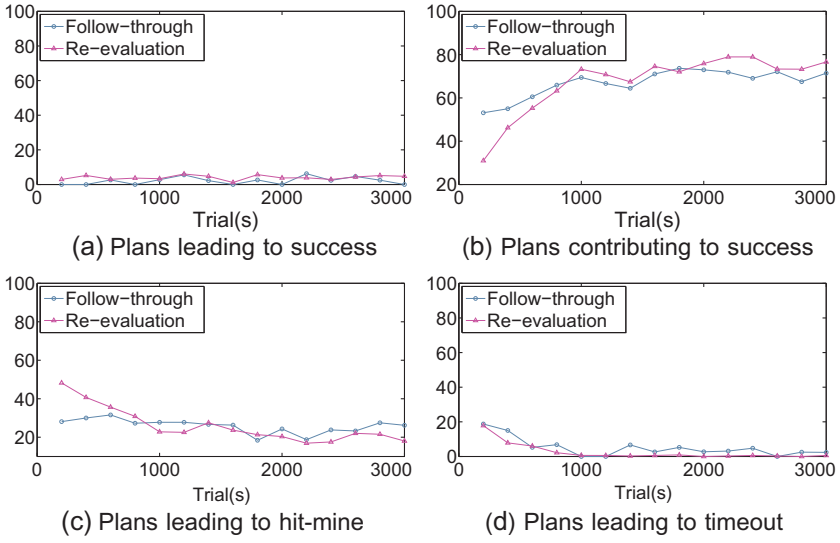


Fig. 9. Proportion of plans leading to various outcomes.

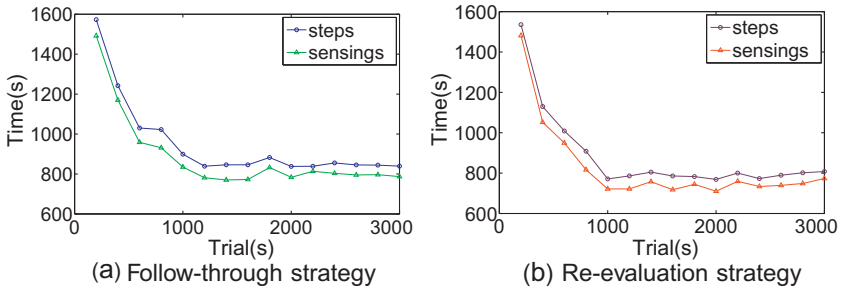


Fig. 10. Number of steps and sensing with a plan vigilance of 0.8.

4.4. Plan efficiency

An important motivation of using plans is the reduction in the frequency of periodic sensing as in the sense-act-learn cycles. In this section, we compare the efficiency of the two strategies in terms of the number of sensing they make. For the follow-through strategy, there is one sensing at the beginning of the plan selection process. The sensory inputs are used to choose the most suitable plan for the particular situation represented by those signals. After that, there is no more sensing during plan execution until all actions in the plan have been performed. For the re-evaluation strategy, one sensing at the beginning of the plan selection process is needed in the same way as the first strategy. In addition, another sensing is required half-way through the plan's execution.

Fig. 10 shows the number of sensing needed by each strategy with a plan vigilance of 0.8. It can be observed that, for both strategies, the percentage of reduction in sensing is approximately 10% of the total number of sensing, with the total number of sensing at the 3000th trial by the re-evaluation strategy (800) slightly less than that of the follow-through strategy (850). This result is interesting as we would expect the re-evaluation strategy (which introduces an extra sensing per plan) to incur more number of sensing than the follow-through strategy. However, as the quality of the solution by the re-evaluation strategy is generally better than that of the follow-through strategy, the numbers of steps and sensing are correspondingly lower than those of the follow-through strategy.

5. Conclusion

The BDI-FALCON hybrid architecture is an integration of BDI and TD-FALCON for their complementary strengths of desire, intention and reinforcement learning. By adopting the same computing principles in the desire, intention and reactive learning modules, we aim to offer a consistent treatment to the design of the integrated architecture.

The desire module maintains an explicit representation of an agent's goals, raising its self-awareness of the underlying reasons of its actions. Having an explicit desire module, in addition, provides the flexibility to define and refine goals, and therefore makes the system more dynamic and adaptive. Furthermore, with the organized set of goals and goal attainment evaluation methods, the agent can supply itself with the reinforcement signals generated internally. Our minefield navigation experiments have demonstrated the plausibility of defining agents with different levels of capability and behaviours.

The intention module equips the system with a deliberative planning capability. It provides an alternative approach enabling the system to function in a more hostile and dynamic environment, wherein the purely reactive sense-act-learn cycles are not always adequate. Following plans reduces the need for periodic sensing and therefore improves efficiency. We have experimented with two plan execution strategies, of which the re-evaluation strategy seems to yield better results in terms of a higher positive contribution among the adopted plans and a better overall success rate.

Moving forward, there remain many challenges in the intention module, especially on the aspects of plan utilization and management. Currently, our plans are stored in a simple one-dimensional symbolic structure. As the plan database grows, more computation cost will be incurred in the plan selection process. Advanced data structures, such as trees or massively parallel neural networks, may be used to enhance the efficiency in storing and searching plans. But before plans can be efficiently organized, the plan representation may need to be revised. In the current implementation, a plan corresponds to a path towards the target. Learning of exact paths however limits the plan's applicability to very specific situations. Acquiring abstract plan representation through techniques, such as sequence learning (Sun & Giles, 2000), is thus an important research direction. Generalizing a plan into a higher level of abstraction will also make the plans more versatile in handling a wider variety of circumstances and keep the size of the plan repository small.

For the desire module, a more sophisticated goal representation will certainly enhance the agent's performance in complex problem solving. Specifically, goal-subgoal hierarchy is one important feature that is missing from our current implementation. By the ability of decomposing a complex goal into simpler subgoals, the latter can be achieved one at a time, laying the path towards the achievement of the primary goal. The concept of goal decomposition will enable the desire module to be more adaptive, in the same manner as the reactive and intention modules, by allowing goals to be created, manipulated and evaluated dynamically.

The last and not the least, we aim to expand our experimental study from the current minefield navigation domain into a publicly available benchmark problem. It is our intention that, through the common platform, we will be able to evaluate the proposed hybrid agent architecture with other conventional BDI systems side-by-side in a case study framework.

Acknowledgements

The reported work was supported in part by the Singapore National Research Foundation Interactive Digital Media R&D Program, under research Grant NRF2008IDM-IDM004-037. The authors thank Samin Karim and Liz Sonenberg for discussion and comments to a previous version of this paper.

References

Anderson, M. (2003). Embodied cognition: A field guide. *Artificial Intelligence, 149*, 91–130.

Anderson, J., Bothell, D., & Byrne, M. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036–1060.

Bratman, M., Israel, D., & Pollack, M. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence, 4*(4), 349–355.

Brooks, R. (1999). *Cambrian intelligence: The early history of the new AI*. MIT Press [A Bradford Book].

Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing, 37*, 54–115.

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks, 4*, 759–771.

Gordan, D., & Subramanian, D. (1997). A cognitive model of learning to navigate. In *Proceedings of nineteenth annual conference of the Cognitive Science Society* (pp. 271–276).

Guerra-Hernandez, A., Fallah-Seghrouchni, A. E., & Soldano, H. (2004). Learning in BDI multi-agent systems. In *Proceedings, fourth international workshop on computational logic in multi-agent systems, Fort Lauderdale, FL*.

Kaelbling, L., Littman, M., & Moore, A. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research, 4*, 237–285.

Karim, S., Sonenberg, L., & Tan, A.-H. (2006). A hybrid architecture combining reactive plan execution and reactive learning. In *Proceedings of the 9th Biennial Pacific Rim international conference on artificial intelligence (PRICAI'06), LNAI 4099, Guilin, China* (pp. 200–211).

Karim, S., Subagdja, B., & Sonenberg, L. (2006). Plans as products of learning. In *IAT'06: Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology* (pp. 139–145).

Kinny, D., & Georgeff, M. P. (1991). Commitment and effectiveness of situated agents. In *Proceedings of twelfth international joint conference on artificial intelligence* (pp. 82–88).

Lebiere, C., & Wallach, D. (2000). Sequence learning in the act-*r* cognitive architecture: Empirical analysis of a hybrid model. In R. Sun & C. Giles (Eds.), *Sequence Learning. LNAI* (Vol. 1828, pp. 188–212). Springer-Verlag.

Norling, E. (2004). Folk psychology for human modelling: Extending the BDI paradigm. In *Proceedings, AAMAS'04, New York, USA* (pp. 202–209).

Pollack, M. E. (1992). The uses of plans. *Artificial Intelligence, 57*, 43–68.

Si, J., Barto, A. G., Powell, W. B., & Wunsch, D. (Eds.). (2004). *Handbook of learning and approximate dynamic programming*. Wiley-IEEE Press.

Subagdja, B., & Sonenberg, L. (2005). Learning plans with patterns of actions in bounded-rational agents. In *Proceedings of the 9th international conference, KES 2005* (Vol.3, pp. 30–36).

Sun, R. (2000). Beyond simple rule extraction: The extraction of planning knowledge from reinforcement learners. In *Proceedings of the international joint conference on neural networks, Como, Italy* (pp. 24–27). Piscataway, NJ: IEEE Press.

Sun, R., & Giles, C. (Eds.). (2000). *Sequence learning: Paradigms, algorithms, and applications. LNAI* (Vol. 1828). Springer-Verlag.

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

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Tan, A.-H. (1995). Adaptive resonance associative map. *Neural Networks, 8*(3), 437–446.

Tan, A.-H. (2004). FALCON: A fusion architecture for learning, cognition, and navigation. In *Proceedings of the international joint conference on neural networks* (pp. 3297–3302).

Tan, A.-H. (2007). Direct code access in self-organizing neural architectures for reinforcement learning. In *Proceedings of the international joint conference on artificial intelligence (IJCAI07), Hyderabad, India* (pp. 1071–1076).

Tan, A.-H., & Xiao, D. (2005). Self-organizing cognitive agents and reinforcement learning in a multi-agent environment. In *Proceedings of the IEEE/WIC/ACM international conference on intelligent agent technology (IAT'05), France* (pp. 351–357).

Tan, A.-H., Carpenter, G. A., & Grossberg, S. (2007). Intelligence through interaction: Towards a unified theory for learning. In *Proceedings of ISNN, LNCS* (Vol. 4491, pp. 1098–1107).

Tan, A.-H., Lu, N., & Xiao, D. (2008). Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evaluative feedback. *IEEE Transactions on Neural Networks, 9*(2), 230–244.

Tan, A.-H., & Pan, H. (2005). Predictive neural networks for gene expression data analysis. *Neural Networks, 18*(3), 297–306.

Wallis, P. (2004). Intention without representation. *Journal of Philosophical Psychology, 17*(16), 209–224.

Watkins, C., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*(3/4), 279–292.

Xiao, D., & Tan, A.-H. (2007). Self-organizing neural architectures and cooperative learning in multi-agent environment. *IEEE Transactions on Systems, Man, and Cybernetics - Part B, 37*(6), 1567–1580.