

# A Self-Organizing Neural Network Architecture for Intentional Planning Agents

Budhitama Subagdja  
Intelligent Systems Centre  
Nanyang Technological University  
budhitama[at]ntu.edu.sg

Ah-Hwee Tan  
School of Computer Engineering  
Nanyang Technological University  
asahtan[at]ntu.edu.sg

## ABSTRACT

This paper presents a model of neural network embodiment of intentions and planning mechanisms for autonomous agents. The model bridges the dichotomy of symbolic and non-symbolic representation in developing agents. Some novel techniques are introduced that enables the neural network to process and manipulate sequential and hierarchical structures of information. It is suggested that by incorporating intentional agent model which relies on explicit symbolic description with self-organizing neural networks that are good at learning and recognizing patterns, the best from both sides can be exploited. This paper demonstrates that plans can be represented as weighted connections and reasoning processes can be accommodated through multi-directional activations across different modalities of patterns. The network seamlessly interleaves planning and learning processes towards achieving the goal. Case studies and experiments shows that the model can be used to execute, plan, and capture plans as recipes through experiences.

## Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Intelligent agents; I.2.8 [Problem Solving, Control Methods, and Search]: Plan execution, formation, and generation; I.2.6 [Learning]: Connectionism and neural nets

## General Terms

Theory, Design, Experimentation

## Keywords

BDI agent, fusion ART, iFALCON

## 1. INTRODUCTION

Being able to function and adapt in a complex changing environment is a key characteristic of an autonomous agent. The agent must cope with risks and surprises by continuously reasoning, learning, and planning. The agent plans by constructing a description of a course of action that may solve a given problem or achieving some goals. The output of planning is a *plan* or a *recipe* that the agent can execute

**Cite as:** A Self-Organizing Neural Network Architecture for Intentional Planning Agents, Budhitama Subagdja and Ah-Hwee Tan, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

Copyright © 2009, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

to achieve its goals. When time or resources for planning are limited, the criteria can be made less restrictive by assuming that some partial plans are provided at the design time hence the planning task becomes assembling the plans at runtime. This kind of alternative to plan is a common approach in intentional agents architecture (such as beliefs-desires-intentions (BDI) model [13]). In any case, planning agents commonly rely on extensive use of explicit symbolic representation of plans. The performance of a resource-bounded agent, in particular, depends mainly on careful analysis and design by the developer.

On the other hand, in non-symbolic realms of modeling, uncertainties, possible risks, and lack of knowledge are usually addressed through processes of learning. The information and knowledge are processed and represented implicitly governed by ad hoc calculations with less interventions from the developers or users. Neural networks, as a kind of non-symbolic architecture, are inspired by properties of biological brain. Given the appropriate samples, a neural network model can learn to solve problems or generalize the outputs given new inputs. Although they inherently support learning and pattern recognition, they are still considered to represent the knowledge implicitly. A neural network agent may be able to automatically learn and update its knowledge by experience, but it may also be difficult to ensure that its behavior conforms with some explicitly prescribed instructions.

In this paper, it is argued and demonstrated otherwise that symbolic and non-symbolic approaches support and enrich each other in realizing intentional agents. It is possible to construct a neural network model to realize an intentional agent that accepts and maps symbolic descriptions while being adaptive to different online situations. Moreover, the learning mechanism can use the plan representation as the target knowledge as well. The knowledge captured through the learning include hierarchical structure of plans. Therefore, there is no need to separate the planning from learning processes as it is unnecessary to separate pattern recognition tasks from the respective neural weights adjustment. This approach may also bridge agent models specified in formal symbolic descriptions with their plausible neural embodiments.

The rest of the paper is organized as follows. Section 2 discusses the use of plans and the process of planning in autonomous agents. It provides an overview of intentional agent architectures and how they deal with resources limitations. Section 3 explains and demonstrates how the plan representation and planning mechanisms can be realized in

a self-organizing neural network. Based on a multi-channel adaptive resonance theory network, novel techniques are introduced for computing sequential and hierarchical structures of information. It also includes discussions on how deliberation, planning, and learning processes can be combined together in the neural model. Section 4 shows some results from our experiment of applying the self-organizing neural architecture. Section 5 summarizes and concludes the paper.

## 2. PLANS AND PLANNING AGENTS

Planning is the process of deciding how to achieve an end using the available means. Given the current goals, beliefs about the state of the environment, and a set of actions available, the agent generates a plan as a course of action. Planning can range from assembling a complete course of action which consists of atomic components of the actions available (first principle) to rather selecting and executing predefined plans from a repository of recipes. The appropriate allocation between planning and execution depends on the kind of tasks and environment. The main problem of planning from the first principle is that it is very costly to be applied for agents that make decisions, plan, and act in real time [6].

In practice, a collection of partial plans is assumed to be pre-given offline and the planning task becomes interleaving the assembly and execution of the partial plans. This relaxed method of planning reduces the amount of resources required for reasoning about plans [2, 12]. In domains involving much interactions with the environment and unpredictable situations, feedbacks may also be needed to refine the existing plans to fit with the actual conditions [10]. Thus, plans can have two kinds of roles: as *recipes* or knowledge of actions to achieve certain goals, and as mental-attitudes that determine and characterize the agent's behavior [2, 11]. An agent whose behavior can be predicted and explained based on its mental attitudes can be said as *intentional* [5].

### 2.1 Intentional Agent Architecture

An intentional agent is characterized with mental attributes that are commonly used to characterize people. Derived from *folk psychology*, an intentional agent architecture such as a BDI (Beliefs, Desires, Intentions) [13] agent uses attributes of *beliefs*, *desires*, and *intentions* to direct its actions and selection of plans to achieve its goal. This kind of agent works as an interpreter that operates on those representations of mental attitudes.

*Beliefs* or *belief base* is a data structure that corresponds to a model or knowledge about the world and/or about the agent itself. In most implemented intentional agents, *beliefs* are predicate logic statements or propositions about what currently holds or does not hold (negation) in binary logic (true or false). *Desires* is a set of goal condition that the agent wants to achieve. As computational resources are limited, the agent must choose *intentions* as the state of affairs the agent has decided to pursue.

An intention is a plan as a mental attitude [11]. It is a decision comprising the goal and actions to take. The decision may recurse so that the agent may select intentions at a lower level of abstraction and so on, until arrives at the one that is directly executable. In Procedural Reasoning System (PRS) [8], as one of the first implemented model of BDI agent, *intentions* are stored in the *intention structure*

which has a hierarchical stack structure. The stack contains selected goals that are pending achievement and the current state of the execution.

As a recipe, a plan is a procedural knowledge structure that specifies how to achieve certain goals. The plan typically has the following attributes:

- *Goal*. The plan goal is the postcondition that will hold after the plan is executed.
- *Precondition*. The plan precondition specifies the condition that must be held so that the plan is applicable.
- *Body*. The plan body describes the course of actions that will be executed if the plan is adopted.

When a plan has a goal attribute that matches with current goals and its precondition is satisfied, then the plan can be considered as an applicable plan. A plan to be executed is selected from applicable plans. The selected goals and plans to be committed for execution are retained as *intentions*. The committed intentions provide constraints on further deliberations and the process of finding ways to achieve goals.

### 2.2 Planning and Bounded-Rationality

The main challenge in building a BDI agent is to ensure the right portions of time for deliberation and the appropriate strategy to maintain commitment given the fact that the deliberations are costly. Deliberations are computational processes that might take up resources which can reduce the agent's responsiveness to change. This issue is known as *bounded rationality*: a property of an agent that always functions appropriately while computational resources are scarce [15].

The use of prescribed knowledge for actions is the key aspect contributing to bounded rationality. Plans as recipes can be seen as heuristics that limit the search for appropriate actions. When plans are adopted as intentions, they constrain further considerations of alternatives and thus reduce the need for extensive computational power in the decision making process. Another approach is to control the commitment to execute the intention by choosing whether to maintain the commitment or just drop it and restart the deliberation [9, 14].

Thus, processes of an intentional agent has certain aspects that determine the quality of its decision to achieve the goal:

- *Deliberation*. The strategy used to select which goal and plan to adopt obviously determine the appropriateness of the decision.
- *Plan representation*. The prescribed plans determine which part of goals should be focused on and when. Thus, plans determine the effectiveness of subsequent decision making.
- *Commitment*. Based on the current intention, commitment strategy determine whether to drop the intention or keep it on when certain events occur. The commitment determine that the agent behave appropriately when changes happen in the environment.

This paper argues that those aspects are also supported in the proposed neural network model. The inherent features of learning and generalization of the neural network support even more aspects.

### 3. SELF-ORGANIZING NEURAL NETWORK FOR PLANS

Some of the challenges in using neural networks as the basis for intentional agents can be described as follows:

- Sequential processing. A representation of plan comprises a description of a sequence of actions.
- Hierarchical processing. Intentions can be structurally recursive consisting goals and subgoals relations.
- Learning and performance integration. As an architecture for agent, the model should operate dynamically by experience, integrating all aspects of decision making and execution.

The proposed neural architecture in this paper is based on Adaptive Resonance Theory (ART) [3] as a family of neural architectures that uses the principles derived from an analysis of human and animal cognitive information processing. It is a class of self-organizing neural networks that can learn continuously in a fast but a stable manner. As a distinct feature of ART, there is no separation between learning phase and activation phase, which partially solves the list of challenges mentioned above.

In particular, the proposed model extends a derivation of ART called fusion ART that employs multiple channels of input/output [17]. This multi-channel ART has been proven to integrate different types of learning and different modes of operations in a single architecture [16, 17, 18]. The model of intentional agents presented, extends the fusion ART model to deal with sequences and hierarchical processing comprising common operations for intentional agents.

#### 3.1 Fusion Adaptive Resonance Theory

The basic ART model works as an unsupervised learning system. Unlike other standard neural network models, an ART network classifies the input pattern by searching for a resonance condition: the category selected actually matches the input pattern. The search process requires bi-directional activations which are bottom-up from the input pattern to select a category and top-down from the selected category to measure the matching level of the respective input pattern. Another distinct feature of ART model is that when the search cannot find a resonance, a new category is allocated to fit with the input pattern, allowing the network to grow.

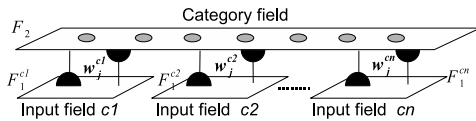


Figure 1: Fusion ART architecture.

The fusion ART employs multiple input fields each may use independent parameters and different encoding schemes (Figure 1). This approach enables the network to process different input modalities, thanks to the approach of integrating Fuzzy Logic and ART [4]. The Fuzzy ART network applies complement coding to cover a wider range of input pattern. Different channels can also be used as outputs so that different operations and learning mechanisms (such as supervised and reinforcement learning) can be applied [17].

The particular structure of multi-channel ART can be described as follows.

**Input vectors and fields:** Let  $\mathbf{I}^{ck} = (I_1^{ck}, I_2^{ck}, \dots, I_n^{ck})$  denote an input vector, where  $I_i^{ck} \in [0, 1]$  indicates the input  $i$  to channel  $ck$ , for  $k = 1, \dots, n$ . With complement coding, the input vector  $\mathbf{I}^{ck}$  is augmented with a complement vector  $\bar{\mathbf{I}}^{ck}$  such that  $\bar{I}_i^{ck} = 1 - I_i^{ck}$ . Let  $F_1^{ck}$  denote an input field as a buffer that holds the input pattern for  $ck$  and  $\mathbf{x}^{ck} = (x_1^{ck}, x_2^{ck}, \dots, x_n^{ck})$  is the activity vector of  $F_1^{ck}$  that holds the input vector (including the complement).

**Category fields:** Let  $F_i$  denote a category field and  $i > 1$  indicate that it is the  $i$ th field. The standard multi-channel ART has only one category field which is  $F_2$  and  $\mathbf{y} = (y_1, y_2, \dots, y_m)$  is the activity vector of  $F_2$ .

**Weight vectors:** Let  $\mathbf{w}_j^{ck}$  denote the weight vector associated with the  $j$ th node in  $F_2$  for learning the input pattern in  $F_1^{ck}$ .

**Parameters:** Each field's dynamics is determined by choice parameters  $\alpha_i^{ck} \geq 0$ , learning rate parameters  $\beta_i^{ck} \in [0, 1]$ , contribution parameters  $\gamma_i^{ck} \in [0, 1]$  and vigilance parameters  $\rho_i^{ck} \in [0, 1]$ .

The dynamics of a multi-channel ART can be considered as continuous resonance search processes which consist of basic operations as follows.

**Code activation:** A node  $j$  in  $F_2$  is activated by the choice function  $T_j = \sum_{k=1}^n \gamma^{ck} \frac{|\mathbf{x}^{ck} \wedge \mathbf{w}_j^{ck}|}{\alpha^{ck} + |\mathbf{w}_j^{ck}|}$ , where the fuzzy AND operation  $\wedge$  is defined by  $(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i)$ , and the norm  $|\cdot|$  is defined by  $|\mathbf{p}| \equiv \sum_i p_i$  for vectors  $\mathbf{p}$  and  $\mathbf{q}$ .

**Code competition:** A code competition process follows to select a  $F_2$  node with the highest choice function value. The winner is indexed at  $J$  where  $T_J = \max\{T_j : \text{for all } F_2 \text{ node } j\}$ . When a category choice is made at node  $J$ ,  $y_J = 1$ ; and  $y_j = 0$  for all  $j \neq J$  indicating a winner-take-all strategy.

**Template matching:** A template matching process checks if resonance occurs or specifically if for each channel  $ck$ , the match function  $m_j^{ck}$  of the chosen node  $J$  meets its vigilance criterion:  $m_j^{ck} = \frac{|\mathbf{x}^{ck} \wedge \mathbf{w}_j^{ck}|}{|\mathbf{x}^{ck}|} \geq \rho^{ck}$ . If any of the vigilance constraints is violated, mismatch reset occurs or  $T_j$  is set to 0 for the duration of the input presentation. Another  $F_2$  node  $J$  is selected using choice function and code competition until a resonance is achieved. If no selected node in  $F_2$  meets the vigilance, an uncommitted node is recruited in  $F_2$  as a new category node selected by default.

**Template learning:** Once a resonance occurs, for each channel  $ck$ , the weight vector  $\mathbf{w}_j^{ck}$  is modified by the following learning rule:  $\mathbf{w}_j^{ck(\text{new})} = (1 - \beta^{ck})\mathbf{w}_j^{ck(\text{old})} + \beta^{ck}(\mathbf{x}^{ck} \wedge \mathbf{w}_j^{ck(\text{old})})$ .

**Activity readout:** The chosen  $F_2$  node  $J$  may perform a readout of its weight vectors to an input field  $F_1^{ck}$  such that  $\mathbf{x}^{ck(\text{new})} = \mathbf{w}_j^{ck}$

#### 3.2 Invariance Principle

In the standard configuration of multi-channel ART, a node in the category field encodes the association between a node in the category field and nodes in input fields in a single period of time. However, a single binary-valued node can not associate different input patterns across different time. Although the category field allows incremental learning by

recruiting uncommitted nodes for new categories, no information about time or order of the pattern occurrences can be captured. Our proposed model extends the multi-channel ART model so that it can associate and group patterns across time.

A suitable way to capture temporal patterns has actually been suggested in some early works of the ART neural network model. Grossberg proposed the *Invariance Principle* that applies to a neural field to represent a serial order of activations which can be used to model how temporal patterns are stored in working memory [7]. The principle suggests that the ratio of values of codes from previous inputs remains invariant as new inputs enter the network. The values of individual codes are arranged in such a way that both the items and the temporal order in which they occurred are encoded by their activity pattern. The principle has been used in STORE (Sustained Temporal Order Recurrent) network that mimics the behavior of working memory based on cognitive data about short-term memory [1].

The invariance principle suggests that to retain the temporal order in a neural field, each entry of activation item multiplicatively modifies the activity of all previous items. By tuning up different parameters in the multiplicative function, different kinds of analog patterns may emerge in the field which reflect the order the activations are presented. In fact, the model has accurately emulated the characteristic of serial learning conforming the psychological data about human working memory.

### 3.3 Sequential Working Memory for Plans

Based on the principle of code invariance, a similar technique can be applied to fusion ART for dealing with sequential and hierarchical patterns and information. *Gradient encoding* is a simplified version of the invariance principle or the original STORE model in terms of how the activation values are set in the neural field. Simple constant increments (decrements) are used instead to determine each activation value. The simplification is used because fusion ART is developed not in order to mimic the behavior of human memory and the use of mere additions (subtractions) in gradient encoding are more practical for realizing the model.

To represent a sequential order in gradient encoding a value can simply indicate a time point or a position in an ordered sequence. To retrieve or reproduce the sequence, the larger activations are recalled first and hence represent codes that activate earlier. Suppose  $t_0, t_1, t_2, \dots, t_n$  denote time points in an increasing order, and  $y_{t_i}$  is a node value of the activity vector of the category field that is activated or selected at time  $t_i$ , the category field follows the gradient encoding if  $y_{t_0} > y_{t_1} > y_{t_2} > \dots > y_{t_n}$  holds.

**Sequence readout:** Let  $F_2$  be the category field applying the sequential encoding. The node with the highest activation is selected and indexed at  $J$ .  $T_J = \max\{T_j : \text{for all } F_2 \text{ node } j\}$ . The selected node then performs readout to the input fields so that  $\mathbf{x}^{ck(\text{new})} = \mathbf{w}_j^{ck}$ . After the readout, the node  $J$  is reset so that  $y_J = 0$  and the select-readout cycle continues until all nodes in  $F_2$  are zero.

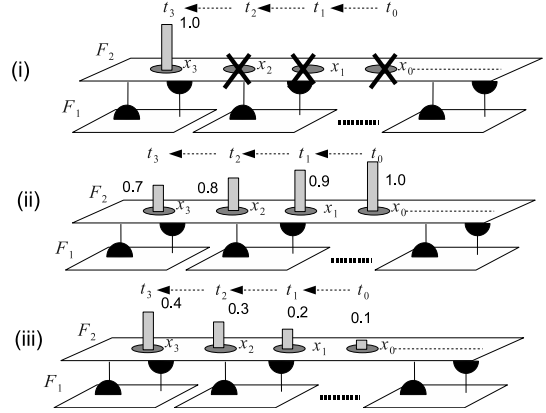
A sequence can be stored by setting the activation value of a selected node according to its order in the sequence relative to the previous selection. Depending on the functionality of the category field, different kinds of ordering can

be employed to store different sequential order. Given the time points  $t_i$  and a node value  $y_{t_i}$  that is activated at its respective time point, if the values are stored in an increasing order so that  $y_{t_0} < y_{t_1} < y_{t_2} < \dots < y_{t_n}$ , the field behaves like a stack or a FILO (First In Last Out) data structure. On the other hand, if the values are stored in a decreasing order so that  $y_{t_0} > y_{t_1} > y_{t_2} > \dots > y_{t_n}$ , the field acts like a queue or FIFO (First In First Out) data structure.

**First In Last Out:** To make the field  $F_2$  behaves as a First In Last Out memory, firstly a node is selected in  $F_2$  with the resonance search process. If the last selected node in  $F_2$  is  $J$ , the value of the node is set to the order parameter  $\tau$  such that  $y_J = \tau$  and then  $\tau^{(\text{new})} = \tau^{(\text{old})} + v$ .  $0 \leq \tau \leq 1$ .  $v$  is the increment (decrement) factor, and  $\tau$  is initialized at zero at the beginning of the sequence.

**First In First Out:** To make  $F_2$  as a First In First Out memory, the selected node value is set to  $\tau$ ,  $y_J = \tau$  and then  $\tau^{(\text{new})} = \tau^{(\text{old})} - v$ .  $\tau$  is initialized at one at the beginning of the sequence and  $0 \leq \tau \leq 1$ .

Figure 2 shows different types of sequential ordering in gradient encoding compared with the standard activation.



**Figure 2: (i) Standard code activation; (ii) First In First Out activation pattern, and (iii) First In Last Out activation pattern**

The gradient encoding technique enables all possible groupings of sequential events to be learnt in a stable and continuous manner. It can be used to leverage the multi-channel ART so that sequences and a temporary hierarchical structure can be learnt and processed. However, this sequential processing requires additional category fields with different types of connection. The additional fields categorizes or chunks sequences that are formed as analog patterns of activation in another category field.

### 3.4 iFALCON: A Neural-Based Intentional Agent Architecture

iFALCON is an intentional agent architecture built by arranging and grouping different fusion ART networks in a certain way. As shown in Figure 3 iFALCON consists of four input/output fields:  $F_1^{c1}$ ,  $F_1^{c2}$ ,  $F_1^{c3}$ , and  $F_1^{c4}$  denoting *beliefs*, *desires*, *critic*, and *action* fields respectively. The *beliefs* and *desires* fields correspond to the *beliefs* and *desires* components of the BDI model. The *action* field represents

the action to take at moment, while the *critic* field reflects the differences between the values in *beliefs* and *desires* field. Instead of just being connected to one category field like in fusion ART, these input (output) fields, are connected to two category fields:  $F_2^c$  and  $F_3^c$  denoting plans and *sequencer* fields which represent the current selected plan and the current executed action in a sequence respectively.  $F_3^c$  operates following FIFO (First In First Out) activation model. Further,  $F_2^c$  and  $F_3^c$  are connected to the top  $F_4^c$  field which operates following the FILO (First In Last Out) sequence model.

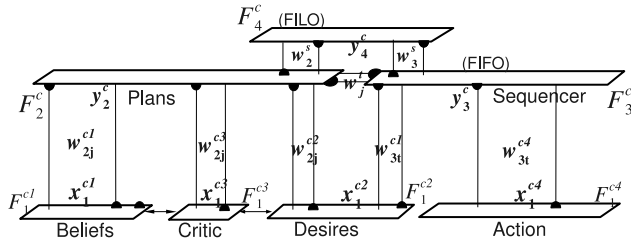


Figure 3: iFALCON

*Beliefs* and *critic* field are connected to the *plans* field, whereas *action* to the *sequencer*. *Desires* field is connected to both plans and sequencer field. There are extra connections between the *plan* and the *sequencer* category fields and each category field is further connected to a FILO (First In Last Out) field.

A plan can be encoded by setting up the appropriate weight values of the neural network connections. Nodes in *plans* and *sequencer* field represent plans and sequences respectively. An attribute of a plan is encoded as weight values in the connections between a plan node and the corresponding input/output field. The use of the two *plan* and *sequencer* fields allows hierarchical groupings of actions and a sequence into a single category node in the *plan* field. Each category node in *sequencer* encodes a pattern of action in the input/output fields. Through the extra connections between *plan* and *sequencer*, a node in *plan* can also encode a collection of actions represented as different activations in *sequencer*.

**Plan insertion by learning:** A new plan can be learnt by applying bottom-up activations to both  $F_2^c$  and  $F_3^c$ . While a node  $J$  in  $F_2^c$  is allocated and active, each action of the plan is presented subsequently one at a time to  $F_1^{c4}$  and/or  $F_1^{c2}$  to activate  $F_3^c$  to capture the sequence in FIFO mode. When the sequential pattern is totally formed in  $F_3^c$ , the sequence can be attached permanently to the plan by adjusting the connection weights between  $F_2^c$  and  $F_3^c$ .

When all plans are assumed to have been encoded appropriately as neural connections and nodes, it is possible to emulate the deliberation and execution cycle of BDI agent. The cycle can be described as follows:

**Critic evaluation:** At the beginning, the values in *beliefs* and the *desires* are compared with a match function  $m^c$ . If  $m^c$  is greater than a vigilance  $\rho$  the goal is satisfied and the process can be stopped. Otherwise, it continues to the next stage.

**Plan selection:** When the match is low, the deliberation

process is conducted by searching a *resonance* condition between the plan field  $F_2^c$  and some input/output fields (*desires*, *desires*, and *beliefs*). The resonance search proceeds to find and select a category node in  $F_2^c$  representing a plan, just like the search process described in the last subsection.

**Plan execution:** After a plan node is selected, the plan is executed by a *readout* process from the selected node in the *plans* field to *sequencer* and further down to the respective input/output fields. A sequence readout of FIFO model is conducted iteratively from *sequencer* to the *action* field, until the ordering parameter  $\tau \leq 0$ .

**Subgoal posting and backtracking:** During the execution process, a subgoal can be posted as a special type of action. A subgoal can be posted when a predefined node in  $F_1^{c4}$  is activated. The subgoal, which is encoded as connection values between a node in *sequencer* and nodes in *desires* field, overrides the activation values of *desires* which trigger a further plan selection process. However, before replacing the *desires*, the current active nodes in *plan* and *sequencer* field are stored into the  $F_4^c$  following FILO memory model. The stored plan in  $F_4^c$  can be restored by conducting a sequence readout process from  $F_4^c$  to the respective nodes in  $F_2^c$  and  $F_3^c$ . The restoration is conducted only when the current adopted plan succeeds or the current critic match function satisfies the vigilance.

### 3.5 Deliberation, Planning, and Learning

The basic execution cycle of the intentional agent architecture described above performs well if the available knowledge (plans) is complete without the chance that the agent make mistakes. The basic execution can not handle the situation appropriately when no plan can be found to solve the current goals. A different strategy can be applied to the execution cycle so that another plan that may contribute to the main goal achievement can still be selected although it does not perfectly match with the current goal. In fact, the standard ART model employs a new node recruitment which ensures that the network always ends up with a classification although no previous categories match the current input.

One approach of selecting plans without explicit directions from a higher level plan is by updating the vigilance parameter of the desires field to relax the resonance search process.

**Planning by relaxing deliberation:** When the resonance search to select a plan fails to find a winner node in  $F_2^c$ , a new category (plan) node is recruited automatically in  $F_2^c$ . If the readout from  $F_2^c$  to  $F_3^c$  produces no activation in  $F_3^c$  (a new plan is just allocated), the current activation patterns in  $F_2^c$  and  $F_3^c$  are stored in  $F_4^c$  in FILO mode. The vigilance  $\rho^{c2}$  of  $F_1^{c2}$  is then reduced by a certain value (e.g 0.1) while performing the resonance search iteratively until an existing node (plan) is found in  $F_2^c$ . In that case,  $\rho^{c2}$  is restored to its original value, and cycle continues as normal. If none still can be found, the activation status of all nodes in  $F_2^c$  are refreshed to allow previous reset node to be reselected.

The planning process above is not designed to produce the optimal selection. Instead, it can only be applied to search a plausible solution that leads the agent to achieving the goal. The reason of relaxing the vigilance parameter of *desires* field is to allow another plan that might have goals coincide with the current desires and is still currently

applicable (the plan precondition still strongly matches with current beliefs) to be selected. The strategy exploits the chance that subsequent applications of the same method can lead the agent to achieve the overall goals.

However, the result of the above planning method can only be used instantly in a single series of the agent performance. For a longer term, the sequence produced by the planning must be permanently captured and assigned to the respective plan. The challenge is that the planning above produces a sequence of plans instead of a sequence of atomic actions. Consequently, the plan that captures the sequence must transform the sequential application of plans into a series of subgoals. In iFALCON, this is handled by storing the sequence in the FILO field temporarily before it is learnt as weighted connections.

**Hierarchical plan learning:** When a new plan allocated for planning is currently stored in  $F_4^c$  and the current adopted plan just successfully achieves the current goal, the value of the  $F_1^{c4}$  (action) field is replaced momentarily to achieving action before the new plan in  $F_4^c$  is restored to  $F_2^c$ . After the upper level plan is restored from  $F_4^c$  to  $F_2^c$ , and the readout from  $F_2^c$  to  $F_3^c$  produces no activation in  $F_3^c$  (indicating the current restored plan is the new allocated one), a node in  $F_3$  is selected by a bottom up activation based on  $F_1^{c2}$  and  $F_1^{c4}$  as a new subgoal action following the FIFO mode of gradient encoding. The activation pattern in  $F_3^c$  is retained for a latter use by adjusting the weighted connections between  $F_3^c$  and  $F_4^c$  following the template learning rule. If any value (greater than zero) exists in  $F_3^c$  because of a readout from  $F_4^c$  to  $F_3^c$  (indicating a sequence has been temporarily retained), the order parameter  $\tau$  of  $F_3^c$  is set to the minimum activation value (but greater than zero) in  $F_3^c$ . The cycle of planning continues until the planning goal is achieved in which the weights between  $F_2^c$  to  $F_3^c$  can be adjusted to store the sequence permanently.

The planning and learning described above shows that the architecture integrates the process of deliberation, planning, plan execution, and plan learning without separations. It follows the principle of ART network that combines classification and learning with no separate phases. The planning depends on pre-existing plans comprising deliberation and execution while the learning depends on the planning to get the source experience. Another characteristic indicated from the processes above is that the FILO  $F_4^c$  field may occasionally retain the values permanently during the plan learning process although its main function is to store values temporarily.

In summary, the deliberation process of intentional agent is supported by the use of resonance search to select a category and the vigilance adjustment to bring out more options. The execution of the selected plan can be realized by the use of temporal encoding in certain fields following FIFO and FILO memory model. It is also possible to apply a certain commitment strategy by arranging the dynamics of ordering parameter and determining the conditions in which deliberation, vigilance adjustment, FILO readout, and learning are activated.

## 4. CASE STUDY AND EXPERIMENT

To test our model, we have implemented the model to solve the *block world* problem. In block world, the task is to

put some blocks into a certain goal configuration. Figure 4 shows an instance of a block world problem in which the target is to stack the blocks from one block configuration (the start) to a different block configuration (the goal). The block world is a domain involving sequential and hierarchical actions structure to solve the problem.

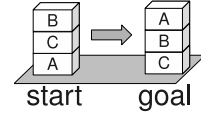


Figure 4: Block World.

To setup the model to solve the block world problem, the respective input/output fields are made to suit the domain problem. *Beliefs* and *desires* fields are expressed as vectors with *complement coding*. In complement coding, an input/output value is represented as a pair of complementing values which corresponds to a pair of node in an input/output field. Each node pair represent the truth value of a proposition. It is also possible to express *don't-care* condition using the complement coding. If the pair for a proposition  $p$  is expressed under complement coding as  $(v, \bar{v})^p$  for  $v$  refers to the truth value of  $p$ , then three different values can be expressed as follows:  $(1, 0)^p \equiv p$ ;  $(0, 1)^p \equiv \neg p$ ;  $(1, 1)^p \equiv$  (dont-care condition). Any proposition always matches the *don't-care* condition as the pair  $(1, 1)^p$  will always result in the same value when applied to the template matching function.

The block world domain is made to use the following propositions: *bupA*, *bbtmA*, *bupB*, *bbtmB*, *bupC*, *bbtmC* to express the situation. Each proposition denotes a condition on the top or the bottom of a block. For example *bupA* denotes that there is another block on top of block A, while *bbtmB* denotes some block under the block B. The negation of a proposition (expressed with a hyphen - symbol) states otherwise. For example *-bupA* and *-bbtmB* denote there is no block on top of A and B is lying on the ground respectively. The list of propositions is mapped to the vector representation as a list of value pair employing the complement coding. Each node in the action field  $F_1^{c4}$  is also associated with a primitive action. Nine external actions are defined as actions that directly change the block world environment state. Furthermore, an internal action is defined as an action for subgoaling. The action field does not employ complement coding and applies a competitive valuation (only a single node is active).

Two types of plans are applied: primitive plans which consists of only a single step of action such as follows,

```
{'goal': ['-bupB', '-bbtmA'],
 'pre': ['-bupA', 'bbtmA', 'bupB'],
 'body': [{'do': ['downA']}]}
'util': [1.0]}
```

and control plans that may include sequences and subgoals such as follows

```
{'goal': ['-bupA', 'bbtmA', 'bupB',
          'bbtmB', 'bupC', '-bbtmC'],
 'pre': [],
 'body': [{'achieve': ['-bupC', '-bbtmC']},
          {'achieve': ['bupC', 'bbtmB']},
          {'achieve': ['bupB', 'bbtmA']}]}
'util': [1.0]}
```

The primitive plans can be used as basic rules that model the environment, while the control plan is used as a strategy to achieve the goal. There are twelve different primitive plans and one control plan in the experiment.

To evaluate the model, two cases are formulated as follows: (1) A block world agent with complete learnt plans (both primitive and control plans); and (2) an agent with learnt incomplete plans (primitive plans only). Each configuration is tested to achieve the goals shown in Figure 4 from twelve different initial blocks configurations (Figure 5).

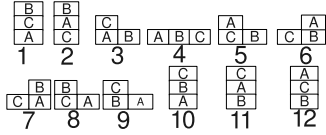


Figure 5: Tested initial block configurations.

Each iFALCON plans configuration is applied to reach the goal from every block configuration. Different configurations may produce different numbers of steps. A random choice mechanism is applied to the internal mechanism of decision so that when more than one node have the same maximum values, a plan node is selected at random. Consequently, the choices may be different even for the same configuration.

Initial block conf.	with prior primitives + control Plans					with prior primitives plans only				
	number of steps		improvement indicated		Failures	number of steps		improvement indicated		Failures
	min	max	improved	max diff.		min	max	improved	max diff.	
1	4	4				5	12			
2	5	5				9	9			
3	3	3								30
4	2	2								30
5	3	3								30
6	3	8	indicated	5	12					30
7	2	2								30
8	3	3				1	1			
9	3	3				3	11	indicated	4	28
10	3	3								30
11	4	11	indicated	7	8					30
12	4	4				5	31			10

Table 1: Results of Block World.

The test looks at the use plan execution, planning and, the autonomous plan learning capabilities. For each block configuration and two different types of initial plans, the architecture is tested to achieve the goal of block world. From consecutive trials (plans learnt from previous trials are retained) the experiment is conducted to test whether it learns plans that can be useful to subsequent runs within the same configuration.

Table 1 presents the results from 30 runs of 30 consecutive trials for each block configuration and each type of initial plans. The table shows the maximum and minimum number of steps from 30 consecutive trials that successfully achieve the goal. It also shows whether a performance improvement is indicated during consecutive trials. A configuration is marked as 'indicated' if there is an increase or convergence of performance (reduction of the number of steps to achieve the goal) in consecutive trials. If there is an indication of improvement, the table also shows the maximum

reduction or difference in the improvement. The number of achievement failures occur in 30 runs is also presented for each configuration.

The experiment reveals that when the control plan is included, most block configurations can be solved. There are cases in two configurations that the agent still fails to achieve the goal (twelve cases in configuration 6 and eight cases in configuration 11). The failures can still happen due to the initial plans that are actually unsuitable when applied to those particular configurations. A deeper analysis through the trace of execution reveals that the last step of the control plan still requires further planning which may add the chance of failure.

However, the two configurations reveal that performance improvements can be made by learning new plans. In configuration 6, there is a case in which the number of steps is reduced by five after learning is conducted in a previous trial. In configuration 11 the number of steps is reduced by seven. The improvement indication, however, does not imply that the learning does not take place in other configurations. In fact, most configurations that have the same number of steps to achieve the goal also have cases involving plan learning at the beginning trial. The learnt plans makes the agent follows the same plans accross different consecutive trials.

In one case, when the complete plans are applied as the initial configuration, the first trial can capture a new plan which can be mapped from the respective neural connections as follows:

```
{'goal': ['-bbtmC', '-bupC'],
 'pre': ['bbtmB', 'bbtmC', '-bbtmA', 'bupC', '-bupB', 'bupA'],
 'body': [{'achieve': ['-bbtmB', '-bupC']},
          {'achieve': ['-bbtmC', '-bupA']}],
 'util': [1.0]}
```

A closer look to the sample learnt plan reveals that the agent can learn plans consisting of a sequence of subgoals rather than atomic actions. Furthermore, the sample learnt plan above indicates that the plan is created to achieve a subgoal in the control plan. It implies that the learning takes place only when necessary to plan or solve certain tasks.

On the other hand, when only primitive plans are provided initially, the result is not as good as when the complete plans are provided. Table 1 reveals that only three (configuration 1, 2, and 8) out of twelve configurations successfully achieve the goal without failures. Moreover, there are seven configurations that totally cannot be solved. Only a few cases in one configuration (configuration 9) indicate performance improvement. Interestingly, there is a configuration in which the application of merely primitive plans becomes superior such as configuration 8. It takes only one step to solve the problem compared with the minimum three steps by its complete plans counterpart for the same configuration.

When only primitive plans are provided initially, a first trial captures a new plan which can be mapped into a symbolic description as follows:

```
{'goal': ['bbtmB', '-bbtmC', 'bbtmA', 'bupC', 'bupB', '-bupA'],
 'pre': ['bbtmB', 'bbtmC', '-bbtmA', 'bupC', '-bupB', 'bupA'],
 'body': [{'achieve': ['-bupA']}, {'achieve': ['bbtmA', 'bupB']},
          {'achieve': ['bupC']}, {'achieve': ['bbtmB', 'bupA']},
          {'achieve': ['-bupC']}, {'achieve': ['bbtmB', 'bupC']},
          {'achieve': ['bbtmA', 'bupB']}],
 'util': [1.0]}
```

The results indicate that the quality of planning and learning is sensitive to the pre-existing knowledge and the initial task

configuration. The right choice to start with may also lead the agent to further learn the appropriate knowledge to solve the task. The sample learnt plan starting with primitives only shows that the plan learnt may not be optimal or even include a series of mistakes.

Overall, the case study has confirmed that the neural network model can function as an intentional agent architecture driven by explicit plans. The results also reveal that the model can emulate and integrate the process of deliberation, planning, and learning beyond intentional or BDI model so that new solutions can be found even though the initial knowledge available is limited and incomplete. However, the learning process also relies on the prior knowledge available. Bad knowledge would even make learn worse.

## 5. CONCLUSION

This paper has presented a model of intentional agent realized as a self-organizing neural network architecture. The model explains how beliefs, desires, intentions, and plans can be mapped into a composition of multi-channel adaptive resonance theory networks. The model uses a new kind of temporal activation encoding to represent sequences and to handle hierarchical processing structure. The encoding technique allows sequences to be grouped and processed resembling the structure and operations of plans. The model emulates the processes involved in an intentional agent by seamlessly integrating deliberation, plan execution, and commitment as a single unit of activation cycles. Furthermore, based on the principle of adaptive resonance theory, planning and learning can be integrated as parts of the activation cycles.

The neural agent model has been implemented and tested to solve problems. Beyond intentional agents, the experiment confirms the capability of planning and learning to explore and capture new solutions. However, the test also reveals that the quality of planning and learning is sensitive to the availability of the appropriate prior knowledge. More in-depth studies are required to obtain the complete picture of the characteristics of the plan learning so that initial plans and network parameters can be setup more effectively.

The experiment conducted has indicated the potential of integrating a myriad of reasoning and learning mechanisms for agents accomplishing certain tasks. The neural agent model suggested comprises bi-directional pathways of activations which make it possible to select a plan or activating a sequential pattern from different directions so that the plan can also be selected based on the presentation of action sequences rather than triggered by goals. Although the model and the experiment are still designed for a single agent domain, it is possible to extend the model to deal with more complex issues involving multiple agents like plan recognition or learning by imitation.

In any case, the neural model for intentional agent architecture can bridge two different approaches of building agents. Top-down formal symbolic approaches can be integrated with bottom-up non-symbolic processes to accomplish a single task domain. Both directions can support and enrich each other to realize an agent architecture that ultimately deliberate, plan, and learn.

## 6. REFERENCES

- [1] G. Bradski, G. A. Carpenter, and S. Grossberg. STORE working memory networks for storage and

- recall of arbitrary temporal sequences. In *Biological Cybernetics*, volume 71, pages 469–480. 1994.
- [2] M. E. Bratman. What is intention? In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, chapter 2, pages 15–31. MIT Press, Cambridge, 1990.
- [3] G. A. Carpenter and S. Grossberg. Adaptive Resonance Theory. In M. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 87–90. Cambridge, MA: MIT Press, 2003.
- [4] G. A. Carpenter, S. Grossberg, and D. B. Rosen. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759–771, 1991.
- [5] D. C. Dennet. *The Intentional Stance*. MIT Press, Cambridge, 1987.
- [6] M. Ghallab, D. S. Nau, and P. Traverso. *Automated Planning: theory and practice*. Elsevier/Morgan Kaufman, Amsterdam, 2004.
- [7] S. Grossberg. Behavioral contrast in short-term memory: Serial binary memory models or parallel continuous memory models? *Journal of Mathematical Psychology*, 3:199–219, 1978.
- [8] F. Ingrand, M. Georgeff, and A. S. Rao. An architecture for real-time reasoning and system control. 7(6):34–44, 1992.
- [9] D. Kinny and M. Georgeff. Commitment and effectiveness of situated agents. In *Proceedings of the 12th International Joint Conference of Artificial Intelligence 1991*, pages 82–88, Sidney, 1991.
- [10] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, 2006.
- [11] M. E. Pollack. Plans as complex mental attitudes. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, chapter 5, pages 77–103. MIT Press, Cambridge, 1990.
- [12] M. E. Pollack. The uses of plans. *Artificial Intelligence*, 57(1):43–68, 1992.
- [13] A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95)*. San Francisco, 1995.
- [14] M. Schut, M. Wooldridge, and S. Parsons. The theory and practice of intention reconsideration. In *Journal of Experimental and Theoretical Artificial Intelligence*, volume 16, pages 261–293. 2004.
- [15] H. A. Simon. *Models of Bounded Rationality*. MIT Press, Cambridge, 1982.
- [16] A.-H. Tan. FALCON: A Fusion Architecture for Learning, COgnition, and Navigation. In *proceedings, International Joint Conference on Neural Networks (IJCNN'04)*, pp. 3297–3302, 2004.
- [17] A.-H. Tan, G. Carpenter, and S. Grossberg. Intelligence Through Interaction: Towards A Unified Theory for Learning. In *proceedings, ISNN'07, LNCS 4491*, Part I, pp. 1098–1107, 2007.
- [18] 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. 9, No. 2 (February 2008), 230–244.