

Planning with iFALCON: Towards A Neural-Network-Based BDI Agent Architecture

Budhitama Subagdja
Intelligent Systems Centre
Nanyang Technological University
budhitama@ntu.edu.sg

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

Abstract

This paper presents iFALCON, a model of BDI (belief-desire-intention) agents that is fully realized as a self-organizing neural network architecture. Based on multi-channel network model called fusion ART, iFALCON is developed to bridge the gap between a self-organizing neural network that autonomously adapts its knowledge and the BDI agent model that follows explicit descriptions. Novel techniques called gradient encoding are introduced for representing sequences and hierarchical structures to realize plans and the intention structure. This paper shows that a simplified plan representation can be encoded as weighted connections in the neural network through a process of supervised learning. A case study using the blocks world domain shows that an iFALCON agent can also do planning to solve problems when the knowledge is incomplete.

1 Introduction

Intelligent agents designed to perform in dynamic environments are required to adapt their behavior in order to fulfill their objectives. The adaptation may involve decisions on what and how to achieve the objectives. An intentional agent architecture such as beliefs-desires-intentions (BDI) model [5] exploits prescribed procedural knowledge and explicit representation of mental attitudes to make the agent achieves its goal. The agent can be designed to follow *recipes* as guidances while at a certain moment choosing an alternative mean to achieve the goal.

The performance of a BDI agent relies on how good the agent designer specifies the domain specific knowledge and mental attitudes of the agent. However, the capability to invent new procedures or recipes is still not supported by the BDI architecture [4]. Learning or planning from the first principle may reduce the responsiveness to changes in the environment as additional reasoning processes would sig-

nificantly take processing time. Although it is possible that the agent constructs or acquires its own recipes from experience autonomously, this approach is still under the principle of explicit prescribed knowledge [12]. Thus, to make the agent learns requires a careful consideration on how the agent may interact with the environment and fulfill its tasks [9].

On the other hand, neural network architectures, as biologically-inspired models of computation, have been widely applied to domains that require learning and generalization. As opposed to symbolic processing approaches that rely on explicit descriptions, neural networks operate by matching inputs with distributed learnt patterns encoded in weight values. The performance of a neural network model depends on a prior learning phase. 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.

There have been great interests in consolidating intentional agent models and self-organized learning systems. Some approaches use hybrid neural network models [13, 8]. Others still maintain symbolic framework enhanced with particular learning algorithms [10, 6, 12]. Most of the hybrid systems assume *bottom-up* learning processes. Their primary concern is in extracting procedural knowledge from reactive performances of the agent. However, there are still open issues in how to reuse the extracted knowledge effectively. Most of the extracted procedures using the hybrid methods consists of only plain sequences of actions. Moreover, the influence of prior knowledge in such hybrid systems is mostly unconsidered. The motivation behind this paper is towards discovering plausible agent models that are inherently adaptive with self-organized learning features but also accommodating explicit plans description and mental attitudes. We are exploring towards a natural way of the consolidation supporting hierarchical structures beyond learning plain sequences.

In this paper, we propose an architecture of intentional

agent using a neural network model. The architecture is developed towards bridging the gap between a self-organizing neural network that autonomously adapts its knowledge and the BDI agent model that follows explicit descriptions. Unlike other approaches that suggest hybrid structures, our architecture is entirely built as a new kind of neural network model called iFALCON.

iFALCON is based on a multi-modal neural network model called fusion ART [16]. Fusion ART employs multiple neural fields that can process different input and output modalities. iFALCON extends the fusion ART model by representing sequences and hierarchical structures. This paper shows that iFALCON has features as follows: *Plan encoding*: a representation of plans as *recipes* can be encoded as weighted connections through a process of supervised learning; *Plan selection*: a plan for achieving the agent's goal can be selected as a neural node activation to be scheduled for execution; *Plan execution*: the selected plan can be executed by activating connected nodes exhibiting sequences of actions and the intentions hierarchy; *Plan search*: planning to find a novel solution can still be performed even though the encoded plans are incomplete.

This paper is organized as follows: Section 2 provides an overview of the BDI agent architecture. Section 3 presents the proposal of the iFALCON as a new architecture that extends the fusion ART network to fulfill the requirement of the BDI agent model. Section 4 describes how this iFALCON architecture works using a situated blocks world planning domain. The case study shows how plans can be mapped into neural network connections, and how they are executed to achieve goals. Moreover, it demonstrates a planning mechanism for finding a new solution when incomplete information is available. Section 5 summarizes and concludes this paper.

2 BDI: Intentional Agent Architecture

The BDI (Beliefs, Desires, Intentions) agent model [11] is a design framework commonly used in developing agents that behave both deliberately and reactively in a complex changing environment. The main principle is to use explicit representations of the agents' own mental attitudes (in terms of attributes such as *beliefs*, *desires*, and *intentions*) to direct their actions and the selection of appropriate predefined plans as *recipes*. A BDI agent works as an interpreter that operates on the representation 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 which can be updated directly by events captured by sensors or by some changes in the agent's internal state. In most implemented platforms of BDI agent, the *beliefs* may consist of predicate logic statements or propositions indicating truth values (true or false).

The agent has *desires* as a set of goal conditions that the agent wants to achieve. From *desires*, the agent needs to select some goals that are consistent with its *beliefs*. After the goals are determined, the agent then finds a mean to achieve them as a plan. In situations where computational resources are limited, only one goal and a plan is required to be selected. In Procedural Reasoning System (PRS) [7], as might be the first and the most adopted implementation model of BDI agent, the agent does not plan from scratch to find a way to achieve its goal. Instead, a library of manually prescribed plans is already provided. A plan is a procedural knowledge structure that specifies how to achieve certain goals. The plan typically has **goal** attribute as the postcondition that will hold after the plan is executed; **precondition** attribute that specifies the condition that must be held so that the plan is applicable, and; **body** attribute that describes the course of actions that will be executed if the plan is adopted.

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

The basic control operation of a BDI agent is a cycle which consists of the following steps: Updates beliefs; determines options at the given circumstances; deliberates and select intentions to be executed; executes intentions.

In the execution cycle, the agent may achieve subgoals that comprise further deliberation at subordinate levels. The deepest level intention is achieved first before continuing with the upper-level ones.

3 iFALCON: The intentional FALCON

Based on Fusion Adaptive Resonance Theory (fusion ART) [16] network model, we are looking at a network model that can process different mental attitudes and encode plans. fusion ART is a neural architecture that unifies a number of neural network designs, most notably ART [3, 2], Adaptive Resonance Associative Map (ARAM) [14] and Fusion Architecture for Learning, COgnition, and Navigation (FALCON) [15]. The ART neural network and its derivatives use the principles based on an analysis of human and animal cognitive information processing. The original ART model works as a self-organizing neural network that employs unsupervised learning. It comprises the allocation of a new category neuron which allows the network to learn new categories and grow.

Fusion ART extends the original ART model by employing multiple input fields. Each input field employs independent parameters. Different input fields may also have different types of encoding vectors. That enables fusion ART network to process different input modalities.

The use of multiple independent fields enables different stages of operation so that various types of computation can be employed. The multi-fields architecture also extends the unsupervised learning model so that different learning schemes like supervised and reinforcement learning can be applied as well. For example, FALCON employs reinforcement learning by configuring the input fields to act as input *state*, output *action*, and *reward* feedback so that it learns actions policy when it receives a positive feedback.

iFALCON extends the fusion ART model, by arranging and grouping the channels or fields in a different way. As shown in Figure 1 the iFALCON architecture consists of four input/output fields: F_1^{c1} , F_1^{c2} , F_1^{c3} , and F_1^{c4} denoting the *beliefs*, *desires*, *critic*, and *action* fields respectively. The *beliefs* and *desires* fields correspond to the *beliefs* and *desires* components of the BDI model. The *beliefs* and the *desires* field contain the conditions that the agent believes to hold and the conditions that the agent wants to achieve respectively. On the other hand, 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.

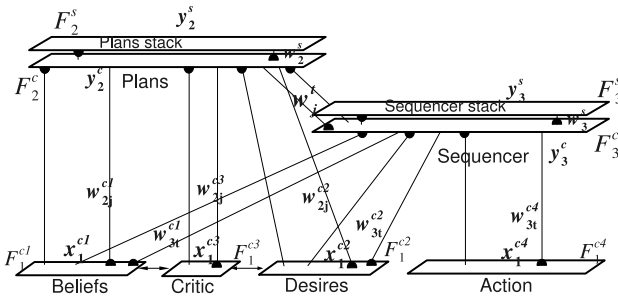


Figure 1. iFALCON Architecture

Whereas the *beliefs* and *desires* fields are connected to both category fields, the *critic* field is only connected to the plans field and the *action* is connected only to the *sequencer*. There are extra connections between the *plan* and the *sequencer* category fields and each category field is further connected to a stack field.

3.1 Plan Encoding

A plan can be encoded by setting up the appropriate weight values of the neural network connections. Nodes in *plan* 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 iFALCON architecture employs a novel technique called *gradient encoding* for representing sequences and stacks. In *gradient encoding*, a value represents an index or a position of a point in a one-dimensional value system. A value may simply indicate a time point or a position in an ordered sequence. This kind of encoding is supported by a model of working memory in the brain for storing temporal sequences [1].

In iFALCON, the gradient encoding is used for representing plans and goal-subgoal hierarchies. The use of the two *plan* and *sequencer* category fields allows hierarchical groupings of actions and a sequence into a single category node in the *plan* field. Each category node in the *sequencer* field encodes a pattern of action in the input/output fields. Through the extra connections between the *plan* and the *sequencer*, a node in the *plan* field can also encode a collection of actions represented as different activations in the *sequencer* field. By applying the gradient encoding to the connections' values, different actions can be arranged in an ordered fashion.

The corresponding nodes in both *plan* and *sequencer* field can be manually defined to represent a plan and the respective weights are adjusted accordingly. However, this manual encoding may redundantly connect different nodes in the *sequencer* field to the same pattern of activation. Alternatively, plans are learnt so that the neural network can automatically recruit the appropriate nodes in the *plan* and the *sequencer* field through a code search activation process. This process can also be said as *supervised learning*, in which the learning requires both the input (*desires* and *beliefs*) and the output (actions) samples to be presented. The respective weights are then adjusted accordingly. In this way, the same node in the *sequencer* field may be connected by different nodes in the *plan* field sharing the same code of action.

The code search activation process comprises a search process to achieve a *resonance* condition between the plan field and the input/output fields. A *resonance* is a condition in which a top-down activation from a node in the category field match with the current activations of the connected fields at the bottom. A search cycle comprises a competitive selection of the highest activation node in the category field followed by a top-down matching. The selected node is reset if it does not satisfy the resonance based on a vigilance criteria. The cycle continues with another node until a resonance is found. If no node can meet the resonance,

a new category node is assigned to encode the pattern of the respective input/output fields. The detailed algorithm is presented below.

Let \mathbf{I}^{ck} and \mathbf{x}^{ck} denote the input vector and the activity vector for F_1^{ck} , respectively. Let \mathbf{y}_2^c and \mathbf{y}_3^c denote the activity vectors in F_2^c and F_3^c respectively. The j th node of F_2^c is associated with the activity vector of F_1^{cl} , by a weight vector \mathbf{w}_{2j}^{cl} , for $l = 1, \dots, 3$. The n th node of F_3^c is associated with F_1^{ci} , by a weight vector \mathbf{w}_{3n}^{ci} , for $n = 1, 2$, and 4.

F_2^s and F_3^s are stack fields connected to F_2^c and F_3^c respectively. Let \mathbf{w}_2^s denote the weight vector associated with the s th node in F_2^s and nodes in F_2^c , and \mathbf{w}_3^s for F_3^s and nodes in F_3^c . The iFALCON dynamics is determined by choice parameters $\alpha^{ck} \geq 0$, learning rate parameters $\beta^{ck} \in [0, 1]$, contribution parameters $\gamma^{ck} \in [0, 1]$ and vigilance parameters $\rho^{ck} \in [0, 1]$.

Specifically, for each F_2^c node j , the choice function T_j^c is computed as follows:

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

where the fuzzy AND operation \wedge is defined by $(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i)$, and the norm $|\cdot|$ is defined by $|\mathbf{p}| \equiv \sum_i p_i$ for vectors \mathbf{p} and \mathbf{q} . A plan competition follows in F_2^c with J as the index of the winner $T_J^c = \max\{T_j^c : \text{for all } F_2^c \text{ node } j\}$. A category choice is made at node J , $y_j^c = 1$; and $y_j^c = 0$ for all $j \neq J$. Resonance occurs if for each channel l :

$$m_J^{cl} = \frac{|\mathbf{x}^{cl} \wedge \mathbf{w}_J^{cl}|}{|\mathbf{x}^{cl}|} \geq \rho^{cl}. \quad (2)$$

If any of the vigilance constraints is violated, mismatch reset occurs in which the value of T_J is set to 0. Another F_2^c node J is then selected and the process repeats until a resonance is found or otherwise, a new *uncommitted* node is recruited. When the search ends, for each channel cl , the weight vector \mathbf{w}_J^{cl} is modified by the following learning rule $\mathbf{w}_J^{cl(new)} = (1 - \beta^{cl})\mathbf{w}_J^{cl(old)} + \beta^{cl}(\mathbf{x}^{cl} \wedge \mathbf{w}_J^{cl(old)})$.

F_3^c can employ the same node search operation as in F_2^c . Following the *gradient encoding*, the values of \mathbf{w}_j^t must be arranged so that if t_0, t_1, \dots, t_n denote time points or a sequence in an incremental order, the weight values follow the condition $w_j^{t_0} > w_j^{t_1} > \dots > w_j^{t_n}$. The weights \mathbf{w}_j^t can be adjusted as follows $\mathbf{w}_j^{t(new)} = (\nu^t \mathbf{y}_3^c) \vee \mathbf{w}_j^{t(old)}$ where the fuzzy OR operation \vee is defined by $(\mathbf{p} \vee \mathbf{q})_i \equiv \max(p_i, q_i)$. The ordering parameter ν^t set the index of the action in the sequence. Initially, ν^t is set to one. As a new input action is presented in the *action* field, the previous selected node in F_3^c is reset and $\nu^{t(new)} = \nu^{t(old)} - v$ to make \mathbf{w}_j^t follows the *gradient encoding*. v is a constant as an increment/decrement factor in gradient encoding.

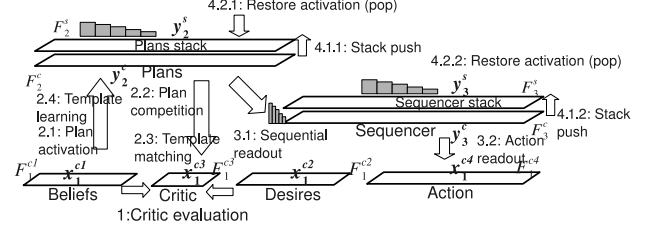


Figure 2. iFALCON execution cycle.

3.2 Plan Selection and Execution

The activities in the composition of neural fields and connections in iFALCON emulates the execution cycle in the BDI agent model. The execution cycle of iFALCON 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 . When the match function is greater than the parameter ρ , it means that the the goals are satisfied. m^c is also used as the value in the critic vector $x_1^{c3} = m^c$. The deliberation cycle continues to the next stage only when $m^c < \rho^{c3}$.

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* 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 *readout* processes (top-down activations) from the selected node in the *plan* field to the *sequencer* field and further down to the respective input/output fields. After activating *sequencer's* nodes a competition is employed to select a node K with the highest activation value so that further readout activation can be applied to the input/output fields. The node K in the *sequencer* is then reset, and the process continues by selecting another node following the order in the sequence until the activation reaches zero. The ordering parameter ν^t keeps track the last readout value in F_3^c so that all $T_k^t > \nu^t$ are reset. By repeating the readout process, sequential activations can be produced in F_3^c . The readout process ends when no activation occurs in F_3^c (all nodes are 0) or the critic passes the vigilance criterion.

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 the connection values between a node in the *sequencer* field and the nodes in the *desires* field, overrides the *desires* which trigger a further plan selection process. However, before replacing the desires, the current active nodes in the plan and the sequencer field are stored into the stack fields.

A stack field is basically a category field but activated following the gradient encoding scheme. By applying the gradient encoding to the activation level of the nodes in the stack field, each level of activation represents the position or the depth level in which the encoded activation pattern in the connected category field were temporarily stored. To push the active node in F_2^c to the stack, a node J in the F_2^s is selected through the process of resonance search excluding the previous active nodes. If the activity vector of F_2^s is \mathbf{T}^s , the activation values in F_2^s are then accumulated based on the leveling parameter $\mathbf{y}_2^{s(new)} = (\tau^{s_2} \mathbf{T}^s) \vee \mathbf{y}_2^{s(old)}$.

The gradient encoding may appear in \mathbf{y}_2^s as more values are pushed. the leveling parameter τ^{s_2} is incremented by v whenever a value is stored into the stack. The same process is also applied to F_3^s to store the position in the sequence. After storing the value to F_2^s , the subgoal can override the values in F_1^2 and trigger another deliberation process.

To retrieve the previously stored plan, the winner node J is determined in F_2^s and a readout is performed to F_2^c through the weight vector \mathbf{w}_2^J and the plan is restored by performing readout further down to F_1^{ck} . The winner node J in F_2^s is then reset to make the latter stored plan will be selected in the next cycle. The leveling parameter τ^{s_2} is also updated by the highest activation value in the stack field to keep track the top most position of the stack. The same operation is performed between F_3^s and F_3^c , but the parameter ν^t is also restored by $\nu^t = T_K^t$, K is the node index in the restored vector \mathbf{y}_3^c .

3.3 Plan Search

It is also possible to control the deliberation and execution of iFALCON by adjusting the parameters dynamically to improve the quality of the decisions. Normally, a high vigilance level can be applied to the *desires*, the *beliefs*, and the *critic* fields so that the plan selection and evaluation is strict. This strategy may be suitable by the assumption that the knowledge (plans) available is complete and there is no chance for the agent to make mistakes. However when no plan can be found, the process will just halt. A strategy can be applied to the iFALCON execution cycle that makes adjustments to the vigilance parameters at the execution time. This enables the network to select a plan even though no plan can be found using the original vigilance. The strategy can be summarized as follows: 1) If no plan (node) can be found, store the current plan to the stack and reduce the *desires*' vigilance ρ^{c2} by a certain value (e.g 0.1); 2) keep reducing ρ^{c2} until a plan can be found; 3) if $\rho^{c2} = 0$ refresh the node activation status; 4) Restore the original ρ^{c2} .

4 Case Study: Blocks World

To test our model, we have implemented iFALCON to solve the *blocks world* problem. In blocks world, the task of the agent is to put some blocks into a certain goal configuration. Figure 3 shows an instance of a blocks world problem in which the target is to stack the blocks from one blocks configuration (the start) to a different blocks configuration (the goal). Although the blocks world domain is simple, it requires sequential and hierarchical actions structure to solve the problem.

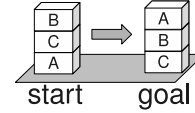


Figure 3. Blocks World.

To start with, the respective fields of iFALCON are setup to represent the situation of the domain. *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.

In our blocks world domain, the following propositions: $bupA$, $bbtmA$, $bupB$, $bbtmB$, $bupC$, $bbtmC$ can be used 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 $\neg bupA$ and $\neg bbtmB$ denote there is no block on top of A and B is lying on the ground respectively. The list of propositions can be 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 blocks 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).

In order to store a plan to iFALCON, the sequential learning is conducted by activating the respective fields. A

plan can be described as a collection of attributes corresponding to weight values that are connected to input fields. For example the following plan describes attributes that will be encoded as weight values:

```
{'plan':
  {'goal': ['-bupB', '-bbtmA'],
   'perc': ['-bupA', 'bbtmA', 'bupB'],
   'body': [{'action': ['downA']}]}
}
```

The *goal* and *perc* refer to weight values connected to the *desires* and the *beliefs* field respectively. The *body* refer to values of connections between F_2^c , F_3^c , and the action field F_1^4 to describe the sequence of actions. By activating F_2^c fields (with the critic set to one by default), and F_3^c through the bottom-up activations, the weight values between F_2^c and F_3^c can be adjusted accordingly.

The continuous line refers to (1, 0) weight pair, and the dashed line refers to (0, 1) pair. The *don't-care* values and non-complement zeros in weight connections are not shown. A more complex plan can be described to comprise a sequence of action in the *body* attribute. The following plan contains a plan body with a sequence of subgoals:

```
{'plan':
  {'goal': ['-bupA', 'bbtmA', 'bupB',
           'bbtmB', 'bupC', '-bbtmC'],
   'perc': [],
   'body': [{'action': ['subgoal'],
                  'goal': ['-bupC', '-bbtmC']},
            {'action': ['subgoal'],
                  'goal': ['bupC', 'bbtmB']},
            {'action': ['subgoal'],
                  'goal': ['bupB', 'bbtmA']}]
}
```

To learn this plan, a node in F_2^c is firstly selected based on the input *desires* and *beliefs*. While the node in F_2^c is activated, the actions are presented sequentially in the respective order to activate different F_3^c nodes. The corresponding weights between F_2^c and F_3^c are then adjusted following the gradient encoding scheme.

Two types of plans are applied: primitive plans which consists of only a single step of action, and control plans that may include sequences. 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. In our implementation, we mapped twelve different primitive plans and one control plan into the network.

To evaluate the model, two cases are formulated as follows: (1) A blocks 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 3 from twelve different initial blocks configurations shown in Figure 4.

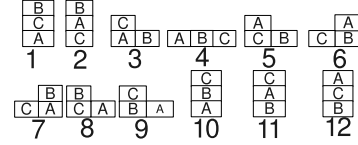


Figure 4. Tested initial blocks configurations.

Each iFALCON plans configuration is applied to reach the goal from every blocks configuration. Different configurations produce different numbers of steps. A random choice mechanism is applied to the internal mechanism of iFALCON 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. Figure 5 presents the results from 100 trials for each configuration.

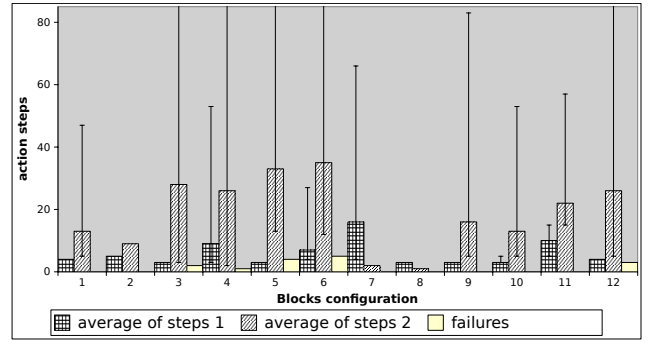


Figure 5. Results of Blocks World.

The results show that when more plans are available (plans configuration 1: primitive plans and a control plan), all blocks configuration can be solved for every trial. Few configurations require many steps (configuration 7 and 4) with some degree of variability, but most of the time, the steps taken are much lesser than the maximum. This indicates that in some cases, even with control plans, the agent can not rely solely on its plans and must make different choices. It is also indicated that the deliberation strategy and backtracking process can lead the choices to the solution although the steps taken can be many. Using the complete set of plans, the goal can be achieved in three steps at minimum.

On the other hand, when the plans learnt are incomplete (with primitive plans only), some failures may occur (configuration 3, 5, 6, and 7). However, the figure shows that, most of the time, the network eventually reaches the solution even though it takes so many steps comparing with the results using the complete set of plans. From 100 trials, maximally only five cases failed (configuration 6). In fact, in a case where the solution is straightforward (configura-

tion 8), it takes only a single step of action rather than three steps like in its complete set of plans counterpart. This indicates that, more knowledge (plans) does not always lead to better performance.

Overall, the case study has confirmed that the iFALCON can function as a BDI agent architecture driven by explicit plans. The results also reveal that iFALCON can emulate the process of reasoning and planning beyond BDI model so that new solutions can be found even though the initial knowledge available is limited and incomplete.

5 Conclusion

This paper has presented a model of BDI agent realized as a neural network architecture. The model explains how beliefs, desires, intentions, and plans can be mapped into artificial neural substrates. The novelty of this model is in the application of the gradient encoding technique for representing stacks and sequence structure. The combination of gradient encoding and the activation mechanism of fusion ART allows sequences to be grouped or encapsulated under a single neural code which is an important feature to realize a BDI agent. The model also enables the storage and reproduction of different neural activations in a sequential manner. The iFALCON model has been successfully implemented and tested to store and execute plans. Furthermore, beyond the BDI model, the test revealed the capability of planning to explore and find new solutions.

However, there are still some important issues left untouched by this paper. Parallel to the symbolic-based BDI agent counterpart, the iFALCON neural network still has some open issues in realizing the appropriate deliberation strategies. In this paper, we start to address this problem by looking at the relationship of the network generalization feature and the *means-ends* reasoning capability. The test conducted has indicated the potential of adjusting vigilance criterion of different neural fields to realize the reasoning mechanism for BDI agents. The next stage in developing iFALCON may also involve the study of autonomous learning complementary to the current supervised learning. In any case, the iFALCON architecture opens up the possibility of enhancing the BDI agent model beyond the symbolic frameworks. The architecture may also set a new direction in exploring the potential of neural network, particularly ART-based, to realize the model of intentional agents that is more biologically plausible.

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] G. A. Carpenter and S. Grossberg. Adaptive Resonance Theory. *The Handbook of Brain Theory and Neural Networks*, pages 87–90. Cambridge, MA: MIT Press, 2003.
- [3] G. A. Carpenter and S. Grossberg, editors. *Pattern Recognition by Self-Organizing Neural Networks*. Cambridge, MA: MIT Press, 1991.
- [4] M. P. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge. The belief-desire-intention model of agency. *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555 of *Lecture Notes in Artificial Intelligence*, pages 1–10, Heidelberg, 1999. Springer-Verlag.
- [5] A. Haddadi and K. Sundermeyer. Belief-desire-intention agent architectures. *Foundations of Distributed Artificial Intelligence*, pages 169–185. John Wiley & Sons, New York, 1996.
- [6] A. G. Hernandez, A. E. Segrouchini, and H. Soldano. BDI multiagent learning based on first-order induction of logical decision trees. *Intelligent Agent Technology: Research and Development*. World Scientific, New Jersey, 2001.
- [7] F. Ingrand, M. Georgeff, and A. Rao. An architecture for real-time reasoning and system control. *IEEE Expert: Intelligent Systems and Their Applications*. 7(6):34–44, 1992.
- [8] S. Karim, L. Sonenberg, and A.-h. Tan. A Hybrid Architecture Combining Reactive Plan Execution and Reactive Learning. *Proceedings of 9th biennial pacific rim international conference on artificial intelligence (pricai-06)*, volume 4099 of *Lecture Notes in Artificial Intelligence*, pages 200–211, Heidelberg, 2006. Springer-Verlag.
- [9] S. Karim, B. Subagdja, and L. Sonenberg. Plans as products of learning. *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2006 Main Conference Proceedings) (IAT'06)*, pages 139–145, Washington, 2006. IEEE Computer Society.
- [10] C. Olivia, C.-F. Chang, C. F. Enguix, and A. K. Ghose. Case-based BDI agents: an effective approach for intelligent search on the world wide web. *Proceedings of the AAAI-99, Spring Symposium on Intelligent Agents in Cyberspace*. 1999.
- [11] A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95)*. San Francisco, 1995.
- [12] B. Subagdja. *Intentional Learning in Bounded-Rational Agents*. PhD thesis, Department of Information Systems, University of Melbourne, 2007.
- [13] R. Sun. *Duality of the Mind: A Bottom-Up Approach Toward Cognition*. Lawrence Erlbaum, Mahwah, 2002.
- [14] A.-H. Tan. Adaptive Resonance Associative Map. *Neural Networks*, 8(3):437–446, 1995.
- [15] A.-H. Tan. FALCON: A fusion architecture for learning, cognition, and navigation. *Proceedings, International Joint Conference on Neural Networks*, pages 3297–3302, 2004.
- [16] A.-H. Tan, G. Carpenter, and S. Grossberg. Intelligence Through Interaction: Towards A Unified Theory for Learning. *International Symposium on Neural Networks (ISNN) 2007*, volume 4491, pages 1098–1107, Nanjing, China, June 2007. LNCS.