11-2011

A brain-inspired model of hierarchical planner

Budhitama SUBAGDJA

Ah-Hwee TAN

A Brain-Inspired Model of Hierarchical Planner

Budhitama Subagdja
School of Computer Engineering
Nanyang Technological University, Singapore
Email: budhitama@ntu.edu.sg

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

***Abstract*—Hierarchical planning is an approach of planning by composing and executing hierarchically arranged plans to solve some problems. Most symbolic-based hierarchical planners have been devised to allow the knowledge to be described expressively. However, a great challenge is to automatically seek and acquire new plans on the fly. This paper presents a novel neural-based model of hierarchical planning that can seek and acquired new plans on-line if the necessary knowledge are lacking. Inspired by findings in neuropsychology, plans can be inherently learnt, retrieved, and manipulated simultaneously rather than discretely processed like in most symbolic approaches. Using a multi-channel adaptive resonance theory (fusion ART) neural network as the basic building block, the so called iFALCON architecture can capture and manipulate sequential and hierarchical relations of plans on the fly. Case studies using a blocks world domain and unreal tournament video game demonstrate that the model can be used to execute, plan, and discover plans and procedural knowledge through experiences.**

***Keywords*-hierarchical planning; plan learning; adaptive resonance theory**

I. Introduction

Various computational models of planning have been devised mostly based on symbolic approaches [1]. In practice, a popular approach of planning is the hierarchical transition model [2] (hierarchical planning) in which hierarchically arranged plans are assumed to be available as pre-given knowledge. This hierarchical approach can take less computational efforts as it does not have to consider all alternative solutions but to focus instead on choosing the right goal (deliberation) and the right plan (means-end reasoning) at the right moment as parts of the execution. An important challenge is how to make this kind of planner discovers and acquires new plans autonomously without relying on much efforts of the system developer or domain experts. Consequently, this includes the ability to acquire the hierarchical arrangements of the plan on the fly.

On the other hand, recent findings in neuropsychology have indicated that these goal oriented and planning-related abilities have also been observed in human and animals and identified to be related to a particular area in the brain [3]. Evidences based on brain lesions and neuroimaging have revealed that the frontal cortex area in the brain is particularly related to higher-level cognitive functions and controls such as integrating multiple sources of input, sustaining intentions, acquiring rules, monitoring the progress of plans, and goals-subgoals decomposition [4]. In other words, the frontal cortex also serves particularly as a kind of flexible and adaptive hierarchical planner in the brain.

This paper presents a neural architecture for hierarchical planning agents featuring on the fly plan discovery and learning. The proposed architecture is made as interconnected modules each corresponds to the functionality of a certain frontal cortex area in the brain. Using neural networks and brain structures as the basis, the planner model can simultaneously process and learn plan representation continuously rather than discretely and serially like in most symbolic-based approaches. Each functional part or module is made as a multi-channel neural network building block called fusion ART [5] which enables continual cycles of categorizing, matching, learning, and growing by allocating new neurons. The proposed architecture, called iFALCON, can map symbolic descriptions into weighted neural connections and turns interacting pattern matching in neural networks into state-space search and manipulation supporting planning processes.

The work in this paper is generally intended to explore an alternative to learning and adaptation mechanisms in hierarchical planning without suggesting a new ultimate planner that can produce optimal solutions. However, our main contributions to the current state-of-the-art include a novel way to represent sequences and transient hierarchical structure in neural networks so that plans can be retrieved, executed, and traced back (backtracking). We also come up with a solution of how planning, learning, and execution can seamlessly interleave each other in dynamic continuous cycles of exploiting and forming hierarchical plans structures.

The rest of the paper is organized as follows. Section II discusses related works in hierarchical planning systems. Section III describes the proposed neural network model of hierarchical planner. Section IV presents and discusses our implementation of iFALCON to illustrate how it can follow plans as prior knowledge while invent and learn new ones when the knowledge is insufficient. Section V summarizes and concludes the work with some discussion on further use and development of the architecture.

II. Related Works

Some symbolic models of hierarchical planning have included learning to acquire new plans when pregiven ones are incomplete. Some approaches have considered *learning-by-doing* in which the definition or parameters of operators are acquired by online observation and practices [6], [7], [8]. This kind of self-directed learning enables the planner to be flexibly applied to complex and dynamic domains like in robotics or reactive control systems. However, the representations of the knowledge acquired are mostly simple and limited only for adjusting predefined parameters. Some HTN (Hierarchical Transition Network) planners have overcame these limitations by employing particular mechanisms to update pre-existing plans or create a new more efficient set of plans from prior knowledge [9], [10], [11]. The HTN learning models, however, are still considered to be offline as they learn prior to the application of the acquired knowledge, although the approach may relieve domain experts from crafting the complete and correct knowledge.

On the other hand, only few neural-inspired models or brain-like structures of planner have been proposed. Most of them focus only on certain limited aspects of planning. Some neural models are developed as controllers for agents or reactive systems [12], [13]. They are not meant for acquiring complete hierarchical structure of plans nor flexibly reuse them but only sustaining limited parameters or brief sequential episodes. Some models have already considered both the biological plausibility and the representational adequacy of reactive planning systems [14]. The system has also been related to the state-space search method by demonstrating a simple backward chaining process to search for a plan over the neural network structure [15]. However, the issue of capturing hierarchical plans on the fly is almost totally untouched.

The model proposed in this paper is intended towards realizing a neural-plausible architecture for hierarchical planning that includes most aspects of planning covering reactive execution control, means-ends reasoning, and capturing plans on-line. Unlike other hierarchical planners, the proposed neural network inherently categorizes and learns plan descriptions as neural activation patterns.

III. The Neural Architecture of Hierarchical Planner

Neuropsychological studies have revealed that the brain frontal cortex area can be associated with higher cognitive functions and cognitive control [4]. Among all the observed functional features, frontal cortex can be related to basic operations supporting hierarchical planning as follows: (1) selecting and initiating goals and task behaviour; (2) sustaining information for performing the tasks and goals achievement (working memory); (3) breaking down goals into subgoals; (4) monitoring and evaluating goals achievement; and (5) hypothesis generation and adapting rules (plans) of behaviour. Our proposed model is made to realize those functional traits of the corresponding brain cortical area in neural networks. Before moving on to the corresponding model, the next part of this section describes the basic building block used to realize each part or module of the whole architecture.

A. Fusion ART: The Building Block

The proposed hierarchical planner model is based on fusion ART [5], a derivation of Adaptive Resonance Theory (ART) neural network [16] with multiple input fields. ART networks apply unsupervised learning to categorize input patterns. They employ bi-directional processes of categorization and prediction to find the best matching category (resonance). They also learn continuously by updating the weights of neural connections at the end of each search cycle. ART may also grow dynamically by allocating a new category node if no match can be found. This type of neural network is chosen as the building block of our model as it enables continuous formation of memory with adjustable vigilance of categorization to control the growth of the network and the level of generalization.

Specifically, fusion ART can be defined as follows: F_1^k and F_2 are the kth input (output) field and the category field of fusion ART (Figure 1) respectively for $k = 1, .., n$. Let $\mathbf{x}^k$ denote F_1^k activity vector and $\mathbf{w}_j^k$ denote the weight vector associated with jth node in F_2. F_1^k is associated with choice parameter $\alpha^k > 0$, learning rate $\beta^k \in [0, 1]$, contribution parameter $\gamma^k \in [0, 1]$, and vigilance parameter $\rho^k \in [0, 1]$. A node J in F_2 is selected through *resonance search* which is the interaction between the application of *choice function* (bottom-up process) and *template matching* (top-down process). The choice function produces activations such that $T_j = \sum_{k=1}^{n} \gamma^k \frac{|\mathbf{x}^k \wedge \mathbf{w}_j^k|}{\alpha^k + |\mathbf{w}_j^k|}$, and through template matching, a node J is selected such that $T_J = \max\{T_j : \forall k, m_j^k = \frac{|\mathbf{x}^k \wedge \mathbf{w}_j^k|}{|\mathbf{x}^k|} \geq \rho^k, \text{for all } F_2 \text{ node } j\}$, where the fuzzy AND operation $\wedge$ is defined by $(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i)$, and the norm $|.|$ is defined by $|\mathbf{p}| \equiv \sum_i p_i$ for vectors $\mathbf{p}$ and $\mathbf{q}$. If no existing F_2 node achieves the criteria such that $\forall j \, \exists k, m_j^k < \rho^k$, an uncommitted F_2 node j' is allocated with $\forall k, \mathbf{w}_{j'}^k = \mathbf{1}$ so that $\forall k, m_{j'}^k \geq \rho^k$ and node j' will be selected with $T_J = T_{j'}$. An equal pair (0,0) could also mean *don't-know* and (1,1) would mean *don't-care* condition.

Given the selected node J, learning takes place by modifying the weight vector $\mathbf{w}_J^k$ such that $\mathbf{w}_J^{k(\text{new})} = (1 - \beta^k)\mathbf{w}_J^{k(\text{old})} + \beta^k(\mathbf{x}^k \wedge \mathbf{w}_J^{k(\text{old})})$. The corresponding weight vector of the chosen F_2 node J can be readout into the input field F_1^k such that $\mathbf{x}^{k(\text{new})} = \mathbf{w}_J^k$.

It is also possible to generalize the knowledge learnt in fusion ART. The generalization can be achieved by lowering the vigilance parameter ρ^k so that slightly different input patterns can still activate the same category. The value of an input item can be paired so that the ART learning represents

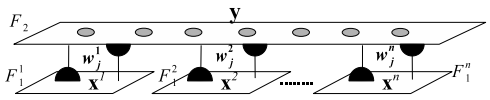

Figure 1. Fusion ART Neural Network

a range of values. Let $\mathbf{I}^k$ be the input vector for $F_1^k, I_i^k \in [0,1]$. $\mathbf{I}^k$ is augmented with $\bar{\mathbf{I}}^k$ such that $\bar{I}_i^k = 1 - I_i^k$. The activity vector $\mathbf{x}^k$ of F_1^k thus augments the input vector $\mathbf{I}^k$ with its complement $\bar{\mathbf{I}}^k$ which are learnt as a $\mathbf{w}_j^k$. If $(w_{ij}^k, \overline{w}_{ij}^k)$ be the corresponding pair of $\mathbf{w}_j^k$, the value of the connection becomes less specific when $w_{ij}^k \neq 1 - \overline{w}_{ij}^k$.

B. Sequential and Hierarchical Representation

The standard configuration of fusion ART does not include a way of representing sequential or hierarchical relations between items. As a part of the proposed model, the fusion ART is extended to associate and group patterns occurring across time. Inspired by the *Item and Order working memory* model in which the temporal order is encoded in the *relative activity* of different node populations [17], fusion ART is extended by implementing constant increments (decrements) to determine the activation value of each selected category node.

The activation pattern of a category field F_n follows a *recency gradient* (First In Last Out or FILO) if the value of the selected node J in F_n is set to τ such that $y_J = \tau$, and τ is updated so that $\tau^{(new)} = \tau^{(old)} + \upsilon$, where $0 \leq \tau \leq 1$, $0 < \upsilon < 1$. τ initializes with 0 at the beginning of the sequence and the sequence ends when it reaches 1. On the other hand, it follows a *primacy gradient* (First In First Out or FIFO), if $y_J = \tau$, and $\tau^{(new)} = \tau^{(old)} - \upsilon$. τ is initialized at 1 at the beginning of the sequence and the sequence ends when it reaches 0.

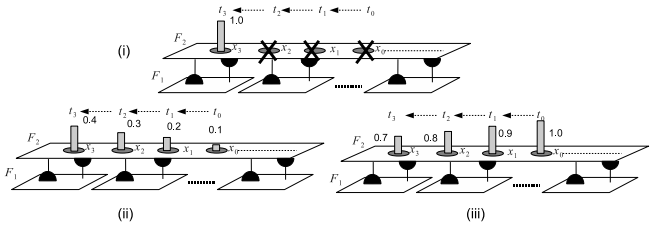

Figure 2. (i) Winner-take-all activation; (ii) Recency gradient activation pattern (FILO), and (iii) Primacy gradient activation pattern (FIFO)

Figure 2 shows different types of sequential ordering compared with the activation using the standard *winner-take-all* activation in fusion ART. The analog pattern formed following the gradient values ordering can directly be learnt and grouped as a category using the same mechanism of resonant search cycle in fusion ART. The sequential pattern can be considered as the input to a fusion ART network. In the proposed architecture, some connected layers of fusion ARTs employ the same technique in their category field allowing another fusion ART to learn the sequential patterns.

C. Planning with iFALCON

Given a triple $\langle \mathcal{S}, \mathcal{G}, \mathcal{P} \rangle$ where $\mathcal{S}$ is the perception (beliefs) about the current state of the world, $\mathcal{G}$ is the set of goals to achieve, and $\mathcal{P}$ is the set of known plans that can be used to solve different types of problem, a planning task is to come up with a plan π to achieve $\mathcal{G}$. A plan π can be described as a tuple $\pi = \langle c_\pi, g_\pi, b_\pi, \upsilon_\pi \rangle$ where c_π is the set of preconditions that makes π applicable to select and initiate, g_π is the goal conditions that will hold after performing or executing π, b_π is a sequence of operations or actions that must be followed in the right order to achieve g_π, and υ_π is an optional attribute of π consisting of a value reflecting the utility or the cost (depending on the problem domain) of π. In hierarchical planning, a plan π can be constructed by directly retrieving it from $\mathcal{P}$ or a plan repository. It is also possible that an action step in b_π is a non-primitive subgoaling action wherein a further planning process is initiated rather than directly changing the task environment.

iFALCON is a neural architecture that arranges different fusion ART networks to emulate the process of planning and plan execution. As shown in Figure 3, iFALCON consists of four input (output) fields and three category fields. F_1^b, F_1^g, F_1^c, and F_1^a denote *beliefs*, *desires*, *critic*, and *action* fields respectively. The *beliefs* field corresponds to the current state of the environment or $\mathcal{S}$ in the problem formulation. The *desires* field represent the goal corresponding to $\mathcal{G}$ in the planning problem formulation. The *critic* reflects the difference between the values in *beliefs* and *desires* or υ_π. The *action* field represents the action to take at a moment to update the environment or initiating further planning (subgoaling).

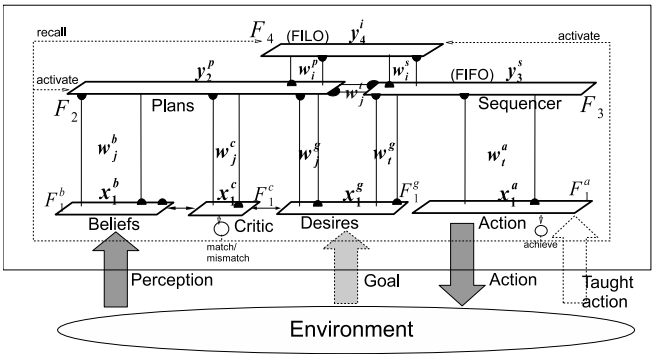

Figure 3. iFALCON Architecture

F_2 and F_3 are category fields of different fusion ARTs representing the set of learnt plans ($\mathcal{P}$) and actions included in every plan respectively. In that case, each node j in F_2 represents a plan π_j and the primacy gradient pattern (FIFO)

of activation formed in F_3 nodes represents the sequence of action in π_j if π_j is selected. This implies that the connection weights $\mathbf{w}_j^b$, $\mathbf{w}_j^g$, $\mathbf{w}_j^t$, and $\mathbf{w}_j^c$ correspond to c_{π_j}, g_{π_j}, b_{π_j}, and v_{π_j} respectively. Figure 4 shows how a symbolic description of a plan corresponds to the neural network structure in iFALCON.

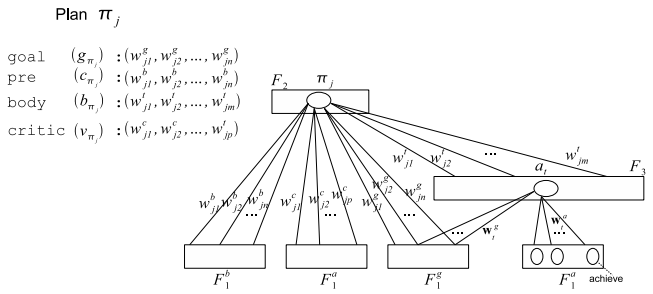

Figure 4. Comparison between symbolic descriptions of plan and the corresponding weighted connections structure

Figure 4 also shows how a non-primitive action corresponds to its neural structure in iFALCON. For example, a subgoal achievement action can be represented as a single node in the input field F_1^a. As the action node in F_3 connects to F_1^g as well, the execution of the achievement action replaces the values of goal field F_1^g with a new subgoal.

On the other hand, F_4 category field gets its inputs from F_2 and F_3 in terms of $\mathbf{y}_2^p$ and $\mathbf{y}_3^s$ vectors respectively. It serves as working memory or a transient buffer holding information about the status of the plan execution. An instance of a plan execution and the overall structure of intention can be stored temporarily by categorizing the inputs ($\mathbf{y}_2^p$ and $\mathbf{y}_3^s$) through resonance search processes following the recency gradient pattern of activations (FILO). Whenever a plan has been successfully executed to achieve its goals, the process can be resumed to the state of the last superordinate plan by selecting and reading out the largest activation of vector $\mathbf{y}_4^i$ in F_4 field.

To realize the basic functionality of frontal cortex as mentioned above, some operations are applied in iFALCON as follows:

1) **Selecting and initiating plans**. Selecting and initiating a plan can be done by applying the resonance search to select a node in F_2 given $\mathbf{x}_1^b$, $\mathbf{x}_1^g$, and $\mathbf{x}_1^c$. The plan body or the sequence of action can then be readout to F_3 field.
2) **Sustaining information for goal achievement**. The content of the plan body can be sustained as the activation pattern in F_3 field as scheduled operations while the highest node is readout and deactivated for execution.
3) **Breaking down goals and subgoals**. At any time point, a non-primitive subgoal (achieve) action may be executed as a part of the plan. This subgoaling action initiates the resonance search in F_4 to sustain the plan and actions pending achievement while the goal values in $\mathbf{x}_1^g$ is replaced by the weights $\mathbf{w}_t^g$ (subgoal).
4) **Monitoring goals achievement**. A matching function is applied to evaluate the difference between F_1^b and F_1^g. In our implementation, we use the function $m_d = \frac{|\mathbf{x}_1^g \wedge \mathbf{x}_1^b|}{|\mathbf{x}_1^g|}$ to evaluate the goal achievement. If the value is greater than or equal a threshold ρ^d then the goal can be considered achieved. Otherwise the plan selection or execution continues.
5) **Hypothesis generation and plans acquisition**. If no plan that fits with the criteria can be found, a new F_2 node is allocated for a new plan while another plan can be selected with less restricted criteria as a hypothesis. If the sequence of hypotheses made eventually achieves the allocated goal, the pattern representing the sequence in F_3 can be learnt as the body of the allocated plan.

Algorithm 1 (iFALCON Execution cycles)**.**

1 WHILE ***True***
2 *Perceive the environment and update* F_1^b
3 IF $m_d = \frac{|\mathbf{x}_1^g \wedge \mathbf{x}_1^b|}{|\mathbf{x}_1^g|} \geq \rho^d$ /*the goal matches with the beliefs*/
4 IF $\mathbf{max}(\mathbf{y}_3) \geq 1$ /* a new sequence has just been formed */
5 $\mathbf{w}_j^{t(\text{new})} = (1-\beta_3)\mathbf{w}_j^{t(\text{old})} + \beta_3(\mathbf{y}_3^t)$ /*learn the sequence*/
6 $\mathbf{w}_j^{c(\text{new})} = (1-\beta_1^c)\mathbf{w}_j^{c(\text{old})} + \beta_1^c(\mathbf{x}_1^c)$ /*learn the critic*/
7 IF $\mathbf{max}(\mathbf{y}_4) > 0$ /* some plans are pending achievements */
8 *readout* F_2 *and* F_3 *from node* i *(***max***) in* F_4*; reset node* i
10 *readout* F_1^g *from node* j *(***max***) in* F_2
11 ELSE ***Finish***
12 ELSE
13 IF $\mathbf{max}(\mathbf{y}_2) \leq 0$ /* no plan is activated or selected */
14 *select* F_2 *node* j *by resonance search*
15 $y_{2.j}^p \leftarrow 1$*;* $y_{2.m}^p \leftarrow 0$*,* $m \neq j$
16 *readout* F_1^g *and* F_3 *from node* j *in* F_2
17 IF $\mathbf{max}(\mathbf{y}_3) \leq 0$ /*no existing plan can be found*/
18 $\mathbf{w}_j^{c(\text{new})} = (1-\beta_1^c)\mathbf{w}_j^{c(\text{old})} + \beta_1^c(\mathbf{x}_1^c)$ /*learn the critic*/
19 *select* F_4 *node* i *by resonance search (recency gradient)*
20 REPEAT /*the start of the forward chaining hypothesis*/
21 $\rho_1^{g(new)} = \rho_1^{g(old)} - \delta$ /*gradually reducing ρ_1^g*/
22 *select* F_2 *node* j *by resonance search*
23 UNTIL *existing* j *is found*
24 *readout* F_1^g *and* F_3 *from* F_2 *node* j *(***max***)*
25 *set action pattern in* F_1^a *to subgoal (achieve) action*
26 *readout* F_2 *and* F_3 *from node* i *(***max***) in* F_4
27 *resonance search* F_3 *with* F_1^g *and* F_1^a*(primacy gradient)*
28 $\mathbf{w}_i^{s(\text{new})} = (1-\beta_i^s)\mathbf{w}_i^{s(\text{old})} + \beta_i^s(\mathbf{y}_3^s); \mathbf{y}_2 = \mathbf{0}$
29 /*store the hypothesis; reset plan field*/
30 *set* ρ_1^g *back to normal (default)*
31 ELSE /*there is an action in F_3 pending execution*/
32 *readout* F_1^a *from* F_3 *node* t *(***max***)*
33 IF *the selected action in* F_1^a *is a subgoal action (achieve)*
34 *select* F_4 *node* i *by resonance search (recency gradient)*
35 *readout* F_1^g *from node* t *in* F_3*; reset* F_2
36 *reset node* t *in* F_3
37 *Execute the action based on* F_1^a

In the current implementation, the strategy used to make the hypothesis is forward chaining wherein the base vigi-

lance ρ_1^g of the *desires* field (F_1^g) is gradually decreasing until an existing plan can be found so that any applicable plan will be selected regardless whether it can achieve the main goal or not. The entire process of hierarchical planning above can be described as pseudo-code in Algorithm 1.

IV. Case Study

We have implemented iFALCON and applied it to solve *blocks world* domain and a real-time first-person-shooter video game called Unreal Tournament. Each problem domain will be illustrated in the following parts.

A. Blocks World

In blocks world domain, the target is to stack blocks from one configuration (initial state) to another configuration (the goal) like shown in Figure 5. For the same goal configuration, there are twelve possible initial state configurations.

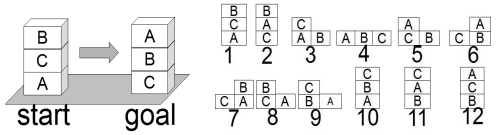

Figure 5. Blocks world problem and different initial blocks configurations.

As a hierarchical planner iFALCON has been successfully applied to solve the blocks world problem by executing plans and decomposing goals and subgoals. The goal condition can be achieved from all twelve initial configurations with minimal prior knowledge (all pre-given plans consist of only a single step of a primitive action) although the steps taken to solve the problem in a planning instance may sometimes not be efficient [18]. The performance in searching for a solution can also be improved by providing a guiding plan (a plan with a sequence of action and subgoals) as an additional prior knowledge [18].

The ability to acquire plans on the fly has also been evaluated in [19] and it has been demonstrated that a reasonable plan can be captured from a planning instance. For example, the following is a symbolic mapping of a plan captured as neural connections in iFALCON in solving the blocks world problem:

```
{goal: [-B_On_Table, C_On_Table, -A_On_Table, -Clear_C,
        -Clear_B, Clear-A],
 pre: [-B_On_Table, -C_On_Table, A_On_Table, -Clear_C,
         Clear_B, -Clear_A],
 body: [{achieve: [Clear_A, B_On_Table]},{achieve:[-Clear_C,
        -B_On_Table]}, {achieve: [-A_On_Table, -Clear_B]}],
 critic: [1]}
```

The minus ('−') sign means negation of the following proposition. However, it is also indicated that the on-line learning feature is still not effective for some initial configurations if only the minimal plans are employed as prior knowledge [19].

In this paper, we further present more results from the evaluation of the iFALCON learning capability in blocks world to confirm that the agent actually learns useful plans. A testing scenario looks at how well iFALCON achieves the goal continuously from a series of different varying initial block configurations. Using the same set of prior knowledge (plans) with a guiding plan as in [18] and [19], 20 independent trials over 500 consecutive series of problem solving episodes are conducted. In a single problem solving episode, a block configuration is selected at random. For each trial and single episode, the number of execution cycles required to achieve the goal, the number of action steps taken to achieve the goal, the difference between the number of action steps taken with the optimal number of steps that can be applied for that particular block configuration, and the number of plans learnt are measured.

In the early stage of the experiment, however, we have realized that if a plan is learnt for one of the possible initial configurations (configuration 11 in Figure 5), further learning may instead corrupt the knowledge for other configurations. This can happen because a plan learnt from the corrupting configuration can have a more general goal criteria (more don't care conditions) that incidentally matches with the original main goal of blocks world. This also indicates that the context of learning from previous trials may influence the effectiveness of the learning if the learning trials are interdependent.

For the reason of space we do not include the early results with the corrupting configuration in this paper. Our direct solution is by taking out the configuration 11 from the set of initial state configurations and rerun the same experiment. Figure 6(i) shows the trends of the average number of execution cycles and action steps without block configuration 11. Figure 6(ii) shows the trends of the average difference between the number of action step and the optimal action possible, and the average number of plans learnt. Figures 6(i) and 6(ii) show that the performance improves and converges. This means also iFALCON can learn useful plans that can be reused to achieve goals in different conditions.

B. Unreal Tournament Gamebot

Unreal Tournament (UT) is a real-time first-person shooter video game. We apply iFALCON as a reasoning engine of a non-player character (NPC) in Unreal Tournament. The objective of this domain is to test whether an iFALCON agent can follow some prescribed plans and goals when the environment is continuous and the agent has only very limited knowledge about the task domain (no information about locations or exact coordinates). This NPC is only tested for its ability to execute plans and the possibility that any plan can be captured as a result of the planning and learning capability driven by the algorithm. The overall performance or improvement as produced by learning is not evaluated. The NPC must accomplish a certain mission to collect certain objects resided in the environment in a particular order. The mission is given explicitly as goals

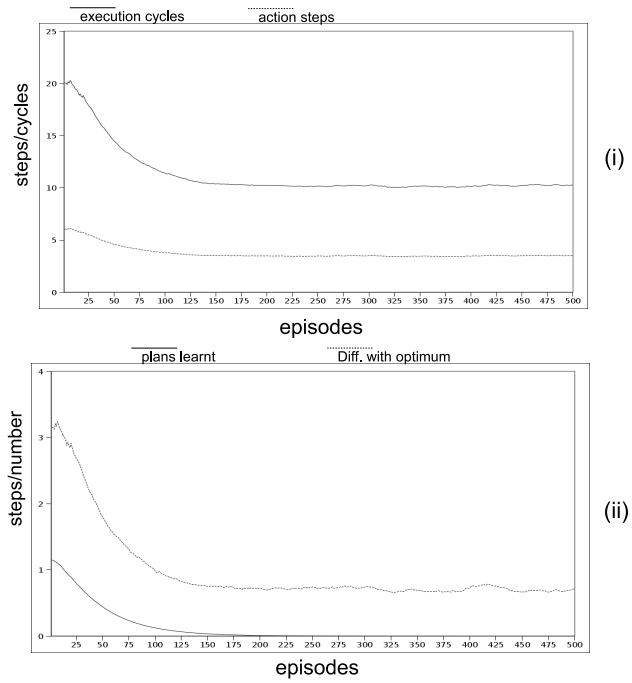

Figure 6. (i) The trend of average over 20 trials of the number of execution cycles and action steps for 500 episodes; (ii) The trend of average over 20 trials of the difference between the number of action step and the optimal action possible, and the number of plans learnt in for 500 episodes

and plans inserted as neural activations and connections in iFALCON. For the reason of space, we omit to show all prior knowledge provided to the agent. Some plans used as prior knowledge in the UT domain are as follows:

```
{goal: [mission_Accomplished],
 pre: [],
 body:
 [{achieve:[got_FlakCannon]},
 {achieve:[got_Rocket]}],
 critic: [1]}

{goal: [got_FlakCannon]},
 pre: [-got_FlakCannon]},
 body:
 [{achieve:[-FlakCannonFar]},
 [{do:[moveto_FlackCannon}],
 critic: [1]}

{goal: [got_Rocket],
 pre: [got_FlakCannon],
 body:
 [{achieve:[-RocketFar]},
 {do:[moveto_Rocket]}],
 critic: [1]}
```

After 824 ticks (1 tick is 100 interaction cycles or about 1 second) the agent can accomplish the mission 6 times, and get 13 items in the right order (get the flak cannon first followed by getting the rocket launcher) which also indicates that the NPC has followed the initial plans correctly. The bot has also captured many plans although most of them consist of a single action step only. Some of them are also redundant (a plan captured is already learnt before) as the basic Algorithm 1 still does not have a mechanism to avoid redundancy. Among the plans captured, some are as follows:

```
{ pre: [-mission_Accomplished, -got_Rocket,
        -got_FlakCannon, WeaponFar, HealthFar,
        -NavpointFar,.........................],
  critic: [1]},
  body: [{achieve: [to_Navpoint]},
         {achieve: [to_Navpoint]}],
  'goal': ['-FlakFar'] }
{ pre: [-mission_Accomplished, -got_Rocket,
         got_FlakCannon, -WeaponFar, HealthFar,
        -NavpointFar.........................],
   critic: [1]},
  body: [{achieve: [to_Navpoint]},
         {achieve: [-NavpointFar']},
         {achieve: [to_Navpoint]},
         {achieve: [to_Navpoint]}],
  'goal': ['-farLauncher'] }
{ pre: [-mission_Accomplished, -got_Rocket,
        -got_FlakCannon, WeaponFar, HealthFar,
        NavpointFar,.........................],
  critic: [1]},
  body: [{achieve: [-NavpointFar']}
         {achieve: [to_Navpoint]},
         {achieve: [to_Navpoint]}],
  'goal': ['-FlakFar'] }
```

The case study in UT domain has shown that iFALCON can function as a plan executor and plan generator simultaneously in a continuous real-time domain. Despite the overall performance and the efficiency and consistency of the learnt plans, the architecture can capture plans that can be useful and meaningful if the domain model and the initial knowledge are considered carefully in the design process.

V. Conclusion

This paper has presented a model of hierarchical planning system realized as a neural network model called iFALCON. The model emulates some functionalities of frontal cortex characterizing a hierarchical planner observed and studied in neuropsychology. The model has been implemented and tested to solve the blocks world problem and to capture plans in the Unreal Tournament real-time video game. Beyond a static hierarchical plan executor, the case study confirms the capability of planning and learning in iFALCON 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 and the context of learning trials. More in-depth studies are required to obtain the complete picture of the characteristics of the plan learning.

The proposed model comprises bi-directional activation pathways which make it possible to select a plan based on different criteria and order. It is also possible to select a plan based on the presentation of action sequences rather than triggered by goals. In the future, it is possible to add the feature of plan recognition beyond the basic functionality of hierarchical planner.

In any case, the proposed model can bridge two different approaches of building planning 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 a system that ultimately deliberate, plan, and learn.

ACKNOWLEDGEMENT

This work was supported by the Singapore National Research Foundation Interactive Digital Media R&D Program, under research Grant NRF2008IDM-IDM004-037.

REFERENCES

[1] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*. Amsterdam: Morgan Kaufman, 2004.

[2] E. D. Sacerdoti, "The nonlinear nature of plans," in *Proceedings of the 4th International Joint Conference on Artificial Intelligence (IJCAI-75)*, 1975, pp. 206–218.

[3] E. K. Miller and J. D. Cohen, "An integrative theory of prefrontal cortex function," *Annual Review of Neuroscience*, vol. 24, pp. 167–202, 2001.

[4] D. Badre and M. D'Esposito, "Is the rostro-caudal axis of the frontal lobe hierarchical?" *Nature Reviews Neuroscience*, vol. 10, pp. 659–669, 2009.

[5] A.-H. Tan, G. A. Carpenter, and S. Grossberg, "Intelligence Through Interaction: Towards A Unified Theory for Learning," in *Proceedings of International Symposium on Neural Networks (ISNN) 2007*, LNCS 4491. Berlin:Springer, 2007, pp. 1098–1107.

[6] X. Wang, "Planning while learning operators," in *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*. AAAI Press, 1996, pp. 229–236.

[7] S. W. Bennet and G. F. Dejong, "Real-world robotics: Learning to plan for robust execution," *Machine Learning*, vol. 23, no. 2–3, pp. 121–161, 1996.

[8] M. Beetz, *Concurrent Reactive Plans: Anticipating and Forestalling Execution Failures*, LNCS 1772. Berlin: Springer, 2000.

[9] O. Ilghami, D. S. Nau, H. Munoz-Avila, and D. W. Aha, "Learning preconditions for planning from plan traces and HTN structure," *Computational Intelligence*, vol. 4, pp. 388–413, 2005.

[10] C. Hogg, H. Munoz-Avila, and U. Kuter, "HTN-MAKER: Learning HTNs with minimal additional knowledge engineering," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI'08)*, 2008, pp. 950–956.

[11] C. Hogg, U. Kuter, and H. Munoz-Avila, "Learning hierarchical task networks for nondeterministic planning domains," in *Proceedings of the Twenty-First International Joint Conference of Artificial Intelligence (IJCAI'09)*, 2009, 1708–1714.

[12] R. Sun, *Duality of the Mind: A Bottom-Up Approach Toward Cognition*. Mahwah: Lawrence Erlbaum, 2002.

[13] G. Baldassarre, "A modular neural-network model of the basal ganglia's role in learning and selecting motor behaviours," *Journal of Cognitive Systems Research*, vol. 3, pp. 5–13, 2002.

[14] L. Shastri, D. J. Grannes, S. Narayanan, and J. A. Feldman, "A connectionist encoding of parameterized schemas and reactive plans," in *Hybrid Information Processing in Adaptive Autonomous Vehicles*, G. Kraetzschmar and G. Palm, Eds. Berlin: Springer Verlag, 1997.

[15] M. Garagnani, L. Shastri, and C. Wendelken, "A connectionist model planning via back-chaining search," in *Proceedings of the Twenty-Fourth Annual Conference of the Cognitive Science Society (CogSci'02) 2002*. Mahwah: Lawrence Erlbaum, 2002, pp. 345–350.

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

[17] S. Grossberg, "Behavioral contrast in short-term memory: Serial binary memory models or parallel continuous memory models?" *Journal of Mathematical Psychology*, vol. 3, pp. 199–219, 1978.

[18] B. Subagdja and A.-H. Tan, "Planning with ifalcon: Towards a neural-network-based bdi agent architecture." in *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'08)*, 2008, pp. 231–237.

[19] ——, "A self-organizing neural network architecture for intentional planning agents," in *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09)*, 2009, pp. 1081–1088.