Integrating Motivated Learning and k-Winner-Take-All to coordinate Multi-Agent Reinforcement Learning

Teck-Hou Teng and Ah-Hwee Tan
School of Computer Engineering
Nanyang Technological University
E-mail:thteng,asahtan@ntu.edu.sg

Janusz A. Starzyk
Russ College of Engineering and Technology
Ohio University, USA
E-mail:starzyk@bobcat.ent.ohiou.edu

Yuan-Sin Tan and Loo-Nin Teow
DSO National Laboratories
E-mail:tyuansin,tloonin@dso.org.sg

***Abstract*—This work addresses the coordination issue in distributed optimization problem (DOP) where multiple distinct and time-critical tasks are performed to satisfy a global objective function. The performance of these tasks has to be coordinated due to the sharing of consumable resources and the dependency on non-consumable resources. Knowing that it can be sub-optimal to predefine the performance of the tasks for large DOPs, the multi-agent reinforcement learning (MARL) framework is adopted wherein an agent is used to learn the performance of each distinct task using reinforcement learning. To coordinate MARL, we propose a novel coordination strategy integrating Motivated Learning (ML) and the k-Winner-Take-All (k-WTA) approach. The priority of the agents to the shared resources is determined using Motivated Learning in real time. Due to the finite amount of the shared resources, the k-WTA approach is used to allow for the maximum number of the most urgent tasks to execute. Agents performing tasks dependent on resources produced by other agents are coordinated using domain knowledge. Comparing our proposed contribution to the existing approaches, results from our experiments based on a 16-task DOP and a 68-task DOP show our proposed approach to be most effective in coordinating multi-agent reinforcement learning.**

I. Introduction

Global objective function of distributed optimization problem (DOP) is typically satisfied by performing multiple distinct and time-critical tasks in a distributed but coordinated manner. In such DOPs, coordination is required due to the sharing of *consumable* resources and the dependency on *non-consumable* resources. Consumable resources are *replenished* by some other tasks and have to be shared among the consuming tasks in an equitable manner. Non-consumable resources are *produced* by tasks using the consumable resources. Team-based activities such as distributed control, robotic teams, automated trading and resource management [1] are examples of real-world problems with such issues.

This work addresses a DOP with multiple single objective tasks. Each task has their own set of discrete states and discrete actions. Due to the adversarial character of the DOP, the performance of the tasks is time-critical and is learned using the multi-agent reinforcement learning (MARL) framework [1]. The task of coordinating the activities of the agents [2] is one of the main issues in MARL. Drawing inspirations from the Motivated Learning (ML) paradigm [3] and the k-Winner-Take-All (k-WTA) approach [4], we aim to propose a novel coordination strategy for coordinating MARL in such DOP.

Several coordination strategies were proposed for applying MARL in small problems [5] [6] [7] and large problems [8] [9] [10]. The solutions for small problems are not known to be scalable to the large problems whereas the large problems seen in earlier works do not comprise tasks with same kind of interaction seen in this work. In addition, none of these works seek to prioritise the allocation of consumable resources to the most urgent tasks. In contrast, this work coordinates MARL in DOPs using the concept of the *pain* signal and the k-WTA approach.

In this work, we approach the above-mentioned DOPs at two levels. At the problem level, we consider it as DOP [11] where multiple tasks are coordinated to satisfy a global objective. At the task level, the performance of each task is learned using reinforcement learning with a self-organizing neural network known as FALCON [12] used as the function approximator. A temporal difference method known as Q-Learning estimates the long-term value of performing the actions in the states. Partitioning the solution of the DOP into multiple stages, each stage is characterised by a specific set of declarative goals [13]. The declarative goal of a task is satisfied by performing the task.

We illustrate such a DOP using a popular computer game known as Starcraft Broodwar (SCBW). Experiments were conducted using a virtual player to solve a DOP simulated using 16-task and 68-task SCBW scenarios. The aim of the experiments is to identify an approach most effective at building up own forces by satisfying the goals of the tasks at each stage using a common set of resources in limited amount of time. In the 16-task scenario, comparisons are made to illustrate the different approaches of implementing the virtual player. In the 68-task scenario, different strategies for selecting the winning tasks are compared.

The presentation of this work continues in Section II with a survey of the related works. The problem formulation is provided in Section III. An overview of the *pain*-based coordination strategy for MARL is provided in Section IV. The use of k-WTA to optimise the use of existing resources is detailed in Section V. An overview of the function approximator is provided in Section VII. The SCBW game is briefly introduced in Section VIII. The experiments and the results are presented in Section IX. Last but not least, Section X contains the conclusion.

II. Related Work

Different approaches were proposed for coordinating multi-agent reinforcement learning (MARL) in small problems comprising cooperative, competitive or cooperative-competitive tasks [1]. WoLF-PHC [6], GIGA-WoLF [5] and Team Q-Learning [7] were illustrated using multiple small problems such as matching pennies, rock-paper-scissors, grid world and soccer game. Without the complexity of larger problems, issues such as learning stability and co-adaptation among the learning agents can be addressed more easily.

Several approaches were also known for coordinating MARL in larger problems [1]. MARL was coordinated in problems with large number of discrete states and actions using the Perceptual Coordination Mechanism and the Observing Coordination Mechanism [8]. Using such an approach, the agents *cooperated with each other* to maximize their own and system-wide utilities. Cooperative behaviour in a pursuit problem comprising four hunter agents and a target agent was learned using a heterogeneous multi-agent architecture [9]. Multi-objective optimization problem in sequential stage games was addressed using distributed stateless learning coupled with different coordination techniques [10]. Using such an approach, the agents learn how to spread themselves to randomly-placed target locations to improve the average partitioning quality. MARL is coordinated in these large problems with agents performing the same types of tasks. In contrast, the MARL framework is used here on large discrete state-action problems with distinct goal-oriented tasks that may either be cooperative or competitive at different times.

Based on the competitive learning (CL) paradigm [14], the Winner-Take-All (WTA) approach finds a winning neuron [15] as it adapts to the input patterns. Popular techniques such as the Self-Organizing Map [16] and the Adaptive Resonance Theory [17] are based on the WTA approach. The k-Winner-Take-All (k-WTA) neural network generalizes the WTA approach to allow for multiple winners and was used in mutually inhibitory networks [18]. Hardware implementation of the k-WTA approach was also considered for real-time signal processing tasks [4]. Motivated by these works, we adopt the k-Winner-Take-All approach and propose a novel strategy for selecting the winning tasks. Unlike known solutions to the knapsack problem [19], our proposed strategy prioritises the allocation of the resources to the most urgent tasks.

Known works using the SCBW game include a bayesian tactician [20] that makes tactical decisions and predicts attacks under constraints and uncertainty. An optimization strategy [21] was proposed to address the build order problem. Reinforcement learning was used to learn the micro-managing of combat units to win against the built-in game AI [22]. A reactive planning agent known as the EISBot [23] and an agent with a task-based architecture known as SCAIL [24] are also known. In particular, the EISBot and the SCAIL are known to play the game well. As the performance of the tasks is not learned, the game is played in a deterministic manner in many of these works. In contrast, this work implements a virtual player where the performance of the tasks is learned using reinforcement learning and coordinated using a strategy that integrates Motivated Learning and k-WTA.

III. The Problem Formulation

In this section, we precede the formulation of our distributed optimization problem (DOP) with a motivating DOP in Section III-A. This is followed by the problem statement in Section III-B.

A. A Motivating Distributed Optimization Problem

Computer game engages human players cognitively by simulating multiple tasks in reduced amount of time. In particular, we find the Starcraft Broodwar (SCBW) game suitable for illustrating our research issues. In this section, we illustrate our work using a motivating DOP comprising four distinct tasks, four types of resources and two types of relations (see Figure 1). The objective here is to produce the Marine and FireBat combat units using the ProduceMarine task and ProduceFireBat task respectively. However, due to the various dependencies these tasks have with the other tasks, several other tasks have to be performed before producing these combat units.

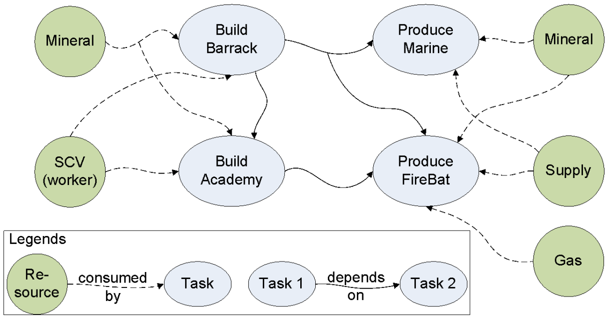

Fig. 1. The complex interaction among the sample tasks

From Figure 1, the performance of BuildBarrack task is seen depending only on resources such as mineral, gas and the Space Construction Vehicle (SCV). However, the BuildAcademy, ProduceMarine and ProduceFireBat tasks are also waiting for the BuildBarrack task to be performed. After the BuildBarrack task has performed, the ProduceMarine and the BuildAcademy tasks can be performed when there are sufficient resources. Likewise, the ProduceFireBat task is waiting on the BuildAcademy task to perform. The performance of the BuildAcademy task leads to the state where the ProduceFireBat task can be performed when the required resources are available.

B. The Problem Statement

The motivating DOP seen in Section III-A illustrates a small *distributed optimization problem* (DOP). This is where multiple tasks have to be performed in a coordinated manner to satisfy a global objective function. The performance of each task is learned using the standard reinforcement learning. Knowing that the optimal action policy can be learned [25], this work is focused on the task of coordinating the activities of the tasks such that the global objective function is satisfied.

In this work, multiple agents are used to perform the tasks in a coordinated and self-organizing manner. Specifically, agent $h \in \mathbf{N}$ performs Task $\tau_h \in \Gamma$ to build up an aspect of the solution for satisfying the global objective function. To perform Task τ_h, agent h is dependent on a set of *consumable* resources $\Lambda_\tau \subset \Lambda$ known as resources and a set of *non-consumable* resources $\Omega_\tau \subset \Omega$ known as technology, where Λ is the set of all resources and Ω is the set of all technologies.

Due to the dependencies existing among the tasks on the resources Λ and the technologies Ω, the DOP can be solved as a coordination problem. Coordination is required due to the dependency on resource Λ because the existing amount of resources W is unlikely to satisfy the resource requirements w_h of task τ_h for all tasks $\tau_h \in \Gamma_\Lambda$ where Γ_Λ is the set of tasks competing for resource Λ.

There is no competition for shared resource Λ when there is sufficient amount of it. Therefore, the **SR**-condition defined below negates the *resource competition* issue.

Definition 1 (Sufficient Resource **SR***):* Given that Task τ_1 consumes λ_{τ_1} amount of resource Λ and Task τ_2 consumes λ_{τ_2} amount of resource Λ.

The **SR**-condition exists between τ_1 and τ_2 when λ amount of shared resource Λ is available such that $\lambda \geq \lambda_{\tau_1} + \lambda_{\tau_2}$.

Also, there is no competition for resource Λ when no resource is shared. Therefore, the **NS**-condition defined below negates the *resource competition* issue.

Definition 2 (No Resource Sharing **NS***):* Given that Task τ_1 consumes λ_{τ_1} amount of resource Λ_1 and Task τ_2 consumes λ_{τ_2} amount of resource Λ_2.

The **NS**-condition exists between τ_1 and τ_2 when $\Lambda_1 \neq \Lambda_2$

Coordination is also required due to the dependency on technologies Ω because task τ_h needs to be performed to produce technology Ω_{τ_h} which is needed by task τ_i to perform. Such a dependency inhibits the concurrent execution of task τ_h and task τ_i. On the other hand, there may exist task τ_j that is not dependent on task τ_h for technology Ω_{τ_h} to perform.

There is no dependency among the tasks when the technologies needed by the tasks to perform are available. Therefore, the **TI**-condition defined below negates the *task dependency* issue.

Definition 3 (Task Independence **TI***):* Let us consider $\Omega_m \in \Omega$ to be the set of missing technologies and $\Omega_e \in \Omega$ to be the set of existing technologies such that $\Omega_m \cap \Omega_e \equiv \emptyset$ and $\Omega_m \cup \Omega_e \equiv \Omega$.

Task τ_1 needs technology $\Omega_{\tau_1} \in \Omega_e$ to perform.

Task τ_2 performs to advance technology $\Omega_{\tau_2} \in \Omega_m$.

The **TI**-condition exists between τ_1 and τ_2 when $\Omega_{\tau_1} \cap \Omega_{\tau_2} \equiv \emptyset$.

A set of tasks $\Gamma_1 \in \Gamma$ sharing limited amount of a set of resources Λ_1 result in *resource competition* within Γ_1. A set of tasks $\Gamma_2 \in \Gamma$ is dependent on a set of technologies $\Omega_1 \in \Omega$ advanced by performing a set of tasks $\Gamma_3 \in \Gamma$ and $\Gamma_2 \cap \Gamma_3 \equiv \emptyset$ result in *task dependency* between Γ_2 and Γ_3. Though these two issues hinder concurrent execution of tasks, such concurrent execution of the tasks is still possible when the following necessary and sufficient condition exists.

Definition 4 (Concurrent Execution of Tasks): Let us use **SR** to denote the *Sufficient Resource* condition, **NS** to denote the *No Resource Sharing* condition, **TI** to denote the *Task Independence* condition and **CE** to denote the condition where concurrent execution of tasks is performed.

The necessary and sufficient condition for **CE** is when there is sufficient resource (**SR**) or ($\vee$) there is no sharing (**NS**) of resources and ($\wedge$) the tasks are independent of each other (**TI**), i.e., $(\mathbf{SR} \vee \mathbf{NS}) \wedge \mathbf{TI} \iff \mathbf{CE}$.

IV. Coordinated Multiagent Reinforcement Learning

This section presents an overview of the pain-based MARL coordination strategy [25] used in this work. A naïve WTA approach where a winning task is selected to execute is employed in this MARL coordination strategy.

A. The Pain Signal

The concept of *pain* from *motivated learning* [3] is used to coordinate MARL and guide reinforcement learning. In this work, *pain* is defined as follows

Definition 5: Pain is defined as a task deficiency known to the agent for the failure to meet its task goal.

From Definition 5, we correlate the *pain* signal p_τ to the task goal ζ_τ using

$$p_\tau = \frac{\zeta_\tau - \gamma_\tau}{\zeta_\tau}, \tag{1}$$

where γ_τ is the current reading of Task τ.

Action policies that recommend effective actions to reduce *pain* signal p_τ are discovered using reinforcement learning. At training iteration n, reward $r_\tau(n)$ of Task τ is derived using

$$r_\tau(n) = \sigma_\tau H_1(|p_\tau(n-1)| - |p_\tau(n)|), \tag{2}$$

where σ_τ is the reward factor, $p_\tau(n)$ is the *pain* signal and $H_1(c)$ is a Heaviside step function defined as

$$H_1(c) = \begin{cases} 1 & c > 0, \\ 0 & c \leq 0. \end{cases}$$

From (2), a reward of σ_τ is given for reducing the pain signal p_τ of Task τ.

B. The Coordination Strategy

The performance of the tasks is coordinated by identifying a winning task τ^*. The tasks are organized into $|\Gamma|$ categories of sub-tasks where Γ is the set of main tasks. From a set of sub-tasks Γ_q of main task q, a winning Subtask τ_q^* is identified using

$$\tau_q^* = \max_{\tau_i \in \Gamma_q} H_2(\varphi_{\tau_i}) p_{\tau_i}, \tag{3}$$

where $\Gamma_q \in \Gamma$ and $H_2(c)$ is a Heaviside step function defined as below

$$H_2(c) = \begin{cases} 1 & c \equiv \text{true}, \\ 0 & c \equiv \text{false}, \end{cases}$$

and the conditional qualifier φ_{τ_i} checks the **TI**-condition using propositional rule set $R_{\tau_i} \equiv \{r_1^{\tau_i}, \ldots, r_{|R_{\tau_i}|}^{\tau_i}\}$. Propositional rule $r_m^{\tau_i}$ is defined as

$$\text{Rule } r_m^{\tau_i} : \text{ IF } \mathbf{X}^{r_m^{\tau_i}} \text{ THEN } \mathbf{Y}^{r_m^{\tau_i}} (\text{ REWARD } R^{r_m^{\tau_i}}),$$

where $\mathbf{X}^{r_m^{\tau_i}}$ is the antecedent, $\mathbf{Y}^{r_m^{\tau_i}}$ is the consequent and $R^{r_m^{\tau_i}} \in [0, 1]$ is the Q-value estimated using (4).

At training iteration n, Subtask τ_q is permissible when $H_2(\varphi_{\tau_q}) = 1.0$. Given that φ_{τ_i} is true iff $\forall m \in \{1, |R_{\tau_i}|\} r_m^{\tau_i}$, there can be no winning Subtask τ_q^* when $\forall \tau_q H_2(\varphi_{\tau_q}) = 0$ or when $\forall \tau_q p_{\tau_q} = 0$.

After selecting a winning Subtask τ_q^* for main task q, a winning main task q^* is identified using $\max_{q \in \Gamma} p_{\tau_q^*}$ where $p_{\tau_q^*}$ is the *pain* signal of winning Subtask τ_q^* of main task q.

C. Staging of Task Goals

The distributed optimization problem (DOP) is segmented into several stages to focus on the performance of specific tasks. Stage e of the gameplay is attained using a set of task goals $\mathbf{G}_e = \{\zeta_\tau^e\}$ for $\tau \in \Gamma$ and $e \in \mathcal{E}$ for $\mathcal{E}$ is the set of all stages of the gameplay.

A *forward* transition $e \rightarrow f$ from stage e to stage f where $e < f$ (semantically) is made when

$$e \rightarrow f \text{ when } \forall \tau \in \Gamma(\gamma_\tau \geq \zeta_\tau^e),$$

where γ_τ is the current reading of Task τ and ζ_τ^e is the task goal of Task τ based on the active task goal set $\mathbf{G}_e$.

A *backward* transition $f \rightarrow e$ from stage f to stage e is made when

$$f \rightarrow e \text{ when } \exists \tau \in \Gamma(\gamma_\tau < \zeta_\tau^e).$$

Using the above approach, the gameplay is segmented into stages to *phase in* multiple sets of task goals during runtime.

V. Strategy for Selecting k Winners

As a contribution of this work, we propose a k-WTA approach for selecting k winning tasks based on the necessary and sufficient conditions (presented in Section III-B) for concurrent execution of tasks. The proposed strategy is a two-step solution to a knapsack problem [19] comprising (1) identifying the most urgent tasks (described in Section V-A) and (2) selecting the better fitting tasks (described in Section V-B).

A. Selecting the more urgent tasks

From (1), the *pain* signal p_τ of Task τ is derived using the current reading γ_τ and the goal task ζ_τ. Over here, we consider Task τ_1 to be more urgent than Task τ_2 based on the following definition.

Definition 6 (Task Urgency): Consider a DOP where there is Task τ_1 with *pain* signal p_{τ_1} and Task τ_2 with *pain* signal p_{τ_2}.

Let us denote the state where only Task τ_1 performs using s_1 and the state where only Task τ_2 performs using s_2.

Assuming agent h has payoff $r_h(s_1)$ for getting to state s_1 and payoff $r_h(s_2)$ for getting to state s_2.

Given that $p_{\tau_1} > p_{\tau_2}$, we have $r_h(s_1) > r_h(s_2)$

And we consider Task τ_1 to be more urgent than Task τ_2 and denote this relation using $\tau_1 \gg \tau_2$

For a set of all sub-tasks Γ_q of main task q, Definition 6 is implemented using (3).

B. Selecting the better-fitting tasks

Individually, the most urgent task may be satisfied by the existing resources. However, the **SR**-condition may not exist for multiple tasks. Therefore, it is necessary to determine the most urgent tasks that are better fit to the existing resources.

Using a DOP with three tasks τ_1, τ_2 and τ_3, we consider Task τ_3 to be a better fit than Task τ_2 to the existing resources based on the following definition.

Definition 7 (Better-Fitting Task): Consider a DOP where there are Task τ_1, τ_2 and τ_3 with *pain* signal p_{τ_1}, p_{τ_2} and p_{τ_3} and, with no **NS**-condition, consume λ_{τ_1}, λ_{τ_2} and λ_{τ_3} amount of shared resource Λ respectively.

Given $p_{\tau_1} > p_{\tau_2} > p_{\tau_3}$, we have $\mathcal{Q}_p = \{\tau_1, \tau_2, \tau_3\}$.

Assuming λ amount of resource Λ and $\lambda_{\tau_1} > \lambda_{\tau_2} > \lambda_{\tau_3}$ such that $\lambda_{\tau_1} < \lambda < (\lambda_{\tau_1} + \lambda_{\tau_2} + \lambda_{\tau 3})$, we consider Task τ_3 to be a better fit than Task τ_2 when **SR**-condition exists for τ_1 and τ_3, i.e, $\lambda_{\tau_3} \leq (\lambda - \lambda_{\tau_1}) < \lambda_{\tau_2}$.

From Definition 6 and Definition 7, we propose our k-WTA approach for coordinating MARL as outlined in Algorithm 1.

Algorithm 1 Our proposed k-WTA approach

```
Require: $\lambda$ amount of resource $\Lambda$
 1: while $\Gamma_q \neq \emptyset$ do
 2:     Select $\tau_q^*$ using (3)
 3:     if $\lambda - \lambda_{\tau_q^*} \geq 0$ then
 4:         add $\tau_q^*$ to $\Gamma_{sr}$ (SR-condition exists)
 5:         $\lambda' = \lambda - \lambda_{\tau_q^*}$
 6:         $\lambda = \lambda'$
 7:     end if
 8:     Remove $\tau_q^*$ from $\Gamma_q$
 9: end while
10: Return $\Gamma_{sr}$
```

In Algorithm 1, a winning task τ_q^* is selected using (3) at Line 2. After that, the resource requirement $\lambda_{\tau_q^*}$ is determined at Line 3 against the current amount λ of resource Λ. Task τ_q^* is added to Γ_{sr} when it satisfies these two necessary and sufficient conditions for concurrent execution of tasks. In this way, MARL is coordinated in a self-organizing manner.

VI. Self-Organizing Property of our k-WTA approach

We focus our analysis of the self-organizing property of Algorithm 1 on the consumption of resources and the attainment of task goals.

A. Resource Consumption

We present the self-organizing property of k-WTA by first considering the consumption of resources using the following lemma.

Lemma 1: For a DOP with a set of tasks Γ_q where the **NS**-condition does not exist, the **TI**-condition exists among the tasks in Γ and λ amount of resource Λ, Algorithm 1 is able to select a set of tasks Γ_{sr} such that the **SR**-condition exists for task $\tau \in \Gamma_{sr}$.

Proof 1: Given a set of tasks Γ_q defined as above exists, we assume $\exists\ \tau \in \Gamma_q\{\lambda_\tau \leq \lambda\}$ and $\left\{ \sum_{\tau \in \Gamma_q}^{|\Gamma_q|} \lambda_\tau \right\} > \lambda$.

In this case, there exists a set of tasks Γ_{sr} such that $\sum_{\tau \in \Gamma_{sr}}^{|\Gamma_{sr}|} \lambda_\tau \leq \lambda$ and $\Gamma_{sr} \subset \Gamma_q$.

From Algorithm 1, a winning task τ^* is identified using (3) and removed from Γ_q

Winning task τ^* is added into Γ_{sr} when $\lambda - \lambda_{\tau^*} \geq 0$, i.e., the **SR**-condition exists.□

B. Task Goal Attainment

We consider the self-organizing attainment of task goals of task $\tau \in \Gamma_q$ using the following lemma.

Lemma 2: For the DOP seen in Lemma 1, the goals of tasks in Γ_q can be attained in a self-organizing manner using Algorithm 1.

Proof 2: From Lemma 1, all the tasks are returned to Γ_q after all tasks in Γ_{sr} executes.

According Definition 6, the performance of tasks in Γ_{sr} changes the urgency of these tasks as they are returned to Γ_q.

Using Algorithm 1, we asssume the **SR**-condition always exist after time Δt such that $\Gamma_{sr} \neq \emptyset$.

$\therefore$ the number of tasks with zero *pain* signal derived using (1) is increased to $|\Gamma_q|$, i.e., all task goals are attained, in time $t < \infty$.$\square$

Using Lemma 1 and Lemma 2, we have provided an analytical foundation of our proposed k-WTA approach for coordinating MARL.

VII. The Self-Organizing Neural Network

A self-organizing neural network [12] that derives from FALCON [26] is used for learning the performance of Task τ. Capable of learning incrementally in real time, FALCON generalizes on the vector patterns without compromising on its prediction accuracy. Action policies are discovered using reinforcement learning in real time. The value of applying the action choices on the states is estimated using Q-Learning.

A. Structure and Operating Modes

Structurally, the FALCON network [26] has a two-layer architecture (see Figure 2), comprising an input/output (IO) layer and a knowledge layer. The IO layer has a sensory field F_1^{c1} for accepting state vector $\mathbf{S}$, an action field F_1^{c2} for accepting action vector $\mathbf{A}$, and a reward field F_1^{c3} for accepting reward vector $\mathbf{R}$. The category field F_2^c in the knowledge layer stores the committed and uncommitted cognitive nodes. Each cognitive node j has three fields of template weights $\mathbf{w}^{ck}$ for $k = \{1, 2, 3\}$.

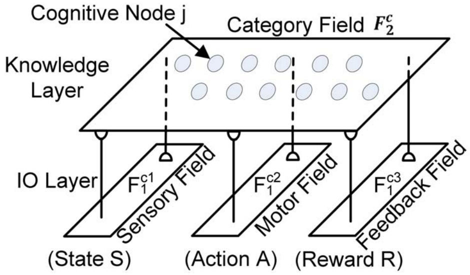

Fig. 2. The FALCON Architecture.

FALCON has three modes of operation - INSERT, PERFORM and LEARN. The Fusion ART algorithm is used to find a winning cognitive node J in these three modes of operation. FALCON operates in the *PERFORM* mode to select action choices for the states, in the *LEARN* mode to learn the effect of these action choices on the states and in the *INSERT* mode to assimilate domain knowledge into itself [27].

B. Value Function Estimation

A temporal difference (TD) method is used to estimate the Q-value of state-action pairs $Q(s, a)$ using feedback from the environment on the performed action a selected using Bounded Q-Learning [28]. At state s', this estimated Q-value is used as the teaching signal to learn the association of state s and the performed action a.

Iterative Value Estimation: The temporal difference method incorporated into FALCON is known as Bounded Q-Learning [28]. It estimates the value of applying action choice a to state s iteratively. The updated Q-value function $Q(s, a)^{(new)}$ is estimated using

$$Q(s, a)^{(new)} = Q(s, a)^{(old)} + \alpha TD_{err}(1 - Q(s, a)), \quad (4)$$

where $\alpha \in [0, 1]$ is the learning parameter and the TD_{err} is the temporal error term derived using

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

where $\gamma \in [0, 1]$ is the discount parameter and the $\max_{a'} Q(s', a')$ is the maximum estimated value of the next state s' and r is the immediate reward value derived using (2)

C. Knowledge Pruning

Ineffective learned knowledge is pruned to facilitate more efficient operation. A confidence-based pruning strategy similar to [26] is adapted to prune the cognitive nodes that encode the ineffective knowledge.

Specifically, cognitive node j has a confidence level c_j where $c_j \in [0.0, 1.0]$and an age σ_j where $\sigma_j \in [0, \mathcal{R}]$. A newly committed cognitive node j has an initial confidence level $c_j(0)$ and an initial age $\sigma_j(0)$. The confidence level c_J of winning cognitive node J is reinforced using

$$c_J^{new} = c_J^{old} + \eta(1 - c_J^{old}),$$

where η is the reinforcement rate of the confidence level. After each training iteration, the age σ_j of cognitive node j is incremented and its confidence level c_j is decayed using

$$c_j^{new} = c_j^{old} - \zeta c_j^{old},$$

where ζ is the decay rate of the confidence level. The age attribute σ_j of cognitive node j prevents pre-mature pruning. Cognitive node j is pruned only when $c_j < c^{rec}$ where c^{rec} is the recommended confidence threshold and $\sigma_j \geq \sigma^{old}$ where σ^{old} is the old age threshold.

VIII. The Simulation Platform

The distributed optimization problem (DOP) is illustrated using a PC-based game known as Starcraft Broodwar (SCBW) seen in Figure 3. Due to the rich body of knowledge and non-trivial gameplay, several groups [23], [24] have used it for their work. Since year 2009, AI competitions using SCBW are organized online and at conferences.

The players can be controlled by either a human player or the built-in AI. Regardless of the type of player, the game is won by eliminating the opponents. There is a selection of three races - terran, zerg or protoss - for the player. The race can either be chosen or randomly assigned to the players. In this work, the players are controlled using the built-in AI and an implementation of our proposed approach.

Botaurus have 30 units:
- 4 Terran Marines
- 12 Terran SCVs
- 2 Terran Firebats
- 2 Terran Medics
- 1 Terran Command Centers
- 2 Terran Supply Depots
- 1 Terran Refinerys
- 3 Terran Barrackss
- 1 Terran Academys
- 2 Terran Bunkers

Run: 0-0 Mode: Learn: PROSEM
HierarchicalMulti-Strategy (k=3)
Version: 4 Productivity: 64.00 Strategy: 1->3

Fig. 3. A screenshot of the SCBW game.

The SCBW game is played at the macro and micro levels. At the macro gameplay, auxiliary but essential tasks such as the resource gathering, building construction, unit production and advancement of technology are performed. The micro gameplay involves the tasks of commanding the units to perform reconnaissance, defend own bases and own units, attack enemy units and to raze the enemy bases.

IX. Performance Evaluation

Experiments were conducted to evaluate the various approaches to coordinate multi-agent reinforcement learning (MARL) for solving distributed optimization problem (DOP). Experiment results from a 16-task DOP (see Section IX-B) and a 68-task DOP (see Section IX-C) are presented. Comparisons made in the 16-task DOP focus on the performance gain from integrating k-WTA and motivated learning. Comparisons made in the 68-task DOP focus on the performance of the strategies for selecting the winning tasks.

A number of parameters and configurations are common to the experiments conducted using these two DOPs. The FALCON parameters as well as the TD-Learning parameters are presented in Table I. The experiments were conducted for 100 game trials. Decisions on the action choices are made at 100 frames interval. The experiment results are the mean values derived using 20 runs of the same experiments and a 10-point sliding window.

TABLE I. Parameters of FALCON and TD Learning

FALCON for $k = \{1, 2, 3\}$	
Choice Parameters α^{ck}	$\{0.1, 0.1, 0.1\}$
Learning Rates β^{ck}	$\{1.0, 1.0, 1.0\}$
Contribution Parameters γ^{ck}	$\{0.33, 0.33, 0.33\}$
Vigilance ρ^{ck}	$\{0.95, 0.0/1.0, \rho^{c3}\}$
ρ^{c3} Adaptation Rate ν	0.95
Confidence ($c_j(0)$, ζ, η)	0.5, 0.0005, 0.5
Pruning - Age threshold σ^{old}	50 iterations
Pruning - Confidence Level c^{rec}	0.65
TD-Learning	
Learning Rate α	0.5
Discount Factor γ	0.1
Initial Q-Value	0.5

A. Evaluation Method

For addressing the distributed optimization problem (DOP), the goal attainment status of the tasks is most suitably illustrated using asset scores derived using the Asset Scoring Methodology (ASM). An asset score μ_τ for Subtask τ is derived using

$$\mu_\tau = 1.0 - \min\left\{\frac{|\zeta_\tau - \gamma_\tau|}{\zeta_\tau}, 1.0\right\}, \tag{5}$$

where ζ_τ is the target level and γ_τ is the current reading of Task τ.

From (5), the asset score μ_q of main task q is derived using

$$\mu_q = \frac{1}{\sum_i^{|\Gamma_q|} \omega_{\tau_i}} \sum_i^{|\Gamma_q|} \omega_{\tau_i} \mu_{\tau_i}, \tag{6}$$

where ω_τ indicates the relative significance of Subtask τ_1 over Subtask τ_2 where $\{\tau_1, \tau_2\} \in \Gamma_q$ and Γ_q is the set of subtasks for main task q.

From (6), the final asset score φ is then derived using

$$\varphi = \frac{1}{\sum_q^{|\Gamma|} \omega_q} \sum_q^{|\Gamma|} \omega_q \mu_q,$$

where Γ is the set of main tasks. In this work, the asset score φ is the main task performance indicator.

B. A 16-*task Distributed Optimization Problem*

In this 16-task DOP, we compare our proposed method (denoted using DWTA) with the $k = 1$ WTA (denoted using WTA) approach, an uncoordinated approach (denoted using UC) and a Monte-Carlo Simulation (denoted using RR). The performance of these approaches is illustrated using the Asset Scores, Active Stages, Number of Active Tasks, Node Population and Decision-Making (DM) time. The task goals of three stages - *opening*, *post-opening* and *mid-game* - and weights of the tasks are presented in Table II. Each game trial is conducted for $10,000$ frames. The asset scores of the UC and RR approaches are based on the *mid-game* stage while those of the WTA and DWTA approaches are based on the active stages.

TABLE II. Weights and Goals of the Tasks

Main Task q	Sub-Task p	Target ζ_τ	Weight ω_τ
Resource	Mine Mineral	$250, 250, 500$	0.95
Management	Gather Gas	$100, 100, 200$	0.75
$\omega_{rsc} = 1.0$	Increase Supply	$26, 26, 48$	0.80
Unit	Produce SCV	$12, 12, 16$	0.50
Production	Produce Marine	$0, 6, 18$	0.75
$\omega_{unit} = 1.0$	Produce FireBat	$0, 2, 6$	0.75
	Produce Medic	$0, 2, 6$	0.75
Building	Refinery	$1, 1, 1$	0.50
Construction	Supply Depot	$2, 2, 6$	0.65
$\omega_{bldg} = 1.0$	Barrack	$0, 2, 2$	0.85
	Bunker	$0, 2, 6$	0.75
	Academy	$0, 1, 1$	0.65
	Engineering Bay	$0, 0, 1$	0.65

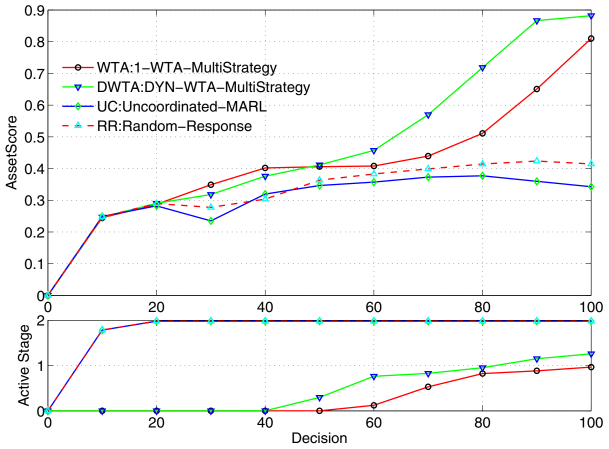

Fig. 4. Comparison of Asset Scores and Active Stages of the different approaches in the 16-task DOP.

Task Performance: Task performance is illustrated using plots of Asset Scores and Active Stage in Figure 4. From Figure 4, the Asset Scores of the UC and RR approach are observed below 0.4 for most decisions. In contrast, the Asset Scores of the WTA and DWTA approaches improve over the decisions. The DWTA approach is observed transiting to the more advanced stages earlier than the WTA approach. Consequentially, the Asset Scores of DWTA approach rise above the WTA approach at around the 50^{th} decision. Therefore, the DWTA approach is considered more efficient than the WTA approach.

Improved Coordination: The top plots in Figure 5 illustrate the number of active tasks. Due to the lack of coordination, the UC and RR approaches are observed with all 16 tasks active. In contrast, the WTA approach has at most one active tasks while the DWTA approach has between 0.55 and 4.72 active

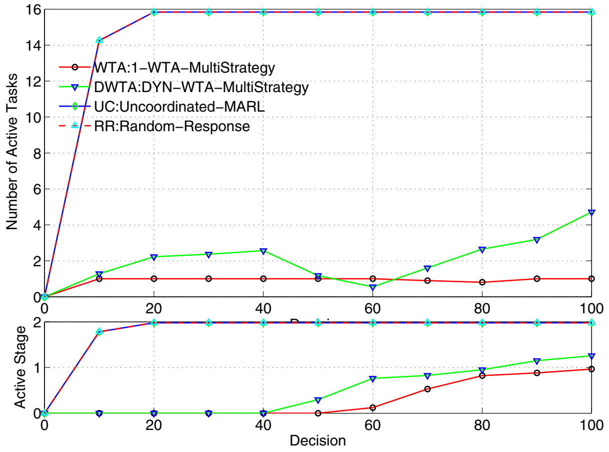

Fig. 5. Comparison of Active Tasks and Active Stages of the different approaches in the 16-task DOP.

tasks. With more active tasks, the DWTA approach reaches the more advanced stages more efficiently than the WTA approach.

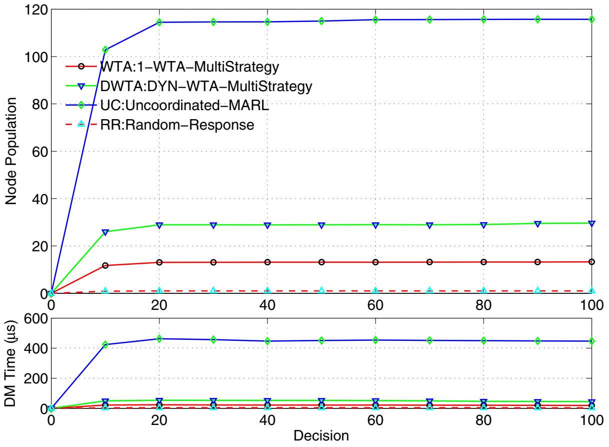

Fig. 6. Comparison of Node Population and Decision-Making Time of the different approaches in the 16-task DOP.

Time and Space Complexity: Figure 6 Top Plot illustrates the node population while Figure 6 Bottom Plot illustrates the decision-making (DM) time. From Figure 6, the UC approach is observed with the largest node population and highest DM time. With zero node population, the RR approach has the least amount of DM time. The DWTA approach is observed with larger node population and higher DM time than the WTA approach. This is due to the higher number of active tasks for the DWTA approach.

C. A 68-*task Distributed Optimization Problem*

Using a 68-task DOP, our proposed DWTA approach is compared with the Greedy (denoted using Greedy) and the Naive (denoted using Naive) methods of the Knapsack problem [19] and a Monte-Carlo Simulation (denoted using RR) . Unlike the Greedy approach, the Naive approach does not consider the remaining tasks when it encounters a task whose resource requirement exceeds the remaining resources. Unlike these two methods, our DWTA approach is a two-step process that prioritises the allocation of the limited resources to the most urgent tasks. The WTA approach is also included as a benchmark comparison in this 68-task DOP.

Specific to this 68-task DOP, each game trial is conducted for $30,000$ frames. Five stages of task goals are used for the 68 tasks. The asset scores are computed using the task goals at the final stage. The same set of parameters of FALCON and TD learning seen in Table I are used. Not shown due to lack of space, the weights and task goals are specified in a similar manner seen in Table II.

Asset Scores: From Figure 7, plots of the asset scores from 300 decisions show our DWTA approach having higher asset scores from around 80^{th} decision. Using the same coordination strategy [25] but different winning task selection strategies, the Naive and Greedy approaches are seen less effective than our DWTA approach. Though effective for the smaller 16-task DOP, the WTA approach is seen with lower asset scores in this larger 68-task DOP. The flat extrapolation of the asset scores of the WTA and RR approaches indicate earlier termination of the game trials as it is eliminated by the opponent.

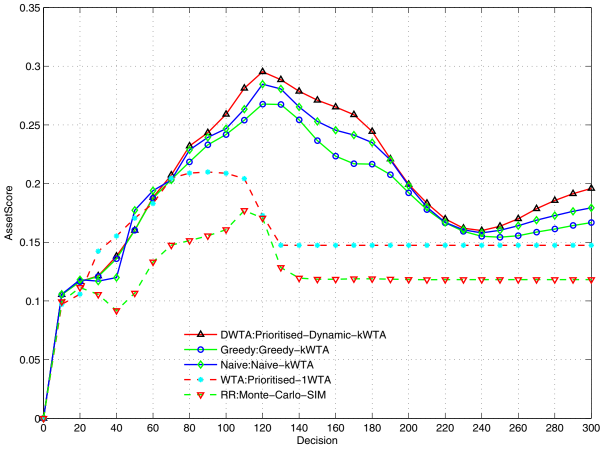

Fig. 7. Comparison of the Asset Scores and Active Stages of the different approaches in the 68-task DOP.

Final Outcome: So far, we concentrated on the asset score as the main performance measure. To demonstrate that this measure indeed indicates better performance, we checked the final outcomes of the game. Playing against the built-in AI opponent, each game trial has a final outcome of loss (0), draw (1) or win (2). From Figure 8, the plots of the final game outcome of 100 game trials show our DWTA approach has the highest mean score of around 0.75. In sharp contrast, the WTA and the RR approaches have the lowest mean score of almost 0.0. Less effective than our DTWA approach, the Naive approach is seen having a higher mean score than the Greedy approach. Such observations affirm the observations made on the plots of the Asset Scores and the Active Stages seen in the top and bottom plots of Figure 7 respectively.

X. Conclusion

We have proposed the integration of k-Winner-Take-All approach (k-WTA) to a ML-based coordination strategy [25] of multiagent reinforcement learning (MARL) for solving distributed optimization problem (DOP). In this work, given the existing resources, we aim to get the maximum number of the most urgent tasks to execute. Using our definition of *Task Urgency* and *Better-Fitting Task*, we proposed the DWTA approach and provided an analytical illustration of its self-organizing property. This is to give the confidence that the various task goals can be attained in time $t < \infty$.

We evaluate the proposed approach by conducting experiments using a 16-task DOP and a 68-task DOP simulated using the Starcraft Broodwar (SCBW) game. Using the 16-task DOP, we compare our proposed DWTA approach with a

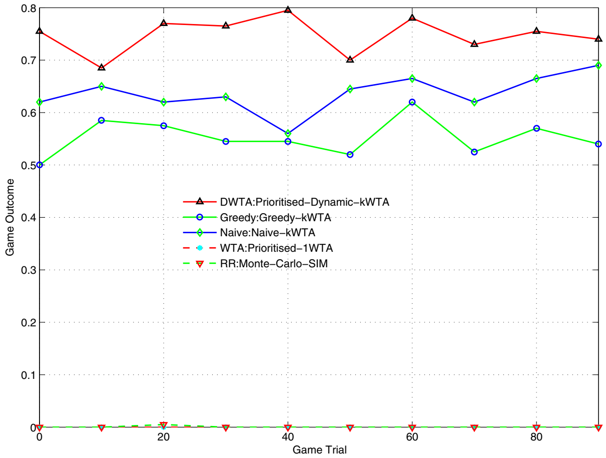

Fig. 8. Comparison of the final game outcome of the different approaches in the 68-task DOP.

single-winner WTA approach, an uncoordinated approach and a random response approach. Task performance is evaluated using the Asset Score, Active Stages, Number of Active Tasks, Node Population and Decision-Making Time. Comparing to the other approaches, the experimental results show our proposed approach is better at getting more tasks to execute. The advanced stages are reached earlier using such an approach. However, due to the higher number of active tasks, our proposed approach has slightly larger node population than the single-winner WTA approach.

Using the 68-task DOP, we compare our proposed strategy for selecting the winning tasks to the Greedy and the Naive approaches for solving the knapsack problem [19]. Plots of the asset scores and the final outcome show our proposed DWTA approach to be more effective when compared to the Greedy and Naive approaches. In addition, the WTA approach effective for the smaller 16-task DOP has not performed as well in the larger 68-task DOP.

In this work, we have illustrated our proposed contributions by implementing a virtual player to a simulated 16-task DOP and a simulated 68-task DOP. The 16-task and 68-task DOPs are simulated for a fixed duration of $10,000$ and $30,000$ frames respectively. In our subsequent work, we will include more tasks to build up to the capability of playing the SCBW game. We shall then evaluate the strength of our work by playing against the built-in AI and against the human players. In addition, we will apply our work to real-world problem domains with similar or higher level of complexity.

Acknowledgments

The reported work is performed for the Human Multiple-Memory System and Learning project in collaboration with DSO National Laboratories, Singapore and was partially supported by the grant from the National Science Centre DEC-2011/03/B/ST7/02518.

References

[1] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, no. 2, pp. 156–172, 2008.

[2] Y.-C. Choi and H.-S. Ahn, "A survey on multi-agent reinforcement learning: Coordination problems," in *Proceedings of the IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications*, 2010, pp. 81–86.

[3] J. A. Starzyk, J. T. Graham, P. Raif, and A.-H. Tan, "Motivated learning for the development of autonomous systems," *Cognitive Systems Research*, vol. 14, no. 1, pp. 10–25, 2012.

[4] T. Kwon and M. Zervakis, "K-WTA networks and their applications," *Multidimensional Systems and Signal Processing*, vol. 6, no. 4, pp. 333–346, 1995.

[5] M. Bowling, "Convergence and no-regret in multiagent learning," *Advances in neural information processing systems*, vol. 17, pp. 209–216, 2005.

[6] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artificial Intelligence*, vol. 136, no. 2, pp. 215 – 250, 2002.

[7] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning." in *Proceedings of ICML*, vol. 94, 1994, pp. 157–163.

[8] O. Abul, F. Polat, and R. Alhajj, "Multiagent reinforcement learning using function approximation," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 30, no. 4, pp. 485–497, 2000.

[9] Y. Ishiwaka, T. Sato, and Y. Kakazu, "An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning," *Robotics and Autonomous Systems*, vol. 43, no. 4, pp. 245 – 256, 2003.

[10] T. Kemmerich and H. Buning, "Coordination in large multiagent reinforcement learning problems," in *Proceedings of WI-IAT*, vol. 2, 2011, pp. 236–239.

[11] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic and Logical Foundations*. Cambridge University Press, 2009.

[12] T.-H. Teng, A.-H. Tan, and L.-N. Teow, "Adaptive computer-generated forces for simulator-based training," *Expert Systems with Applications*, vol. 40, no. 18, pp. 7341–7353, 2013.

[13] M. B. van Riemsdijk, M. Dastani, and M. Winikoff, "Goals in agent systems: A unifying framework," in *Proceedings of AAMAS*, 2008, pp. 713–720.

[14] S. Haykin, *Neural Networks: a comprehensive foundation*, 2^{nd} ed. Prentice Hall, 1999.

[15] W. Banzhaf and H. Haken, "Learning in a competitive network," *Neural Networks*, vol. 3, no. 4, pp. 423 – 435, 1990.

[16] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[17] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, pp. 54–115, 1987.

[18] W. J. Wolfe, D. Mathis, C. Anderson, J. Rothman, M. Gottler, G. Brady, R. Walker, G. Duane, and G. Alaghband, "k-Winner networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 310–315, 1991.

[19] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer-Verlag, 2004.

[20] G. Synnaeve and P. Bessiere, "A bayesian tactician," in *Proceedings of the Computer Games Workshop at the European Conference of Artificial Intelligence*, 2012, pp. 1–7.

[21] D. Churchill and M. Buro, "Build order optimization in starcraft." in *Proceedings of AIIDE*, 2011.

[22] S. Wender and I. Watson, "Applying reinforcement learning to small scale combat in the real-time strategy game starcraft: Broodwar," in *Proceedings of CIG*, 2012, pp. 402–408.

[23] B. G. Weber, M. Mateas, and A. Jhala, "Building human-level AI for real-time strategy games," in *Proceedings of the AAAI Fall Symposium on Advances in Cognitive Systems*, 2011, pp. 329–336.

[24] J. Young, F. Smith, C. Atkinson, K. Poyner, and T. Chothia, "SCAIL: An integrated Starcraft AI system," in *Proceedings of CIG*, 2012, pp. 438–445.

[25] T.-H. Teng, A.-H. Tan, J. A. Starzyk, Y.-S. Tan, and L.-N. Teow, "Integrating self-organizing neural network and motivated learning for coordinated multi-agent reinforcement learning in multi-stage stochastic game," in *Proceedings of IJCNN*, 2014, accepted.

[26] A.-H. Tan, "FALCON: A Fusion Architecture for Learning, Cognition, and Navigation," in *Proceedings of IJCNN*, 2004, pp. 3297–3302.

[27] T.-H. Teng, Z.-M. Tan, and A.-H. Tan, "Self-organizing neural models integrating rules and reinforcement learning," in *Proceedings of IJCNN*, June 2008, pp. 3770–3777.

[28] A.-H. Tan, N. Lu, and D. Xiao, "Integrating Temporal Difference Methods and Self-Organizing Neural Networks for Reinforcement Learning with Delayed Evaluative Feedback," *IEEE Transactions on Neural Networks*, vol. 19, no. 2, pp. 230–244, February 2008.