Integrating Self-Organizing Neural Network and Motivated Learning for Coordinated Multi-Agent Reinforcement Learning in Multi-Stage Stochastic Game

Teck-Hou Teng, Ah-Hwee Tan, Janusz A. Starzyk, Yuan-Sin Tan, Loo-Nin Teow

***Abstract*— Most non-trivial problems involve the performance of multiple goal-oriented tasks. Each task is performed to satisfy their goal. Coordinating the performance of the tasks is required due to the dependencies among the tasks and the sharing of resources. In this work, an agent learns the performance of a task using reinforcement learning with a self-organizing neural network as the function approximator. We propose a novel coordination strategy based on Motivated Learning (ML) to coordinate reinforcement learning of multiple agents. Specifically, we adapt the ML idea of using *pain* signal to overcome the resource competition issue. Dependency among the agents is resolved using domain knowledge of their dependence. To avoid domineering agents, the task goals are staggered over multiple stages. A stage is completed by attaining a particular combination of task goals. Results from our experiments conducted using a popular PC-based game known as Starcraft Broodwar show multiple tasks can be performed effectively using our proposed coordinated learning strategy in a self-organizing manner.**

I. Introduction

In application domains such as distributed control, robotics and automated trading [16], coordinated performance of tasks is required due to the dependencies among the tasks and the sharing of resources. Dependencies among the tasks exist as the pre-requisites to the more advanced tasks. Competition for resources among the tasks exists due to the finite amount and the finite replenishment rate of the shared resources. Using multi-agent reinforcement learning (MARL), an agent learns the performance of a task using reinforcement learning with a self-organizing neural network as the function approximator.

Many works are known for applying MARL on small and large problems [16][1]. Use of MARL on small problems comprising of agents with different patterns of interaction includes [17][18][2][3][19]. Use of MARL on larger problems such as pursuit problem and adversarial food-collecting world (AFCW) problem includes [20][21][22]. However, the use of MARL is not seen from our survey of works [4][5][6][7] for problems similar to this work.

In this work, we proposed a novel coordination strategy for conducting MARL in large problems of practical interest illustrated using a popular PC-based game known as Starcraft Broodwar (SCBW). Capable of incremental learning in real time, an ART-based neural network known as FALCON [23] is used as the function approximator for learning the performance of the task. Inspired by [24], the agents are coordinated using *pain* signal derived with respect to the task goals. The stochastic game is segmented into several stages. Different sets of task goals for the same set of tasks are used for the stages. Results from our experiments conducted using SCBW show that multiple tasks can be performed most effectively using our proposed coordinated learning strategy.

Corresponding author: Teck-Hou Teng (email: thteng@ntu.edu.sg). This work is supported by the DSO National Laboratories under Research Grant DSOCL11258 and by the National Science Centre under Research Grant DEC-2011/03/B/ST7/02518.

The presentation of this work continues in Section II with survey of MARL works and recent applications of learning techniques in SCBW. The problem formulation is provided in Section III. Details of the coordination strategy are presented in Section IV. This is followed by a succinct presentation of FALCON in Section V. The SCBW game is briefly introduced in Section VI. Section VII presents the experiments and the analysis of the experimental results. Section VIII contains the conclusion.

II. Related Work

In this section, we survey existing coordination strategies for multi-agent reinforcement learning (MARL) and the use of learning techniques in the Starcraft Broodwar (SCBW) problem domain. The SCBW is a popular choice for AI research because it is a challenging strategic game with multiple goal-oriented tasks.

The coordination problem in MARL has been addressed for fully cooperative, fully competitive and mixed tasks [16]. In [25], a Fuzzy Subjective Task Structure was proposed to solve fully cooperative tasks problem using reinforcement learning. Team-Q learning [19] is applied to dynamic fully cooperative tasks while FMQ algorithm [2] was proposed for the static version. On the other hand, fully competitive tasks can be addressed using MiniMax-Q [3]. Static problems comprising of tasks with a mixture of these two types of interactions can be addressed using GIGA-WoLF[17] while WoLF-PHC [18] addresses the dynamic version of such problem. Many of these techniques address specific type of task interaction in small problems represented using matrices [16].

The MiniMax-Q algorithm [3] adapted standard Q-learning to stochastic games. However, the opponent's action is considered in the value function of MiniMax-Q. In contrast, our proposed coordination strategy does not need to have such consideration. In [8], coordination of multiple agents is learned using the local state information of other agents. However, it is unrealistic to assume the availability of state information of other agents for the problem addressed in this work. In [9], a channel-based exogenous coordination language known as Reo was proposed for specifying the dynamic composition of multi-agent system. In [10], expert

coordination knowledge was used to restrict the joint action space and direct exploration.

Uses of approximate MARL in large and more practical problems are also known. Two coordination mechanisms [20] were proposed for the AFCW problem. A heterogeneous multiagent architecture [21] was proposed to address pursuit problem with continuous states. In another work, normalized Gaussian network was used as the function approximator [22] in a two-chasers-one-prey problem. While none of them addresses the type of problem addressed here, they are successful in solving their chosen problems.

Different approaches are known for problems similar to this work. Reinforcement learning was used to discover strategies to score close to 100% win against the built-in AI [6]. Evolutionary computation was also used to evolve tactical combat AI [4]. Capable of playing the full SCBW game, the EISBot [5] was a reactive planning agent with goal-driven autonomy while the SCAIL [7] employed a task-based architecture. In another work, an Adaptive Strategy Decision mechanism was proposed to play the SCBW game in stages[15]. However, none of them addresses the same problem using the MARL approach.

III. The Problem Formulation

We begin this section with a motivating problem domain in Section III-A. This is followed by the problem statement in Section III-B.

A. A Motivating Problem Domain

In this section, we include a motivating problem domain to illustrate the task dependency and resource competition issues. The SCBW game is chosen because it suitably illustrates these two issues. In this illustration, there are four categories of tasks - Task 1 Resource Management, Task 2 Unit Production, Task 3 Building Construction and Task 4 Tactical Command - seen in Figure 1 with the complex interaction seen in Figure 2.

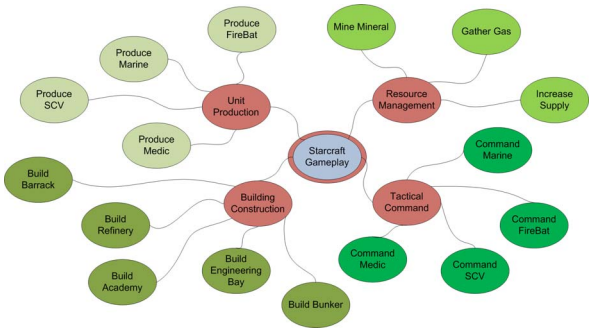

Fig. 1. The tasks performed using multiple agents

From Figure 2, Space Construction Vehicles (SCVs) are required for the performance of all sub-tasks of Task 1. In this respect, Task 1-1, Task 1-2 and Task 1-3 compete with each other for the use of SCV. In addition, Task 1-2 is dependent on Task 3-1 for the construction of Refinery over an unoccupied gas well. The replenishment of supply level needs an SCV and minerals to construct Supply Depot. In turn, Task 2 and Task 3 depend on Task 1 for the consumption of relevant resources.

The sub-tasks of Task 2 compete with each other, Task 3 and Task 1-3 for the consumption of resources. In addition, Task 2-2 to Task 2-4 are dependent on Task 3-2 for the construction of Barrack. Task 2-3 and Task 2-4 are also dependent on Task 3-4 for the construction of Academy. In turn, Task 4-1 to Task 4-4 depends on Task 2-1 to Task 2-4 respectively. In addition, Task 2-3 and Task 2-4 depend on Task 1-2 for the gas resources and is transitively dependent on Task 3-1. This is because there will be no gas resource without any refinery.

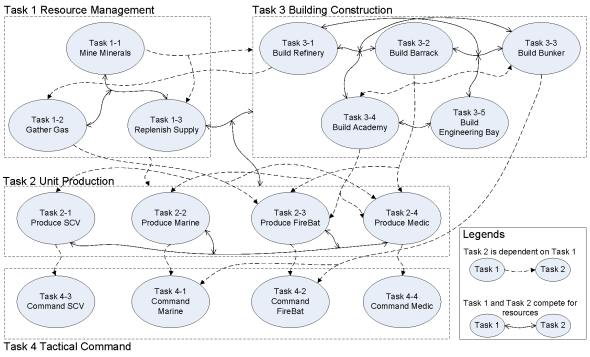

Fig. 2. The complex interactions among the tasks

Task 3 depends on Task 1 for the minerals. Yet, Task 3 also competes with Task 1 for use of SCVs. This is because SCVs are required for the construction of buildings in Task 3 and for the replenishment of the resources in Task 1. Task 3-3 and Task 3-4 depend on Task 3-2 for the construction of Barrack. From the onset, only Task 3-1, Task 3-2 and Task 3-5 compete with each other for resources. Task 3-1 competes with Task 3-3 and Task 3-4 for resources only after it builds the first Barrack.

Task 4-1 and Task 4-2 depends on Task 3-3 for the construction of Bunker. This is because Task 4-1 and Task 4-2 command their respective units to move into the Bunker. In addition, Task 4-1 depends on Task 2-2 for the production of Marines and Task 4-2 depends on Task 2-3 for the production of FireBat. In general, Task 4 does not compete with Task 1 and Task 3 and it is only transitively dependent on Task 1 and Task 3.

B. The Problem Statement

This work addresses a MARL problem in a stochastic game defined as follows

Definition 1 (Stochastic Game): A stochastic game is a tuple (Q, N, A, P, r) where:

- $\mathbf{Q}$ is a finite set of games;
- $\mathbf{N}$ is a finite set of agents, indexed by h;
- $\mathbf{\Gamma} = \tau_\mathbf{1} \times \cdots \times \tau_\mathbf{h}$, where $\tau_\mathbf{h}$ is a finite set of actions available to Agent h.
- $P : \mathbf{Q} \times \mathbf{\Gamma} \times \mathbf{Q} \mapsto [0, 1]$ is the transition probability function.
- $\mathbf{R} = (r_1, \cdots, r_h)$, where $r_h : \mathbf{Q} \times \mathbf{\Gamma} \mapsto \mathbb{R}$ is a real-valued payoff function for Agent h.

The stochastic game does not have a normal form because a function approximator [23] is used to generalize on the states. Using reinforcement learning, P is approximated based on the actual distribution of states reached in the

game [27]. The real-valued payoff function $\mathbf{R}$ is derived using our proposed method that integrates need for resource with proper action as described in Section IV-A.

In this stochastic game, Agent h performs Task $\tau_h \in \Gamma$ to build up a particular aspect of the stochastic game. We aim to get multiple agents to perform multiple tasks efficiently in a coordinated and self-organizing manner.

The performance of Task τ is dependent on the availability of a set of depletable resources $\Lambda_\tau \subset \Lambda$ and a set of non-depletable resource known as technologies $\Omega_\tau \subset \Omega$ where Λ is the set of all resources and Ω is the set of all technologies.

When the agent performs task τ, it consumes λ_τ amount of resource λ where $\lambda \in \Lambda_\tau$. Given two tasks τ_1 and τ_2, *resource competition* between τ_1 and τ_2 exists according to Definition 2.

Definition 2 (Resource Competition): Competition for resource λ occurs between Task τ_1 and Task τ_2 when the following conditions exist.

- Task τ_1 consumes λ_{τ_1} amount of resource $\lambda \in \Lambda_{\tau_1}$ and Task τ_2 consumes λ_{τ_2} amount of resource $\lambda \in \Lambda_{\tau_2}$ such that $\Lambda_{\tau_1} \cap \Lambda_{\tau_2} = \lambda$
- For λ_c amount of resource λ, there is insufficient amount of resource λ to satisfy Task τ_1 and Task τ_2, i.e., $\lambda_c < \lambda_{\tau_1} + \lambda_{\tau_2}$

Due to competition for resource λ between Task τ_1 and Task τ_2, either Task τ_1 or Task τ_2 may fail to perform if there is not enough resources to perform both tasks. Therefore, task performance has to be coordinated to satisfy the following coordination criterion.

Coordination Criterion 1: *Let s_1 denote the state where Task τ_1 is performed and Task τ_2 is blocked and s_2 denote the state where Task τ_1 is blocked and Task τ_2 is performed.*

From Definition 1, for state s_1, agent h has payoff $r_h(s_1)$ and, for state s_2, agent h has payoff $r_h(s_2)$.

Assuming $\lambda_{\tau_1} < \lambda_{\tau_2}$ and $r_h(s_2) > r_h(s_1)$, Task τ_1 will have to be blocked to allow resource λ to accumulate to λ_{τ_2} amount such that Task τ_2 can be performed.

Besides resource competition, multiple tasks may depend on each other such that task τ_1 produces a technology needed by task τ_2. In this respect, a *task dependency* scenario between Task τ_1 and Task τ_2 exists according to Definition 3.

Definition 3 (Task Dependency): We consider Task τ_1 to be dependent on Task τ_2 when the following condition exists

- Performance of Task τ_1 is dependent on a set of technologies Ω_{τ_1}
- Performance of Task τ_2 advances a set of technologies Ω_{τ_2}
- $\Omega_{\tau_1} \cap \Omega_{\tau_2} \neq \emptyset$

From Definition 3, Task τ_1 and Task τ_2 have to be coordinated to satisfy the following coordination criterion.

Coordination Criterion 2: *Assume there is sufficient resource λ for Task τ_1 and Task τ_2 and Task τ_1 is dependent on Task τ_2 as defined in Definition 3.*

Let s_1 denote the state where Task τ_1 is performed before Task τ_2 and s_2 denote the state where Task τ_2 is performed before Task τ_1.

From Definition 1, for state s_1, we have payoff $r_h(s_1)$ and, for state s_2, we have payoff $r_h(s_2)$.

It is known that $r_h(s_1) < r_h(s_2)$ because Task τ_1 is dependent on Task τ_2.

$\therefore$ Task τ_1 will have to be blocked to allow Task τ_2 to perform.

IV. Coordinated Multiagent Reinforcement Learning

Task τ is performed to either accumulate resource $\lambda_\tau \in \Lambda$ or advance technology $\Omega_\tau \in \Omega$. Perrformance of multiple tasks is learned and coordinated using a *pain* signal.

A. The Pain Signal

Motivated learning (ML) [28] is a recent machine learning paradigm proposed to complement reinforcement learning. Primitive *pain* signal is used to motivate learning in a self-organizing manner. In [24], a pain-based neuronal structure was proposed to get an agent to respond dynamically to non-stationary hostile environments. In this work, we adopt the concept of the *pain* signal to coordinate the learning and performance of multiple tasks. Unlike [24], an agent here needs only to satisfy the goal of a task.

From *motivated learning* [24], *pain* related to the lack of resources is used to coordinate MARL and guide reinforcement learning. Specifically, *pain* is defined as follows.

Definition 4: Pain is defined as a task deficiency of the agent due to the failure to meet its task goal.

Over time, by addressing the cause in a consistent and effective manner, *pain* can be reduced and eliminated. Action policies identifying the appropriate actions are identified using reinforcement learning.

From Definition 4, *pain* is correlated to the task goals. In this respect, the pain signal p_τ for Task τ is derived using the task goal ζ_τ and the current level γ_τ below.

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

From (1), the pain signal p_τ will be $p_\tau > 0$ when $\gamma_\tau < \zeta_\tau$, $p_\tau < 0$ when $\gamma_\tau > \zeta_\tau$ and $p_\tau = 0$ for when $\gamma_\tau = \zeta_\tau$. Task τ becomes inactive when $\gamma_\tau = \zeta_\tau$ and active when $\gamma_\tau < \zeta_\tau$.

To guide reinforcement learning, the pain signal $p_\tau(n)$ is used at training iteration n to derive reward $r_\tau(n)$ of task τ using

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

where $p_\tau(n-1)$ is the pain signal at iteration $n-1$, σ_τ is the reward factor 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. In addition, *coordination criterion 1* can be satisfied by using the pain signal p_τ to coordinate MARL.

B. The Coordination Strategy

We propose a novel coordination strategy to satisfy the coordination criteria defined in Section III-B. Using this strategy, tasks with larger *pain* signal have priority over tasks with smaller *pain* signal. Unlike [16], with the use of *pain* signal, our coordination strategy does not need to consider the actions of the other agents.

Using the winner-take-all (WTA) approach, the winning task τ^* is a *permissible* task. In this problem formulation, 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 sub-task τ_q^* is identified using the *Subtask Competition* process defined below.

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

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} is evaluated using propositional rule set $R_{\tau_i} \equiv \{r_1^{\tau_i}, \ldots, r_{|R_{\tau_i}|}^{\tau_i}\}$. Specifically, φ_{τ_i} is true iff $\forall m \in \{1, |R_{\tau_i}|\}(r_m^{\tau_i})$. The term $H_2(\varphi_{\tau_q})$ is included to satisfy *coordination criterion 2*.

Propositional rule $r_m^{\tau_q}$ for $m \in \{1, \ldots, |R_{\tau_q}|\}$ is defined as

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

where $\mathbf{X}^{r_m^{\tau_q}}$ is the antecedent set, $\mathbf{Y}^{r_m^{\tau_q}}$ is the consequent set and $R^{r_m^{\tau_q}} \in [0, 1]$ is the Q-value estimated using (5).

At training iteration n, sub-task τ_q is permissible when $H_2(\varphi_{\tau_q}) = 1.0$. There can be no winning sub-task τ_q^* when $\forall \tau_q H_2(\varphi_{\tau_q}) = 0$ or when $\forall \tau_q p_{\tau_q} = 0$.

It is insufficient to use just the *pain* signal to coordinate MARL. It is also necessary to determine whether the conditions are right for Task τ_1 and Task τ_2 to be performed. Such task conditions, specified as R_{τ_1} and R_{τ_2}, are semantic knowledge known as the *task pre-requisites*.

Self-Organizing Property: There is also a non-zero probability where $p_{\tau_1} = p_{\tau_2}$ and $H_2(\varphi_{\tau_1}) = H_2(\varphi_{\tau_2}) = 1$. In such a circumstance, it is sufficient to randomly decide on either task τ_1 or task τ_2. This is because by performing either of the tasks, the circumstance where $p_{\tau_1} = p_{\tau_2}$ shall cease to exist.

A winning main task $q^* \in \mathbf{Q}$ has to be identified after selecting a winning sub-task τ_q^* for main task q. The winning main task q^* is identified using the *Main Task Competition* process defined below.

$$q^* = \max_{q \in \Gamma} p_{\tau_q^*} \quad (4)$$

where $p_{\tau_q^*}$ is the pain signal of winning sub-task τ_q^* of main task q. Using (3) and (4), we distinguish our proposed approach from those surveyed in [16] by not having any centralized coordination policy.

C. Playing Multi-Stage Stochastic Game

Like [15], the stochastic game is partitioned into several stages of gameplay to focus on the performance of specific tasks. For $\tau \in \Gamma$, attainment of task goal ζ_τ, i.e., $\gamma_\tau = \zeta_\tau$ is equivalent to attaining a stage of the gameplay. We define a stage of the gameplay below.

Definition 5 (Gameplay Stage): A gameplay stage is defined as a segment of the gameplay marked by the need to satisfy specific set of task goals.

An elegant and versatile approach of using multiple stages (MS) is described here. Specifically, we propose to attain Stage e of the gameplay using a set of task goals $\mathbf{G}_e$ where $\mathbf{G}_e = \{\zeta_\tau^e\}$ for $\tau \in \Gamma$ and $e \in \mathcal{E}$ for $\mathcal{E}$ is the set of all stages of the gameplay.

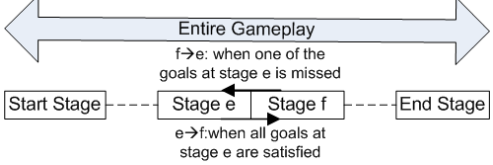

Fig. 3. Forward and Backward Transition of stages during the gameplay

Illustrated in Figure 3, 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 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

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

By segmenting the gameplay into multiple stages, multiple sets of task goals are used to *phase in* the task goals during the gameplay. The goal tasks are used to derive the *pain* signals.

V. The Self-Organizing Neural Network

A self-organizing neural network [23] that derives from FALCON [11] is used for Task τ. Capable of learning incrementally in real time, FALCON is a function approximator that generalizes on the vector patterns without compromising on its prediction accuracy. Action policies are discovered through real-time interactions with the environment using reinforcement learning [26]. The value of applying the action choices on the states is estimated using a temporal difference method known as Q-Learning.

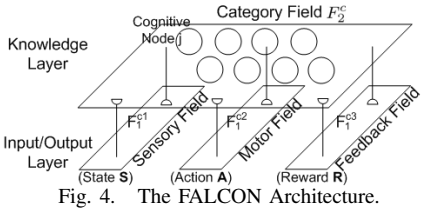

Fig. 4. The FALCON Architecture.

A. Structure and Operating Modes

Structurally, the FALCON network [11] has a two-layer architecture (see Figure 4), comprising of 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 $k = 1, \ldots, 3$.

FALCON has three modes of operation - INSERT, PERFORM and LEARN. The Fusion ART algorithm outlined in Algorithm 1 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. It operates in the *LEARN* mode to learn the effect of these action choices on the states. Though not used in this work, FALCON can operate in the *INSERT* mode to assimilate domain knowledge into itself [12].

Algorithm 1 The Fusion ART algorithm

Require: Activity vectors $\mathbf{x}^{ck}$ and all weights vector $\mathbf{w}_j^{ck}$
1: **for** each F_2^c node j **do**
2: **Code Activation**: Derive choice function T_j^c using

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

where the fuzzy AND operation $(\mathbf{p} \wedge \mathbf{q})_i \equiv min(p_i, q_i)$, the norm $\|.\|$ is defined by $|\mathbf{p}| \equiv \sum_i p_i$ for vectors $\mathbf{p}$ and $\mathbf{q}$, $\alpha^{ck} \in [0, 1]$ is the choice parameters, $\gamma^{ck} \in [0, 1]$ is the contribution parameters and $k = 1, \ldots, 3$
3: **end for**
4: **repeat**
5: **Code Competition**: Index of winning cognitive node J is found using

$$J = \arg\max_j \{T_j^c : \text{for all } F_2^c \text{ node } j\}$$

6: **Template Matching**: Check whether the match functions m_J^{ck} of cognitive node J meet the vigilance criterion

$$m_J^{ck} = \frac{\|\mathbf{x}^{ck} \wedge \mathbf{w}_J^{ck}\|}{|\mathbf{x}^{ck}|} \geq \rho^{ck}$$

where $\rho^{ck} \in [0, 1]$ for $k = 1, \ldots, 3$ are the vigilance parameters
7: **if** vigilance criterion is satisfied **then**
8: *Resonance State* is attained
9: **else**
10: **Match Tracking**: Modify state vigilance ρ^{c1} using

$$\rho^{c1} = \min\{m_J^{ck} + \psi, 1.0\}$$

where ψ is a very small step increment to match function m_J^{ck}
11: **Reset**: $m_J^{ck} = 0.0$
12: **end if**
13: **until** *Resonance State* is attained
14: **if** operating in LEARN/INSERT mode **then**
15: **Template Learning**: modify weight vector $\mathbf{w}_J^{ck}$ using

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

where $\beta^{ck} \in [0, 1]$ is the learning rate
16: **else if** operating in PERFORM mode **then**
17: **Activity Readout**: Read out the action vector $\mathbf{A}$ of cognitive node J using

$$\mathbf{x}^{c2(new)} = \mathbf{x}^{c2(old)} \wedge \mathbf{w}_J^{c2}$$

Decode $\mathbf{x}^{c2(new)}$ to derive recommended action choice a
18: **end if**

B. Incorporating Temporal Difference Method

Using Algorithm 2, 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 Algorithm 1 [26]. 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 the Bounded Q-Learning [26]. It estimates the value of applying action choice a to state s iteratively. The updated Q-value function $Q^{(new)}(s, a)$ is estimated using

$$Q^{(new)}(s, a) = Q(s, a) + \alpha TD_{err}(1 - Q(s, a)), \quad (5)$$

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

Algorithm 2 TD-FALCON algorithm

1: Initialize FALCON
2: Sense the environment and formulate a state representation s
3: Choose to explore at a probability of ϵ
4: **if** Exploration **then**
5: Use *Exploration Strategy* [13] to select an action choice a
6: **else if** Exploitation **then**
7: Use *Direct Code Access* [14] to select an action choice from existing knowledge
8: **end if**
9: Use action choice a on state s for state s'
10: Evaluate effect of action choice a to derive a reward r from the environment
11: Estimate the Q-value function $Q(s, a)$ following a temporal difference formula given by $\Delta Q(s, a) = \alpha TD_{err}$
12: Present $\mathbf{S}$, $\mathbf{A}$ and $\mathbf{R}$ for **Learning**
13: Update the current state $s = s'$
14: Repeat from Step 2 until s is a terminal state

C. Knowledge Pruning

Ineffective learned knowledge is pruned to facilitate more efficient operation. A confidence-based pruning strategy similar to [11] 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.

VI. The Simulation Platform

The stochastic game is illustrated using a commercial PC-based game known as Starcraft Broodwar (SCBW). Released by Blizzard Entertainment Inc. in year 1998, the SCBW game continues to entertain and challenge human gamers. Since year 2009, competitions among these AI-driven players are organized annually at conferences such as CIG and AIIDE. Backed by a rich body of knowledge and a very rich gameplay, numerous works [4][5][6][15][7] were known using the SCBW. Like these works, we illustrate our proposed coordination strategy for MARL by implementing a player of the SCBW game.

There are the macro and the micro gameplay. 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.

The game is won by eliminating the opponents. There is a selection of three races - terran, zerg or protoss - for the

Fig. 5. A snapshot of the SCBW gameplay.

player. The race can either be chosen or randomly assigned to the players. Illustrated in Figure 5, the players of the SCBW game reported in this work are controlled using the built-in AI, an implementation of our proposed approach and other benchmark approaches.

VII. Performance Evaluation

Experiments were conducted with the aim of identifying an effective approach to coordinate MARL. We denote our Pain-based Coordination (PC) Strategy in multi-stage (MS) stochastic game using PC-MS. An approach (PC-SS) of using the PC strategy in single-stage (SS) stochastic game, a Random Response (RR) approach, an uncoordinated MARL (UC) approach and an approach (EK) similar to [10] are included for comparison.

The experiments based on these approaches share a number of common settings. Reinforcement learning is conducted using an updated version of FALCON [23] with the parameters presented in Table I. The experiments were conducted for 100 training iterations. Each training iteration lasts for $10,000$ frames. A total of 200 decisions are made at 50 frames interval. The experimental results are averaged using 20 runs of the same experiment. In addition, these experimental results are further averaged using a sliding window of 10 training iterations to produce the plots.

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 $(c_j(0), \zeta, \eta)$	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

An Asset Scoring Methodology (ASM) is proposed to provide an aggregated view of the goal attainment status of the tasks. Given that the tasks are organized hierarchically, the asset scores are first derived for the sub-tasks. The asset scores of the sub-tasks are then aggregated to give an asset score of the main task.

An asset score μ_τ based on the pain signal p_τ described in Section IV-A is derived for sub-task τ using

$$\mu_\tau = 1.0 - \min\left\{\frac{|\zeta_\tau - \gamma_\tau|}{\zeta_\tau}, 1.0\right\} \tag{6}$$

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

For when $\gamma_\tau < 2\zeta_\tau$, the asset score μ_τ is guaranteed to be $0.0 \leq \mu_\tau \leq 1.0$. The asset score μ_τ saturates at 0.0 when $\gamma_\tau \geq 2\zeta_\tau$. Further differentiation of the performance of task τ is not necessary when $\gamma_\tau \geq 2\zeta_\tau$.

Each main task q has sub-task τ_i where $i \in \{1, \ldots, |\Gamma_q|\}$ and Γ_q is the set of sub-tasks for main task q. Therefore, from (6), the asset score μ_q of main task q is derived using

$$\mu_q = \frac{1}{\sum_i^{|\Gamma_q|} \omega_{\tau_i}} \sum_i^{|\Gamma_q|} \omega_{\tau_i} \mu_{\tau_i} \tag{7}$$

where ω_τ is an empirical value that indicates the relative significance of sub-task τ_1 over sub-task τ_2 when $\{\tau_1, \tau_2\} \in \Gamma_q$ and Γ_q is the set of sub-tasks for main task q.
From (7), 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 sφ is the main task performance indicator.

B. The Results

The performance of the coordination strategies is illustrated using the Asset Scores, Active Stages, Node Population and Decision-Making Time. The weights of the tasks and task goals from three stages - *opening*, *post-opening* and *mid-game* - are presented in Table II. The task goals of the PC-MS approach are based on the active stage of the gameplay while those of the other approaches are based on a single stage gameplay approach.

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	$12, 12, 16$	0.5
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	$1, 1, 1$	0.5
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

Task Performance: From Figure 6 Top Plot, the asset scores illustrate the effectiveness of the different approaches. Similar outcome is observed for all approaches up to the 40^{th} decision. The PC-SS approach has the highest asset score up to the 140^{th} decision where it is overtaken by the EK approach. Our proposed PC-MS approach outperforms the PC-SS approach at around the 170^{th} decision and the EK

approach at around the 180^{th} decision. In comparison, the UC approach is observed with similar performance to the RR approach.

From Figure 6 Bottom Plot, the PC-MS approach is observed forward transitioning to more advanced stage at around the 100^{th}. According Figure 6 Top Plot, this is also where the asset scores of the PC-MS approach begin to improve at increasing rate. In contrast, all the other approaches begin with the targeted stage from the onset. However, the asset scores of these approaches are observed lower than the PC-MS approach after 200 decisions. Only the EK approach is observed with asset scores close to the PC-MS approach after 200 decisions.

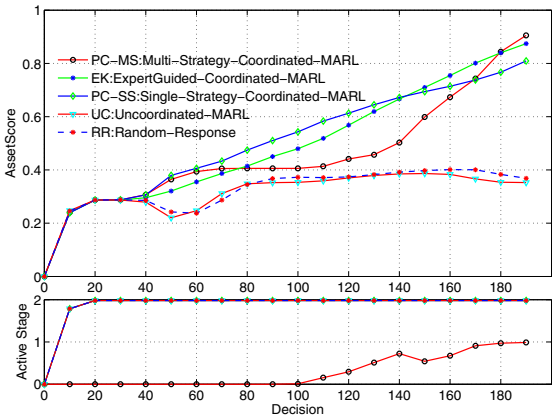

Fig. 6. Comparison of the aggregated asset scores and active stages.

Space Complexity: In the MARL framework, a FALCON with specific state and action field configuration is used for each task. From Figure 1, this means 16 FALCONs are used for the 16 tasks performed during the gameplay. Plots of the node population in Figure 7 Top Plot is an aggregated view of the number of cognitive nodes of the 16 FALCONs. Zero node population is observed for the RR approach because it does not use any FALCON for the tasks. Therefore, with the exception of the RR approach, the pruning technique presented in Section V-C is used in all FALCON-based approaches.

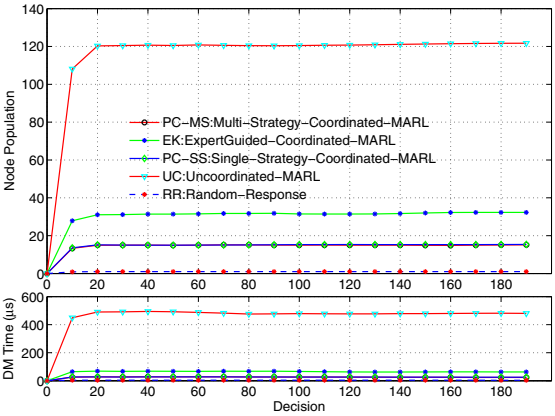

Fig. 7. Comparison of the node population and decision-making time.

From Figure 7 Top Plot, the PC-MS and the PC-SS approaches are observed with similar number of cognitive nodes because both approaches perform the same set of tasks by coordinating MARL using the same technique. In contrast, the PC-MS approach uses multiple stages to achieve these final goals while the PC-SS approach directly aim for these final goals from the onset. Therefore, such observations imply the use of multiple stages have no impact on the node population. However, significant difference is observed in the asset scores of these two configurations in Figure 6 Top Plot.

Also from Figure 7, the EK approach has larger node population than the PC-MS and the PC-SS approaches. Such observations indicate the use of expert knowledge to coordinate MARL is less efficient than our proposed coordination strategy. In addition, the UC approach is observed with the largest node population while the RR approach has zero node population.

Time Complexity: Plots of decision-making (DM) time in Figure 7 Bottom Plot capture the dynamic aspect, i.e., the activities of the implemented models while being correlated to the node population. The DM time is the average amount of time taken to select action choices for the tasks. The tasks in the PC-MS and PS-SS approaches are coordinated using the proposed ML-based technique, expert knowledge is used in the EK approach to coordinate MARL while the UC and RR approaches execute the tasks in sequential order.

From Figure 7 Bottom Plot, the RR approach is observed with the lowest decision-making time because no FALCON is used. In contrast, the UC approach has an average decision-making time of around $450\mu s$. The DM time of the PC-MS and PC-SS approaches fluctuates between $23\mu s$ and $28\mu s$. This is significantly less than DM time of the EK approach fluctuates between $62\mu s$ and $68\mu s$.

Self-Organizing Property: In contrast to reactive planning [5], the order of execution of the tasks are determined in real time. Task dependencies with other tasks and the competition for resources are overcome using the proposed coordination strategy seen in Section IV. Using this approach, tasks become *permissible* for execution when their pre-requisites are satisfied by other tasks. Priority to the shared resources is allocated to the *permissible* tasks with the less satisfied goals. Over time, the goals of the tasks are satisfied in a self-organizing manner.

The efficacy of such an approach can be observed from Figure 8 Top Plot where the number of combat units produced is illustrated. Five agents perform five tasks for producing each type of unit. By just specifying the task goals as seen in Table II, the PC-MS approach is seen close to satisfying the task goals of the Unit Production task at the Post-Opening stage (active stage 1).

VIII. Conclusion

We propose novel coordination strategy to coordinate multi-agent reinforcement learning (MARL) in stochastic game with issues of *resource competition* and *task dependency*. Inspired by [24], the proposed solution integrates the *pain*-based motivated learning (ML) approach with MARL to address these issues. Each instance of the SCBW game is segmented into several stages of gameplay. Each stage is marked by the need to satisfy specific set of task goals. The *pain* signals are derived with respect to the task goals of

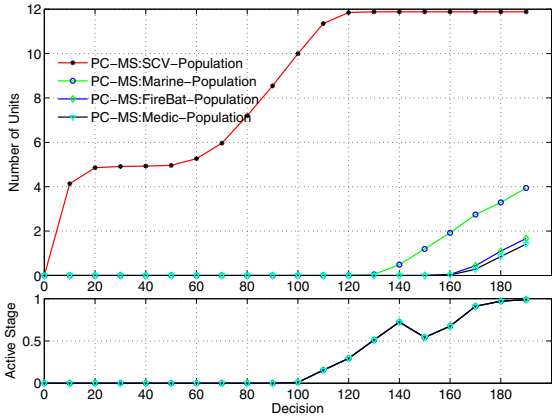

Fig. 8. Self-organizing production of units in correlation to the active stages

the active stage. Each time, the resources are used by the winning task to address its goal. Over time, the task goals are satisfied in a coordinated and self-organizing manner.

We compare our proposed coordination strategy for MARL with four other approaches. The Random Response approach is included as a baseline. An uncoordinated approach is included to highlight the differences with the coordinated approaches. A ML-based coordinated MARL approach used with a single stage gameplay is included for comparison with our proposed coordinated MARL approach of using multiple stages. Considering expert-guided coordination [10] as an accurate approximation of the ground truth, we had included such an approach for comparison. The performance of these approaches is illustrated using asset scores, node population and decision-making time. The resultant self-organizing property is also illustrated using the unit production task. From the experiments conducted using Starcraft Broodwar (SCBW), the results show the Multi-Stage approach has the highest asset score while having similar node population as the other approaches. Expectedly, the uncoordinated approach has the least optimal outcome.

The task goals for the gameplay stages used in this work are pre-determined. During the gameplay, these task goals are used without any adaptation to coordinate MARL. This approach is rigid and is likely to lead to sub-optimal outcome for the tasks at each stage of the gameplay. Currently, only experienced human gamers can adapt the task goals optimally in real time. Therefore, a computational model with similar capability is seen as natural progression of this work. In addition, we are working towards an integrated solution capable of playing and winning the full game against the built-in AI and the human gamers.

REFERENCES

[1] 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.

[2] S. Kapetanakis and D. Kudenko, "Reinforcement learning of coordination in cooperative multi-agent systems," in Proceedings of the *National Conference on Artificial Intelligence*, 2002, pp. 326-331.

[3] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in Proceedings of the *International Conference on Machine Learning*, 1994, pp. 157-163.

[4] N. Othman, J. Decraene, W. Cai, N. Hu, Y.-H. Low and A. Gouaillard, "Simulation-based optimization of StarCraft tactical AI through evolutionary computation," in Proceedings of the *IEEE Conference on Computational Intelligence and Games*, 2012, pp. 394-401.

[5] B. G. Weber, M. Mateas, 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.

[6] S. Wender and I. Watson, "Applying reinforcement learning to small scale combat in the real-time strategy game StarCraft: Broodwar," in Proceedings of the *IEEE Conference on Computational Intelligence and Games*, 2012, pp. 402-408.

[7] J. Young, F. Smith, C. Atkinson, K. Poyner and T. Chothia, "SCAIL: An integrated Starcraft AI system," in Proceedings of the *IEEE Conference on Computational Intelligence and Games*, 2012, pp. 438-445.

[8] F. S. Melo and M. Veloso, "Learning of coordination: exploiting sparse interactions in multiagent systems," in Proceedings of the *International Conference on Autonomous Agents and Multiagent Systems*, 2009, pp. 773-780.

[9] M. Dastani, F. Arbab and F. de Boer, "Coordination and composition in multi-agent systems," in Proceedings of the *International Conference on Autonomous Agents and Multiagent Systems*, 2005, pp. 439-446.

[10] Q. Lau, M.-L. Lee and W. Hsu, "Coordination guided reinforcement learning," in Proceedings of the *International Conference on Autonomous Agents and Multiagent Systems*, 2012, pp. 215-222.

[11] A.-H. Tan, "FALCON: A Fusion Architecture for Learning, COgnition, and Navigation," in Proceedings of the *International Joint Conference on Neural Networks*, 2004, pp. 3297-3302.

[12] T.-H. Teng, Z.-M. Tan and A.-H. Tan, "Self-organizing neural models integrating rules and reinforcement learning," in Proceedings of the *International Joint Conference on Neural Networks*, 2008, pp. 3770-3777.

[13] T.-H. Teng and A.-H. Tan, "Knowledge-based exploration for reinforcement learning in self-organizing neural networks," in Proceedings of the *IEEE/WIC/ACM International Conference on Intelligence Agent Technology*, 2012, pp. 332-339.

[14] A.-H. Tan, "Direct code access in self-organizing neural networks for reinforcement learning," in Proceedings of the *International Joint Conference on Artificial Intelligence*, 2007, pp. 1071-1076.

[15] S. Yi, Adaptive strategy decision mechanism for StarCraft AI, in Proceedings of the *EU-Korea Conference on Science and Technology*, 2010, pp. 47-57.

[16] L. Busoniu, R. Babuska and B. de Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156-172, 2008.

[17] M. Bowling, "Convergence and no-regret in multiagent learning," *Advances in Neural Information Processing Systems*, vol. 17, pp. 209-216, 2005.

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

[19] M. L. Littman, "Value-function reinforcement learning in markov games," *Cognitive Systems Research*, vol. 2, no. 1, pp. 55-66, 2001.

[20] O. Abul, F. Polat and R. Alhajj, "Multiagent reinforcement learning using function approximation," *IEEE Trans. Syst. MAn Cybern. C, Appl Rev.*, vol. 30, no. 4, pp. 485-497, 2000.

[21] 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.

[22] H. Tamakoshi and S, Ishii, "Multiagent reinforcement learning applied to a chase problem in a continuous world," *Artificial Life and Robotics*, vol. 5, no. 4, pp. 202-206, 2001.

[23] 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.

[24] 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.

[25] G. Chen, Z. Yang, H. He, K.-M. Goh, "Coordinating Multiple Agents via Reinforcement Learning," *Autonomous Agents and Multi-Agent Systems*, vol. 10, no. 3, pp. 273-328, 2005.

[26] A.-H. Tan, N. Lu and X. Dan, "Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evaluative feedback," *IEEE Trans. Neural Netw.*, vol. 19, no. 2, pp. 230-244, Feb, 2008.

[27] Y. Shoham and K. Leyton-Brown, "Multiagent systems: algorithmic, game-theoretic and logical foundations," Cambridge University Press, 2009.

[28] J. A. Starzyk, "Motivation in embodied intelligence," in *Frontiers in Robotics, Automation and Control*, I-Tech Education and Publishing, 2008, pp. 83-110.