

Cooperative Cognitive Agents and Reinforcement Learning in Pursuit Game

D. Xiao and A. H. Tan

School of Computer Engineering, Nanyang Technological University
Nanyang Avenue, Singapore 639798
{XIAO0002,asahtan}@ntu.edu.sg

Abstract

This paper illustrates how a self-organizing cognitive architecture, known as TD-FALCON, can learn to function and cooperate in a dynamic environment. TD-FALCON learns the value functions of the state-action space estimated through a temporal difference (TD) method. The learned value functions are then used to determine the optimal actions based on an action selection policy. To tackle a multi-agent predator/prey pursuit task, we develop a cooperative strategy using a high-level compressed state representation and a hybrid reward function. Experiments show that TD-FALCON agent teams operating with the proposed cooperative strategy produce superior performance with a high level of efficiency and scalability.

1 Introduction

A key characteristic of autonomous agents is the ability to function and adapt by themselves through reinforcement learning in a complex and dynamic environment [1]. Self-organizing neural models, known as FALCON (Fusion Architecture for Learning, COgnition, and Navigation) [2, 3], adapt by learning multiple mappings simultaneously across multi-modal input patterns, involving states, actions, and rewards, in an online and incremental manner. A specific class of FALCON models, called TD-FALCON, learns the value functions of the state-action space estimated through temporal difference (TD) algorithms. The learned value functions are then used to determine the optimal actions based on an action selection policy.

Autonomous agents seldom work in isolation. To operate in a multi-agent environment, an agent faces many new challenges. First of all, the agent is affected by the actions of the other agents and in turns its action also affects the environment and the other agents. Secondly, as agents continue to learn, their behaviours

change over time and the environment becomes highly dynamic. An agent must therefore be able to predict the actions or to model the reasoning processes of the others in some way.

Our previous work based on a minefield pursuit task shows that TD-FALCON systems are able to perform in a multi-agent setting without explicit cooperative mechanism [3]. In this paper, we investigate a more challenging predator/pursuit task which demands an explicit mechanism of cooperation [4]. To this end, we develop a cooperative strategy using a high-level state representation incorporating shared cooperative signals and a hybrid reward function combining individual and team payoffs. Our experiments show that appropriate state representation plays a critical role in producing superior performance while maintaining efficiency and scalability. In addition, individual agents need not have in-depth knowledge of the other agents' underlying models to accomplish the task successfully.

The rest of the paper is organized as follows. Section 2 provides a summary of the FALCON architecture and the associated value function learning and prediction algorithms. Section 3 presents the TD-FALCON algorithms, consisting of the action selection policy and the value function estimation mechanism. Section 4 describes the predator/prey pursuit task. Section 5 motivates and presents the proposed mechanism of cooperation. Section 6 reports our experimental results. The final section concludes and highlights our future effort.

2 FALCON ARCHITECTURE

FALCON is an extension of predictive Adaptive Resonance Theory (ART) networks for learning multi-modal pattern mappings across multiple input channels. For reinforcement learning, FALCON makes use of a 3-channel architecture, consisting of a sensory field F_1^{c1} for representing the current state, an action

field F_1^{c2} for representing the available actions, a reward field F_1^{c3} for representing the values of the feedback received from the environment, and a cognitive field F_2^c for encoding the relations among the values in the three input channels (Figure 1). We describe how FALCON can be used to predict and learn value functions for reinforcement learning below.

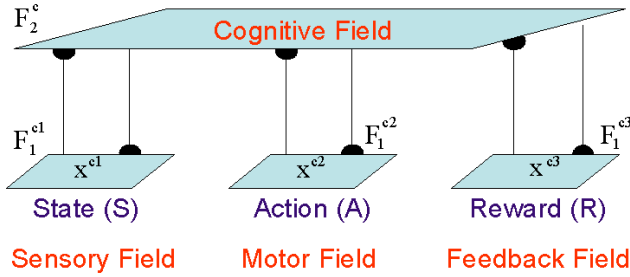


Figure 1: The FALCON architecture.

Input vectors: Let $\mathbf{S} = (s_1, s_2, \dots, s_n)$ denote the state vector, where s_i indicates the sensory input i . Let $\mathbf{A} = (a_1, a_2, \dots, a_m)$ denote the action vector, where a_i indicates a possible action i . Let $\mathbf{R} = (r, \bar{r})$ denote the reward vector, where $r \in [0, 1]$ and $\bar{r} = 1 - r$. Complement coding has been found effective in ART systems in preventing code proliferation.

Activity vectors: Let \mathbf{x}^{ck} denote the F_1^{ck} activity vector. Let \mathbf{y}^c denote the F_2^c activity vector.

Weight vectors: Let \mathbf{w}_j^{ck} denote the weight vector associated with the j th node in F_2^c for learning the input patterns in F_1^{ck} . Initially, all F_2^c nodes are uncommitted and the weight vectors contain all 1's.

Parameters: The FALCON's dynamics is determined by choice parameters $\alpha^{ck} > 0$ for $k = 1$ to 3; learning rate parameters $\beta^{ck} \in [0, 1]$ for $k = 1$ to 3; contribution parameters $\gamma^{ck} \in [0, 1]$ for $k = 1$ to 3 where $\sum_{k=1}^3 \gamma^{ck} = 1$; and vigilance parameters $\rho^{ck} \in [0, 1]$ for $k = 1$ to 3.

The dynamics of FALCON in learning and predicting, based on fuzzy ART operations [5], is described in the following sections.

2.1 Predicting Value Functions

Under the predicting mode, FALCON receives input values in one or more fields and predicts the values for the remaining fields. Upon input presentation, the input fields receiving values are initialized to their respective input vectors. Input fields not receiving values are initialized to \mathbf{N} , where $N_i = 1$ for all i . For predicting value functions, the state and action vec-

tors are presented to FALCON. Therefore, $\mathbf{x}^{c1} = \mathbf{S}$, $\mathbf{x}^{c2} = \mathbf{A}$, and $\mathbf{x}^{c3} = \mathbf{N}$.

The predicting process of FALCON consists of three key steps, namely code activation, code competition, and activity readout, described as follows.

Code activation: A bottom-up propagation process first takes place in which the activities (known as choice function values) of the cognitives (known in the F_2^c field) are computed. Given the activity vectors $\mathbf{x}^{c1}, \dots, \mathbf{x}^{c3}$, the choice function T_j^c of each F_2^c node j is computed as follows:

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

where the fuzzy AND operation \wedge is defined by $(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i)$, and the norm $|\cdot|$ is defined by $|\mathbf{p}| \equiv \sum_i p_i$ for vectors \mathbf{p} and \mathbf{q} .

Code competition: A code competition process follows under which the F_2^c node with the highest choice function value is identified. The system is said to make a choice when at most one F_2^c node can become active after code competition. The winner is indexed at J where

$$T_J^c = \max\{T_j^c : \text{for all } F_2^c \text{ node } j\}. \quad (2)$$

When a category choice is made at node J , $y_J^c = 1$; and $y_j^c = 0$ for all $j \neq J$. This indicates a winner-take-all strategy.

Activity readout: The chosen F_2^c node J performs a readout of its weight vectors to the input fields F_1^{ck} such that

$$\mathbf{x}^{ck(\text{new})} = \mathbf{x}^{ck(\text{old})} \wedge \mathbf{w}_J^{ck}. \quad (3)$$

Finally, the reward vector \mathbf{R} associated with the input state and action vectors \mathbf{S} and \mathbf{A} is given by $\mathbf{R} = \mathbf{x}^{c3}$.

2.2 Learning Value Functions

For learning value functions, the state, action, and reward vectors are presented to FALCON. Therefore, $\mathbf{x}^{c1} = \mathbf{S}$, $\mathbf{x}^{c2} = \mathbf{A}$, and $\mathbf{x}^{c3} = \mathbf{R}$. Under the learning mode, FALCON performs code activation and code competition (as described in the previous section) to select a winner J based on the activity vectors $\mathbf{x}^{c1}, \mathbf{x}^{c2}$, and \mathbf{x}^{c3} . To complete the learning process, template matching and template learning are performed as described below.

Template matching: Before code J can be used for learning, a template matching process checks that the weight templates of code J are sufficiently close to

their respective input patterns. Specifically, a resonance occurs if for each channel k , the *match function* m_J^{ck} of the chosen code J meets its vigilance criterion:

$$m_J^{ck} = \frac{|\mathbf{x}^{ck} \wedge \mathbf{w}_J^{ck}|}{|\mathbf{x}^{ck}|} \geq \rho^{ck}. \quad (4)$$

When a resonance occurs, Learning then ensues, as defined below. If any of the vigilance constraints is violated, mismatch reset occurs in which the value of the choice function T_J^c is reset to 0 for the duration of the input presentation. The search process then repeats to select another F_2^c node J until a resonance is achieved.

Template learning: Once a node J is selected for firing, for each channel k , the weight vector \mathbf{w}_J^{ck} is modified by the following learning rule:

$$\mathbf{w}_J^{ck(\text{new})} = (1 - \beta^{ck})\mathbf{w}_J^{ck(\text{old})} + \beta^{ck}(\mathbf{x}^{ck} \wedge \mathbf{w}_J^{ck(\text{old})}). \quad (5)$$

For an uncommitted node J , the learning rates β^{ck} are typically set to 1. For committed nodes, β^{ck} can remain as 1 for fast learning or below 1 for slow learning in a noisy environment.

3 TD-FALCON

The general sense-act-learn algorithm of TD-FALCON is summarized in Table 1. Given the current state s and a set of available actions \mathcal{A} , the FALCON network is used to predict the value of performing each available action. The value functions are then processed by an action selection strategy (also known as policy) to select an action. Upon receiving a feedback (if any) from the environment after performing the action, a TD formula is used to estimate the value of the next state. The value is then used as the teaching signal for FALCON to learn the association of the current state and the chosen action to the estimated value.

3.1 Action Selection Policy

To achieve a balance between exploration and exploitation, the e-greedy policy selects the action with the highest value with a probability of $1 - e$, where e is a constant between 0 and 1, and takes a random action, with probability e [6].

With a fixed e value, the agent will always explore the environment with a constant level of randomness. In practice, it is beneficial to have a higher e value in the initial stage to encourage exploration of paths and a lower e value in the later stage to optimize the

-
1. Initialize the FALCON network.
 2. Given the current state s , for each available action a in the action set \mathcal{A} , predict the value of the action $Q(s, a)$ by presenting the corresponding state and action vectors \mathbf{S} and \mathbf{A} to FALCON.
 3. Based on the value functions computed, select an action a from \mathcal{A} following an action selection policy.
 4. Perform the action a , observe the next state s' , and receive a reward r (if any) from the environment.
 5. Estimate the value function $Q(s, a)$ following a TD formula given by $\Delta Q(s, a) = \alpha TD_{err}$.
 6. Present the corresponding state, action, and reward (Q-value) vectors (\mathbf{S} , \mathbf{A} , and \mathbf{R}) to FALCON for learning.
 7. Update the current state by $s=s'$.
 8. Repeat from Step 2 until s is a terminal state.
-

Table 1: Generic dynamics of TD-FALCON.

performance by exploiting familiar paths. A decayed e-greedy policy is thus proposed to gradually reduce the value of e over time. The rate of decay d_e is typically inversely proportional to the complexity of the environment as problems with a larger input and action space will take a longer time to explore.

3.2 Value Function Estimation

One key component of the TD-FALCON (Step 5) is the iterative estimation of value functions $Q(s,a)$ using a temporal difference equation

$$\Delta Q(s, a) = \alpha TD_{err} \quad (6)$$

where $\alpha \in [0, 1]$ is the learning parameter and TD_{err} is a function of the current Q-value predicted by FALCON and the Q-value newly computed by the TD formula.

Using the Q-learning rule, the temporal error term is computed by

$$TD_{err} = r + \gamma \max_{a'} Q(s', a') - Q(s, a) \quad (7)$$

where r is the immediate reward value, $\gamma \in [0, 1]$ is the discount parameter, and $\max_{a'} Q(s', a')$ denotes the maximum estimated value of the next state s' . The update rule is applied to all states that the agent traverses. With value iteration, the value function $Q(s, a)$ is expected to converge to $r + \gamma \max_{a'} Q(s', a')$ over time. To ensure that all input values are bounded between 0 and 1, a scaling term is incorporated to form the bounded Q-Learning rule given by

$$\Delta Q(s, a) = \alpha TD_{err} (1 - Q(s, a)). \quad (8)$$

4 The Predator/Prey Pursuit Task

Our pursuit game field is composed of 16×16 squares, with one prey and four predators. The prey typically starts at the center of the field and the predators are distributed on the four edges of the field. This is a *typical layout*. In more complex cases, predators may be distributed anywhere in the game field. The layout is named as *random layout* (Figure 2). The rules of the game are summarized as follows.

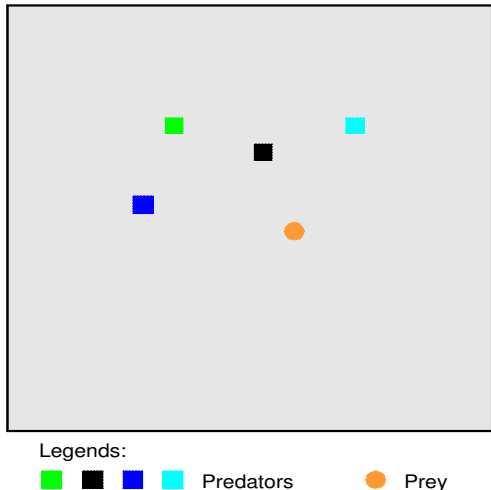


Figure 2: A random layout of the pursuit game.

1. At each time step, a predator can select to move in one of the four directions, namely move up, move right, move down, or move left at the pace of one square, or remain stationary. No predator can go out of the game field. Two predators cannot occupy the same square.
2. The prey also selects to move in one of the four directions but only does so at alternate time steps. In general, it will select the shortest path to one of the four boundaries, unless there is a predator in its way. If all the four direct paths to the boundaries are blocked, the prey will take a random action.
3. At each game trial, the predators will attempt to encircle the prey, making it not movable. When the prey is completely surrounded by the four predators, the trial is ended successfully.
4. When the prey reaches any side of the game field, the trial is considered as a failure. When the trial is not completed within 100 steps, it is also deemed as a failure.

5 Mechanism of Cooperation

5.1 The State Representation

To enable cooperation, an agent needs to be aware of its surrounding agents. Using monolithic Q-learning, the state space of each agent is augmented with information of the other agents [7, 8, 9]. The problem is that the dimensionality of the state space for each agent grows exponentially with respect to the number of agents, leading to the problem of combinatorial explosion [10].

Consider a world with n agents, the state of an agent i can be represented by an n -tuple (c_1, c_2, \dots, c_n) , where c_i represents the coordinates of the prey, relative to the agent and $c_j (j \neq i)$ indicates the relative positions of the other agent j with respect to agent i . For each other agent (or prey), the number of possible relative positions is given by $(2d+1)^2$, where d is the visual depth of agent i . In our game field, the maximum visual depth is 15. Therefore the number of possible states that a predator interacts with another predator (or the prey) will be $N = (2d+1)^2 = (2 \times 15 + 1)^2 = 961$. Since there are a total of four sets of coordinates to track, the size of the state space for a single predator balloons to $S = N^4 = 961^4 = 852,891,037,441$.

Based on our analysis of the problem, the optimal action of a predator depends on its bearing rather than its distance to the prey. This implies that the canonical distance between the predators and the prey may not be strictly required. Suppose the bearing from the predator i to the prey is b^i where $0 \leq b^i \leq 7$, the state vector of a predator i is given by $(\mathbf{B}^i, \mathbf{B}^o)$, where the prey bearing vector \mathbf{B}^i is computed by

$$B_j^i = \begin{cases} 1 & \text{if } j = b^i \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

and the other-bearing vector \mathbf{B}^o is the concatenation of the bearing vectors from the other predators to the prey. Considering bearing values for just 8 directions, the combined state vector translates to a state space of 8^4 .

Instead of representing the bearing of each agent, we introduce a high level concept called *Center of Agent Team* (CAT) and incorporate into the state vector the bearing from the CAT to the prey. Whereas a typical predator bearing vector consists of eight possible values, the CAT bearing vector has an extra value denoting the situation wherein CAT coincides with the prey. Suppose the bearing from the CAT to the prey is b^c (where $0 \leq b^c \leq 8$), the state vector is given by

$(\mathbf{B}^i, \mathbf{B}^c)$, where the prey bearing vector \mathbf{B}^i is computed as before and the CAT bearing vector \mathbf{B}^c is given by

$$B_j^c = \begin{cases} 1 & \text{if } j = b^c \\ 0 & \text{otherwise.} \end{cases} \quad (10)$$

In total, the size of the state space for each predator is now $8 \times 9 = 72$, much less than the baseline solution based on relative positions.

5.2 The Reward Scheme

In a canonical TD-FALCON algorithm, an agent is not aware of how its movement contributes to the entire team and the outcome of the game. Previously, profit sharing [11] has been used by averaging the rewards received by all the agents. However, this approach fails to consider the specific situation of each individual agent. In a successful trial, the prey is eventually immovable after being surrounded by the four predators. When this happens, the distance between each predator and the prey is the minimum. In addition, the *Center of Agent Team* (CAT) coincides with the prey. When a predator is pursuing the prey, the distance between them is decreasing and the CAT is also approaching the prey. Therefore, a predator makes a contribution to the overall task by a move that reduces its distance to the prey and at the same time, gets the CAT closer to the prey. To incorporate both the individual and team payoffs, the reward function of a predator i is defined as $r^i = \frac{1}{(1+d^i)(1+d^c)}$, where d^i and d^c are the distance from the predator and the CAT to the prey respectively.

6 Experimental Results

We conducted three sets of comparative experiments to evaluate the efficacy of the proposed cooperative strategy. The first set of the experiments, involving teams of non-cooperating agents, provided the baseline performance for comparison. The second set of experiments involved TD-FALCON teams using an all-bearing state representation. The last set of experiments employed TD-FALCON teams using the CAT bearing strategy. The sensory state representation and the reward functions of the various experiments are summarized in Table 2. All experiments were based on the random game field layout. All TD-FALCON agents used the standard parameter values: choice parameter $\alpha^{c1} = \alpha^{c2} = \alpha^{c3} = 0.001$, learning rate $\beta^{c1} = \beta^{c2} = \beta^{c3} = 1.0$, vigilance $\rho^{c1} = \rho^{c2} = \rho^{c3} = 0.9$, initial exploration rate $e = 0.6$, decay rate $d_e = 0.0004$,

Q-learning rule learning rate $\alpha = 0.5$, and discount parameter $\gamma = 0.01$.

Strategy	State Representation	Reward (r^i)
Non-cooperating	$\mathbf{D} = (s_0, s_1)$ $\mathbf{B}^i = (b_0, b_1, \dots, b_7)$	$\frac{1}{(1+d^i)}$
All Bearing	$\mathbf{B}^i = (b_0, b_1, \dots, b_7)$ $\mathbf{B}^o = (b_8, b_9, \dots, b_{31})$	$\frac{1}{(1+d^i)(1+d^c)}$
CAT Bearing	$\mathbf{B}^i = (b_0, b_1, \dots, b_7)$ $\mathbf{B}^c = (c_0, c_1, \dots, c_8)$	$\frac{1}{(1+d^i)(1+d^c)}$

Table 2: The state vectors and reward functions used by the various cooperative strategies.

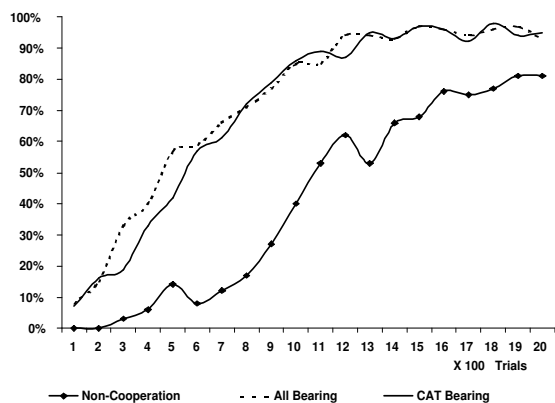


Figure 3: Average success rates for the pursuit task.

Figure 3 shows that cooperating TD-FALCON teams have a clear performance advantage over their counterpart without cooperation. After 1500 trials when the exploration rate dropped to zero, the success rates reached 95%, compared with 80% achieved without cooperation. Figure 4 again illustrates the benefit of cooperation in terms of the average number of steps taken by the agent teams to surround the prey. In spite of slight fluctuations, the trend clearly shows that the cooperating teams are more efficient.

Figure 5 compares the three strategies based on the number of cognitive codes created by each individual TD-FALCON agent. We observe that the use of CAT bearing results in a significantly smaller number of cognitive codes learned. The sharp code reduction is believed to be the result of the removal of the relative position information as well as the bearings of other agents from the state vector. It is notable that while the growth of cognitive codes stays flat all the way for the CAT bearing strategy, those of two other strategies keep trending up till around 1500 trials, when the exploration of new codes stops.

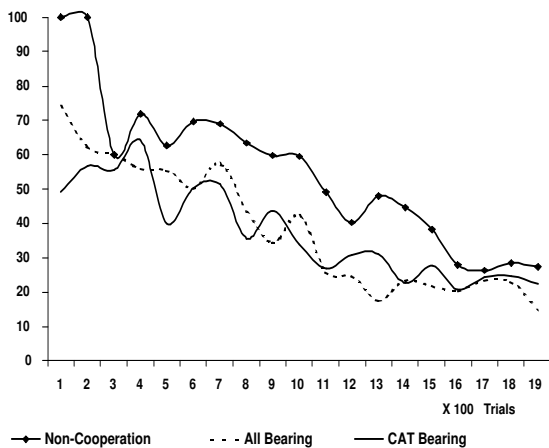


Figure 4: Number of steps taken to capture prey.

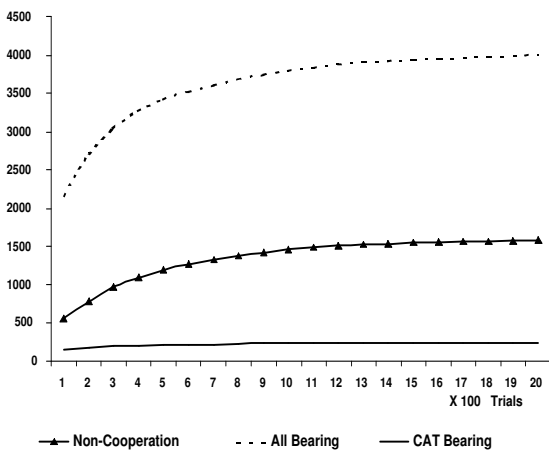


Figure 5: Number of cognitive codes learned.

7 Conclusion

Two critical issues of multi-agent systems are scalability and co-adaptation dynamics. Our work indicates that appropriate coding of state representation can greatly enhance the scalability of the multi-agent systems. Specifically, high level cooperative signals, such as CAT, are effective in producing superior performance with a high level of efficiency and scalability.

By using a self-organizing learning architecture, TD-FALCON offers an efficient solution to adapt and operate in a real-time multi-agent environment in an online and incremental manner. Having obtained very encouraging results, our next step is to conduct a thorough analysis of the cooperative strategy and to perform a quantitative performance comparison with alternative multi-agent learning systems.

References

- [1] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, Cambridge, MA: MIT Press, 1998.
- [2] A. H. Tan, "FALCON: A fusion architecture for learning, cognition, and navigation," in *Proceedings of IJCNN*, 2004, pp. 3297–3302.
- [3] A.-H Tan and D. Xiao, "Self-organizing cognitive agents and reinforcement learning in a multi-agent environment," in *Proceedings, IEEE/ACM/WIC International Conference on Intelligent Agent Technologies*, 2005.
- [4] M Brenda, V. Jagannathan, and R. Dodhiawala, "On optimal cooperation of knowledge sources - an empirical investigation," Tech. Rep., Boeing Advanced Technology Center, 1986, Technical Report BCS-G2010-28.
- [5] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759–771, 1991.
- [6] A. Pérez-Urbe, *Structure-Adaptable Digital Neural Networks*, Ph.D. thesis, Swiss Federal Institute of Technology-Lausanne, 2002.
- [7] M. L. Littman, "Markov games as a framework for multiagent reinforcement learning," in *proceedings of the 11th international conference on Machine learning*, San Francisco, CA, 1994, pp. 157–163.
- [8] T. W. Sandholm and R. H. Crites, "Multiagent reinforcement learning in the iterated prisoner's dilemma," *Biosystems*, vol. 37, pp. 147–166, 1995.
- [9] M. Tan, "Multi-agent reinforcement learning: independent vs. cooperative agents," in *Proceedings of the 10th International Conference on Machine Learning*. 1993, pp. 330–337, Morgan Kaufmann.
- [10] N. Ono and K. Fukumoto, "Multi-agent reinforcement learning: a modular approach," in *Proceedings of the 2nd International Conference on Multi-Agent Systems*, 1996, pp. 252–258.
- [11] S. Arai and K. Sycara, "Effective learning approach for planning and scheduling in multi-agent domain," in *Proceedings of the 6th International Conference on Simulation of Adaptive Behavior (From animals to animats 6)*, 2000, pp. 507–516.