To appear in Proceedings of the 6th ISAB -From animals to animats 6- pp.507-516

Effective Learning Approach for Planning and Scheduling in Multi-Agent Domain

Sachiyo Arai* **Katia Sycara***
*Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213
sachiyo.katia@cs.cmu.edu

Abstract

The point we want to make in this paper is that Profit-sharing; a reinforcement learning approach is very appropriate to realize the adaptive behaviors in a multi-agent environment. We discuss the effectiveness of Profit-sharing theoretically and empirically within a *Pursuit Game* where there exist multiple preys and multiple hunters. In our context of this problem, hunters need to coordinate adaptively one another to capture all the preys, without sharing information, predefined organization and any prior knowledge around their environment. *Pursuit Game* itself is very simple but can be extended to a real problem. Our approach, Profit-sharing, is contrastive to other reinforcement learning approaches which are based on *Dynamic Programming* , such as Temporal Difference method and Q-learning, in that Profit-sharing guarantees convergence to a effective policy even in domains that do not obey the Markov property, if a task is episodic and a credit is assigned in an appropriate manner. Profit-sharing is also different from Q(1) and Sarsa(1) methods in that it does not need eligibility trace to manage the delayed reward.

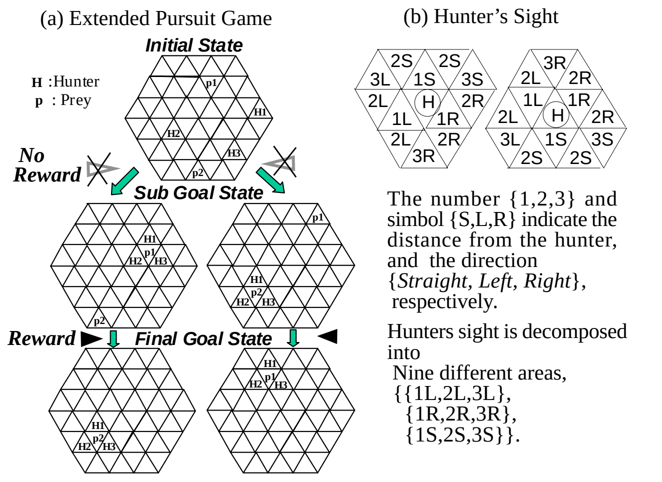
Though our monolithic implementation here seems to be an impractical in a real word, we need to discuss the validity of algorithm as a multi-agent reinforcement learning context before introducing some structured frameworks into the monolithic method to extend its application. The contribution of this paper is that we introduce Profit-sharing as the effective algorithm in the multiagent domain and report its advantages and limitations without hierarchies.

1. Introduction

Many existing approaches have used a symbolic representation to reason about agent interaction within multi-agent planning domains (Georgeff 1983), and within the context of dynamic domains (Firby 1987). These approaches normally adopt a top-down strategy, and hence require an explicit model of the environment and a definition of the communication protocol used for multi-agent cooperation. Although the corresponding agents work successfully in complex, dynamic domains, it can be difficult to design whole parts of the agent's knowledge. As the number of agents within these multi-agent communities rises, it is becoming increasingly difficult to design this knowledge statically.

For dynamic domains (such as the one presented in this paper), it is not unreasonable to design agents that use local *condition-action rules* to react to each world state, as it can be very difficult to model the whole domain. The problem therefore becomes that of determining how these rules should be designed for dynamic environments. In recent years, bottom-up approaches such as reinforcement learning have become increasingly popular for determining these condition-action, or state-action rules, without having a priori models of the environment. Furthermore, the structured reinforcement learning approach has also presented for recent years to extend its performance in a single agent's context. However, there are still several important issues that arise when applying these bottom-up approaches to multi-agent domains, such as a *Pursuit Game*, where the agents may have to face perceptual aliasing and uncertainty of other agents' intentions. Our implementation using Profit-sharing here is monolithic and not introduced any hierarchical structure, which seems to be an impractical in a real word, on purpose to make sure of the validity of this algorithm as a multi-agent reinforcement learning context as is often the case with an animal's society.

Profit-sharing is contrastive to other reinforcement learning approaches which are based on *Dynamic Programming* (DP), such as Temporal Difference method and Q-learning, in that Profit-sharing guarantees convergence to a effective policy even in domains that do not


(a) Extended Pursuit Game
Initial State
H :Hunter
p : Prey
No Reward
Sub Goal State
Reward
Final Goal State
(b) Hunter's Sight
2S 2S 3L 1S 3S 2L H 2R 1L 1R 2L 2R 3R
3R 2L 2R 1L 1R 2L H 2R 3L 1S 3S 2S 2S

The number {1,2,3} and simbol {S,L,R} indicate the distance from the hunter, and the direction {*Straight, Left, Right*}, respectively.

Hunters sight is decomposed into Nine different areas, {{1L,2L,3L}, {1R,2R,3R}, {1S,2S,3S}}.

Figure 1: Pursuit Game of Multiple Preys

obey the Markovian property, if a task is episodic and a credit is assigned in an appropriate manner. The domain including multiple learning entities does not seem to be assumed as the Markov Decision Processes (MDPs). Even though the domain can be assumed as the MDPs, the state transition probabilities of domain appear to be changed from the animal's viewpoint, because its sensory limitation causes the perceptual aliasing problem. Therefore, it seems reasonable to suppose that the adaptive behavior of animals' is realized by the Profit-sharing, which collects stochastic data on a successful action of each state rather than by DP-based algorithms which require Markovian property of the domain. We demonstrate empirically that our Profit-sharing approach is effective within this domain and clarify some of the requirements that face multi-agent reinforcement learning problems.

In Section 2, we describe our *Pursuit Game* definition from the perspective of a reinforcement learning approach, and present our agent model. Section 3 introduces the principles of Profit-sharing, the *Rationality Theorem*, which makes Profit-sharing powerful, and its advantage over other learning algorithms which are usually found within multi-agent domains. An empirical comparison of the performance of multiple agents using other learning approaches; Profit-sharing, Q-learning and Modular-Q approach, is presented via several experiments in Section 4. The acquired rules that result in near-optimal behavior for the agents in *Pursuit Game* domain are also presented. Finally, we discuss the applicability and effectiveness of the Profit-sharing based method for real-world dynamic domains, and summarize our future work.

2. Problem Domain

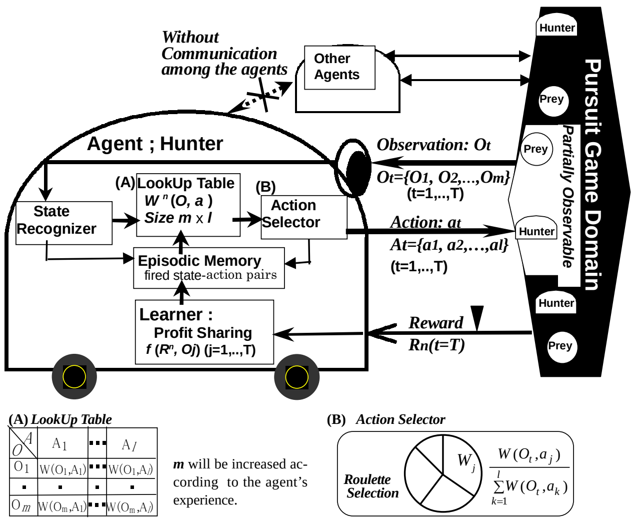
Pursuit Game, originally suggested in (Benda 1985), has been used to test a variety of coordination strategies. There are many researchers which treat this problem with different motivations respectively. The focus in (Gasser 1989),(Stephens 1989) and (Stephens 1990) is on the generally effective organization among hunters. And (Levy 1992) focuses on the coalition problem among hunters from the viewpoint of game theory. Usually, the environment is a grid world in which four hunters and a single prey are placed randomly initially and the goal of the hunters is to surround the prey, which moves randomly.

In this paper, we treat hexagonal-world where there exist multiple preys and three hunters as a multiple-preys domain as shown in Figure1(a). Each hunter is assumed to be a learning agent, whereas the prey does not learn and moves randomly in the environment. The final goal of the hunters' is to capture all the preys in the environment. Because of the well known problem of state explosion in reinforcement learning approaches, we are obliged to create an environment with triangular cells to reduce the size of the state space to observe an emergence of a scheduling strategy among the hunters . Here, three hunters are required to capture each prey. In our settings, each hunter can know the location of a prey(preys) only when the prey(preys) is(are) in the hunter's sight which is defined as shown in Figure1(b). The sight of hunter is decomposed into nine different areas and each area represents its status in terms of {vacancy, existence of the hunter, existence of the prey}(*Note:* other hunters and preys are distinguishable from each other, and they cannot co-exist in the same location at the same time.)

This *Pursuit Game* domain is an example of one that exhibits the following characteristics. First, there are several agents which are all "self-interested"; i.e. they pursue their own goals but can not achieve them without cooperation. Second, the agreement among the agents must be required to purse the common goal. Third, the agents are required to behave rationally rather optimally, in such an uncertain domain. By "rational", we mean that each agent should reach their goals in a finite time period; i.e. the agent should not become trapped within infinite loops in the state machine. Although in a real animal world, the acquired policy need not be optimal, it is important that this policy is rational. Fourth, the domain is both uncertain and dynamic as a real animal world. Because these characteristics,it is very difficult to design rules through mathematical analysis, as the information required by each agent is not only distributed but also changes over time.

2.1 Modeling

Each hunter is modeled as a reinforcement learning agent in an unknown environment, where there is no communication with the other agents, and there are no intermediate subgoals for which intermediate rewards can be



O \ A	A1	···	Al
O1	W(O1,A1)	···	W(O1,Al)
·	·	·	·
Om	W(Om,A1)	···	W(Om,Al)

Figure 2: Model of a Profit-sharing Agent

given. Thus, no reward is generated until the agent reaches a prey. It should be noted that there are other agents within the environment that are also learning independently of each other, without sharing sensory inputs or policies. As a result, the other agents appear as additional components within the environment, whose behavior is dynamic and unpredictable.

Each agent consists of four modules (Figure 2); a *State Recognizer*, a *LookUp Table*, an *Action Selector*, an *Episodic Memory* and the *Learner*, which includes the Profit-sharing algorithm. Initially, the agent observes O_t, the partially available state of its environment at time t. An action is then selected from the action set A_t, which contains all the available actions at time t, using a *Roulette Selection* method, which select an action in proportion to its weight (see Fig.2(a)(b)). After the action is selected, the agent determines if a reward has been generated. If there is no reward after action a_t, the agent stores the state-action pair, (O_t, a_t), in its *Episodic Memory*, and repeats this cycle until a reward is generated. The process of moving from a start state to the final reward state is known as an *episode*. Once the agent receives the reward, R, it reinforces the rules stored in its episodic memory by modifying the look-up table using the credit assignment function which satisfies *Rationality Theorem*(Miyazaki 1999)(see Section 3.).

2.2 Requirements of Multi-agent Reinforcement Learning

There are three problems which have previously been encountered when reinforcement learning approaches are applied to domains with the same characteristic as our domain. The first is due to the "agent's sensory limitation", in which the agent is fooled into perceiving two or more different states as the same state. This is known as *perceptual aliasing* (Whitehead 1990). If all these different states require the same action, then perceptual aliasing is desirable, as it results in a generalization of the state space. However, if each state requires a different action, then this can lead to the agent becoming "confused", and hence performing the wrong action. The second problem is due to *concurrent learning* (Sen 1995)(Arai 1997), in which the dynamics of the environment vary unpredictably as, due to learning, each agent modifies its own policies and behaviors asynchronously. Thus, midway though the learning process, an agent cannot estimate the model of state transitional probabilities for its environment. These two problems can result in non-determinism within state transitions. The third problem is that the approach should minimize the amount of memory required to make an agent behave effectively.

3. Profit-sharing Approach

The Profit-sharing algorithm is different from other methods, such as Q-learning (Watkins 1992) and Temporal Difference Learning (Sutton 1988), which make the assumption that an environment can be modeled by a Markov Decision Process(MDP). Under the Markovian assumption, the agent can perceive a set S of distinct states of its environment, and has a set A of actions that is can perform. At each discrete time step t, the agent senses the current state o_t, chooses a current action a_t, and performs it. The environment responds by giving the agent a reward $r_t = r(o_t, a_t)$ and by producing the succeeding state $o_{t+1} = \delta(o_t, a_t)$. These functions δ and r are part of the environment and are not necessarily known by the agent. Domains that obey the Markovian assumption are called MDP as the functions $\delta(o_t, a_t)$ and $r(o_t, a_t)$ depend only on the current state and action.

An agent that learns using Q-learning modifies the value of the current state-action pair, $Q(o_t, a_t)$, using the value of sequential state $V(o_{t+1})$ to estimate the current value $V(o_t)$, as shown in Figure 3, Eq. (1). At each time step, the agent updates $Q(o_t, a_t)$ by recursively discounting future utilities and weighting them by a positive learning rate α. Thus, $Q_n(o_t, a_t)$ corresponds to the nth modification of Q's components, o_t and a_t. The parameter $\gamma (0 < \gamma < 1)$ is a discount parameter, and $V(o_{t+1})$ is given by Figure 3, Eq.(2). Therefore, if o_{t+1} is an aliasing state, the agent fails to estimate not only the value of the current state-action pairs o_t, but also the values of the following states o_{t+1} and corresponding actions. This failed estimation will then be propagated through the learning process.

Profit-sharing uses trial and error experiences, and reinforces effective rules, instead of estimating values using the sequential state's value. Therefore, it uses this policy to escape states susceptible to perceptual alias-

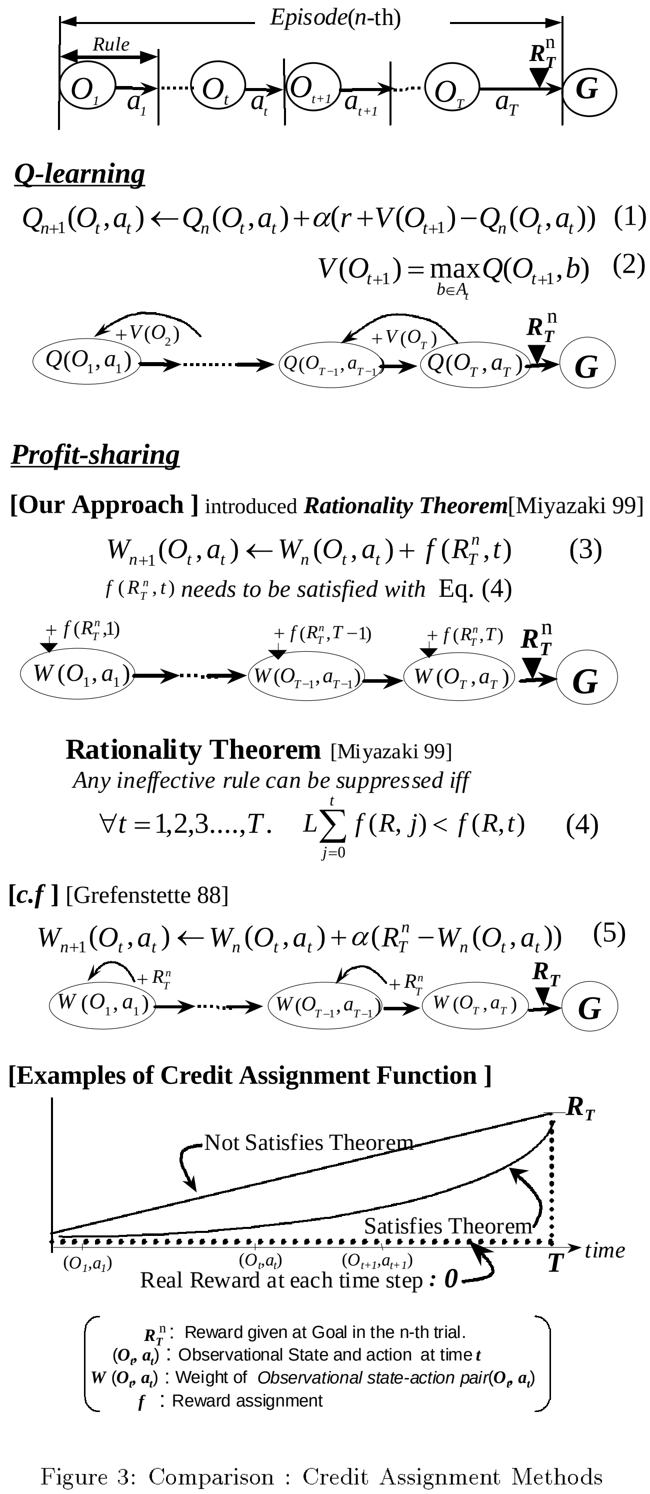

Episode(n-th)
Rule
O_1 a_1 O_t a_t O_{t+1} a_{t+1} O_T a_T R_T^n G
Q-learning
$Q_{n+1}(O_t,a_t) \leftarrow Q_n(O_t,a_t)+\alpha(r+V(O_{t+1})-Q_n(O_t,a_t))$ (1)
$V(O_{t+1}) = \max_{b\in A_t} Q(O_{t+1},b)$ (2)
$+V(O_2)$ $+V(O_T)$ $Q(O_1,a_1)$ $Q(O_{T-1},a_{T-1})$ $Q(O_T,a_T)$ R_T^n G
Profit-sharing
[Our Approach] introduced Rationality Theorem[Miyazaki 99]
$W_{n+1}(O_t,a_t) \leftarrow W_n(O_t,a_t)+f(R_T^n,t)$ (3)
$f(R_T^n,t)$ needs to be satisfied with Eq. (4)
$+f(R_T^n,1)$ $+f(R_T^n,T-1)$ $+f(R_T^n,T)$ R_T^n $W(O_1,a_1)$ $W(O_{T-1},a_{T-1})$ $W(O_T,a_T)$ G
Rationality Theorem [Miyazaki 99]
Any ineffective rule can be suppressed iff
$\forall t=1,2,3....,T. \quad L\sum_{j=0}^{t} f(R,j) < f(R,t)$ (4)
[c.f] [Grefenstette 88]
$W_{n+1}(O_t,a_t) \leftarrow W_n(O_t,a_t)+\alpha(R_T^n - W_n(O_t,a_t))$ (5)
$+R_T^n$ $+R_T^n$ $W(O_1,a_1)$ $W(O_{T-1},a_{T-1})$ $W(O_T,a_T)$ R_T G
[Examples of Credit Assignment Function]
R_T
Not Satisfies Theorem
Satisfies Theorem
time
(O_1,a_1) (O_t,a_t) (O_{t+1},a_{t+1}) T
Real Reward at each time step : 0
R_T^n : Reward given at Goal in the n-th trial.
(O_t, a_t) : Observational State and action at time t
$W(O_t, a_t)$: Weight of Observational state-action pair(O_t, a_t)
f : Reward assignment

Figure 3: Comparison : Credit Assignment Methods

ing. This property also makes the agent robust within uncertain domains, and reduces memory requirement as it only stores rules which are essential to navigate the state space. Since our *Pursuit Game* domain cannot be assumed to be an MDP and has a very large state space, Profit-sharing is a more suitable approach.

3.1 Concept of our Rationality

Our multi-agent reinforcement learning approach is based on Profit-sharing, originally proposed by (Grefenstette 1988). The original version used Profit-sharing as a credit assignment method based on trial-and-error experiences, without utilizing any form of value estimation. However, this approach does not guarantee the rationality of an acquired policy. To guarantee convergence to a rational policy in a non-Markovian domain, we introduce the *Rationality Theorem* (Miyazaki 1999), which the credit assignment function should satisfy. Although in general, the acquired policy need not be optimal for multi-agent situations, it is important that this policy is rational. A rational policy is one that is guaranteed to converge on a solution; i.e. the agent should not become trapped within infinite loops in the state machine. The function that assigns a reward among rules in the episode is called a *credit assignment function*, f (in Figure 3, Eq.(3)(4)), where L is the number of available actions (;L appears in Eq.(4) in Figure 3) at each time step, and $f(R,t)$ denotes an assignment value for the state-action pair which is fired at time t.

In our Profit-sharing algorithm, the weight of each rule is reinforced according to its distance from the goal. For example, at time t, an agent enters state o_t and selects action a_t, and continues this cycle until it receives a reward R at time T. At this point, the episode consists of the state-action pairs $((o_t,a_t),(o_{t+1},a_{t+1}),\cdots\cdots,(o_T,a_T))$, as shown in Figure 3. Each state-action pair is then assigned some credit, according to the function $f(R,t)$. Thus, the last state-action pair, (o_T,a_T) is assigned credit R; the penultimate state, (o_{T-1},a_{T-1}), is assigned credit $f(R,T-1)$, and so on. The weight of each state-action pair within the episode is modified by Eq.(3) in Figure 3. It is important to note that the weight of o_{t+1} is not required when modifying the weight of o_t.

3.2 How to realize an Effective behavior

To illustrate the difference between our approach and the DP-based one, consider the state diagram in Figure 4, where the state value, V, represents the minimum step to a reward. In this example, the highest value of V is 1, and an agent moves to the smaller valued state. The values of states 1a and 1b, V(1a), and V(1b) are 2 and 8, respectively. Although these two states are different,

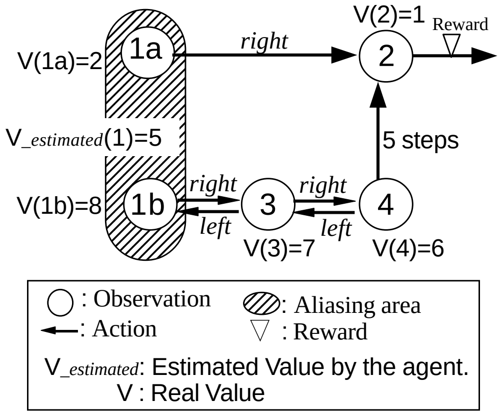


Figure 4: An example of confusion in perceptual aliasing area This figure is quoted from (Miyazaki 1999)

they are perceived by the agent as being the same state (i.e. state 1).

If the agent visited to state 1a and 1b with equal frequency, and it estimates these state values using DP-based algorithm, $\mathsf{V}(1) = \frac{2+8}{2} = 5$. Therefore the value of state 1 became smaller than the value of state 3, i.e. $\mathsf{V}(3) = 7$. If the agent uses these state values, it will move *left* into state 3. Otherwise, the agent moves *right* into state 1. This means that the agent learns the irrational policy where it only transits between states 1b and 3.

On the other hand, the agent using Profit-sharing just reinforces the state-action pairs; $(1, Right)$ as a successful rule when its observable state is *state* 1. Therefore, it uses this policy to escape states susceptible to perceptual aliasing. This property also makes the agent robust within uncertain domains, and reduces memory requirement as it only stores rules which are essential to navigate the state space, while DP-based methods require to keep the value of whole state spaces.

In order to acquire the effective policy, we need to use the credit assignment function which satisfies this theorem. Take the state diagram in Figure 4, for example, to explain how to discard cases that result in cyclic behavior. At time t, an agent is in the state 3. If it moves left, it enters state 1. It can then return to state 3 by moving right. Thus, the agent could cycle between these two states indefinitely, before moving onto another state (e.g. state 4) which will lead to the goal (state 2). If the agent's episode consists of the state-action pairs: $(3, Left)_{t=1} \rightarrow (1, Right)_{t=2} \rightarrow (3, Left)_{t=3} \rightarrow (1, Right)_{t=4} \ldots (3, Right)_{t=T-3} \rightarrow (4, Up)_{t=T-2} \rightarrow (2, Right)_{t=T-1} \rightarrow$, $(Goal)_{t=T}$, and the function f is constant, the weight of $(3, left)$ will be larger than that of $(3, right)$, as the agent will have visited $(3, left)$ several times. Also in the cases where the function f is an arithmetic decreasing function, or an geometric function without consideration of the number of available actions, the agent is not guaranteed to get out of the loop. On the other hand, if f satisfies the theorem, the weight of $(3, right)$ will always larger than that of $(3, left)$, whatever path an agent moves in its trials. Even though the agent falls into the several loops in one trial, the weight of action which make it escape from there will dominate the other actions which caused the looped behavior.

3.3 Related Work

The perceptual aliasing problem has been addressed by a number of studies, and to date, two solutions have been proposed. The first is *memory-based* (Chrisman 1992)(MacCallum 1993), which maintains a history of state-action pairs for each episode. The second adopts a *stochastic policy* (Jaakkola 1994) where the agent selects a random action to escape from partially observable states. The first solution requires additional memory to store the tuple history. The approach adopted by our Profit-sharing algorithm is based on the latter solution, which includes TD(1) and the Monte-Carlo methods (Singh 1996) in that they do not use the values of consecutive states. Our approach differs from TD(1) and Monte Carlo in that our method does not use the values of state (or state-action pairs) which require extra memory to keep eligibility traces to manage the delayed reward.

A number of studies have recently explored the concurrent learning problem. Sub goals were used by (Mataric 1997)(Stone 1999) to find effective rules using Eq.(5) (Figure 3), but there is no theoretical background for this approach. This problem has also been discussed theoretically for the Q-learning approach (Hu 1998).

There are many context of the *Pursuit Game* with different motivations in the research of Multi-agent reinforcement learning. Q-learning is applied in (Tan 1993)(Ono 1997). The focus of (Tan 1993) is to evaluate the effectiveness of information sharing among agents where the relations among them are pre-defined. (Ono 1997) suggests a method to avoid the state explosion problem in reinforcement learning. Their results show that Q-learning could work well even in the multi-agent environment. But in their experiments, a hunter does not need to consider the effects of other hunters, i.e.; the relative positions among the hunters could be ignored, the randomly moving prey is independent of the hunters' policies, and the hunters are given a common goal to move in the same direction towards the prey.

There has also been research work concerned with multiple goals in a single agent environment. In (Humphry 1997) and (Whitehead 1993), the active goals are changing in the environment, but the agent does not need to achieve all the goals but only to act appropriately for each combination of goals. A composite goal is defined by sequentially combined multiple elemental goals in (Singh 1992), where rewards are generated only when the system achieves a subgoal in a prescribed order. The definition of Singh's(Singh 1992) composite goal is related to ours, but our elemental goal, which correspond to capturing one prey, is not independent of capturing the rest of the preys.

4. Experiments

To compare our Profit-sharing approach (presented in the previous Section) with Q-learning, Modular-Q-learning(Ono 1997) and *Whitehead's*(Whitehead 1993) approach, we experimented with there conditions, each of which is labeled H3P1, H3P2 and H3P3, where the number of non-learning-preys are 1, 2 and 3 in the environment respectively.

In our experiments, three hunters started from different locations and their task was to capture the whole prey(s) in the environment as quickly as possible. There are four actions within the action set, $A_t = \{Stay, Right, Left, Straight\}$, but hunters and preys cannot occupy the same position. Each hunter could see other hunters and prey only when they exist in his sight. Hunters and preys act in turns. In each episode, they are set in random positions of the triangular toroidal environment as shown in Figure 1. The reward R is given only when hunters have finished capturing all the preys.

The parameters are set as follows.

Profit-sharing: A geometrically decreasing function $f(R,t) = R(0.3)^{T-t}$ (*common ratio*= 0.3) was used to assign a credit to each state-action pair sharing a reward(R). It satisfies the *Rationality Theorem* described above. The hunter selects its action by a roulette-based weighting of the conflicting rule set.

Q-learning: The *learning rate*= 0.05 and *discounting factor*= 0.9. When the hunter reaches the goal state (i.e. capturing 1 prey), it receives a reward of 1.0. The Q-learning agent uses the Boltzmann distribution, as shown in Eq.(6), to select its action.

$$p(a_i|s) = \frac{e^{Q(s,a_i)/T}}{\sum_{k \in actions} e^{Q(s,a_i)/T}} \quad (T = 0.2) \qquad (6)$$

In the Modular-Q-learning, the learner of each hunter consists of two modules, and the action which has highest Q-value will be selected in each step.

When one prey is captured, the episode of each hunter terminates and each hunter assigns credit on the rules of his history(episode). Then the hunters repeat the above process until there is no prey in the environment. The evaluation metric is determined by averaging the number of steps required by hunters to capture the whole preys.

Approach	*Environments* Number of Hunters &Preys	*Required Number of Steps to capture the 1st Prey Av.(S.D)* After **5,000** Episodes	After **50,000** Episodes	After **500,000** Episodes	After **1,000,000** Episodes
Profit-sharing	*3Hunters & 1 Prey*	35.8(4.6)	22.1(2.8)	19.4(1.6)	17.6(1.4)
	3Hunters & 2 Prey	96.2(18.2)	34.3(4.8)	23.1(1.8)	22.3(1.7)
	3Hunters & 3 Prey	426.3(83.2)	87.0(9.4)	34.5(3.0)	28.0(2.4)
Monolithic Q-learning	*3Hunters & 1 Prey*	217.4(54.6)	120.5(21.6)	79.4(19.6)	52.4 (11.8)
	3Hunters & 2 Prey	**Unstable behavior after 1,000,000 episodes**			672.9(547.6)
	3Hunters & 3 Prey				2188.4(6624.6)
Modular Q-learning*	*3Hunters & 1 Prey*	88.9(28.7)	58.3(15.6)	35.3(7.6)	21.8 (6.3)
	3Hunters & 2 Prey	**inapplicable**			
	3Hunters & 3 Prey				

Modular Q-learning [Ono 1997]

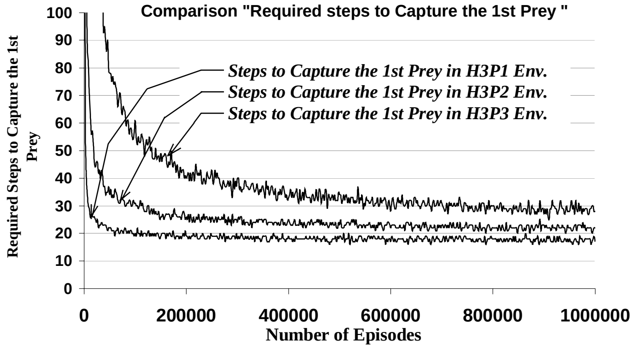


Figure 5: Performance of Learning: Comparison "Required steps to Capture the 1st Prey

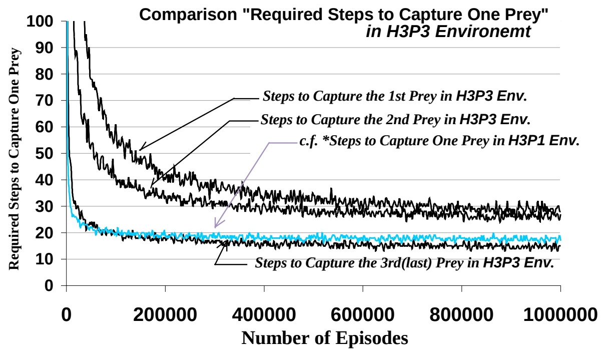
Experiments consist of 10 trials, each of which consists of 1,000,000 episodes. The lookup table is reset for each trial.

4.1 Result1: Emerging Behavior without Communication

First, we apply Profit-sharing, Q-learning(Monolithic) and Modular-Q-learning (Ono 1997) to the hunters without communication among them. Monolithic-Q-learning is the simple One-step Q-learning without any predefined structure. Figure 5 shows the learning curves of the required steps to capture the first prey in the cases of H3P1, H3P2, and H3P3. Figure 6 shows the learning curves of the required steps to capture one prey in the H3P3. In each figure, the x-axis indicates the number of episodes and the y-axis indicates the average of required steps in 10 trials. The purpose of these experiments are to show the performance of the Profit-sharing in the multiple-goals and agents environment overcoming perceptual aliasing and agents' concurrent learning.

When the hunter learns by Profit-sharing, each hunter does plan as to the path and schedule of the preys without global information. But the larger the number of preys exist in the environment, the more steps are required and also increase the standard deviation in capturing the first prey. In addition, convergence becomes slower for larger number of preys, because the state spaces are getting larger. The state space in this experiment is $((Num.ofHunters - 1) + (Num.ofPreys))^{15}$. This indicates that it is getting more difficult to coordi-

Environments 3 Hunters ,3 Preys	*Required Number of Steps to capture **One** Prey*			*Av.(S.D)*
	After **5,000** Episodes	After **50,000** Episodes	After **500,000** Episodes	After **1,000,000** Episodes
To capture the 1st Prey	486.6(190.7)	78.0(5.3)	34.3(2.0)	28.6(1.8)
To capture the 2nd Prey	218.7(61.5)	53.7(3.8)	29.5(1.8)	26.8(1.5)
To capture the 3rd Prey	50.9(22.0)	23.5(3.6)	16.4(2.6)	14.8(1.2)
*C.f. To capture one Prey in the H3P1**	35.6(7.6)	22.3(3.9)	19.3(3.2)	17.0(1.8)



*** *Note* : To capture one prey in the H3P1 Env. is Not the same as the case with Global-agent to manage the target prey in each stage of capture the prey.**

Figure 6: Performance of Learning in H3P3: Comparison:"Required steps to Capture each Prey in H3P3

nate the hunters' decisions as to which prey to attack. But, what we notice in Figure 6 is that the required steps to capture the *last(3rd) prey* in H3P3 environment is less than that in H3P1 condition after episode 80,000. This fact implies that hunters pursue multiple preys not independently but simultaneously as we expected. The results illustrated in Figure 5 indicate that Q-learning-approaches(Monolithic and Modular) fail to converge for either world (only the results for H3P1 world are acceptable).

We found that the hunter created the deterministic policy, which means that one observable state o_i is mapped to one action a_j, even in the kind of deadlocked situation where we could not design the knowledge such as the multiple preys are in the same distance from the hunters (as defined in Eq.(7)). But in some states (24.6% of the hunter's state space in H3P3 case), the hunter created the stochastic policy where two actions dominated over other actions. These stochastic policy works effectively in our domain where the hunter does not have any information about actions(output) of other hunters' and preys'. i.e.; the number of whole state space is around 9240 and the number of state which has deterministic policy is around 2290, In the other states (6950), hunter takes a stochastic policy.

As to the state spaces, we found that only 46.2% of the states are necessary to capture the preys in H3P3. Because the rest of these states are reinforced little, there are no certain strategy toward them. Therefore, when we use this learning result in the off-line situation, we can shrink the state space. On the other hand, DP-based methods like Q-learning require to keep the value

Environments 3 Hunters ,3 Preys	*Required Number of Steps*			*Av.(S.D)*
	After **5,000** Episodes	After **50,000** Episodes	After **500,000** Episodes	After **1,000,000** Episodes
To capture 3 preys Without Global	762.9(186.4)	155.1(7.2)	84.5(3.4)	76.2(1.8)
To capture 3 preys With Global	136.4(32.6)	69.3(7.8)	59.6(7.1)	59.4(6.4)
To capture 1 prey With Global	49.5(9.8)	28.6(4.3)	24.7(3.7)	22.1(2.6)
*C.f. To capture one Prey in the H3P1**	35.6(7.6)	22.3(3.9)	19.3(3.2)	17.0(1.8)

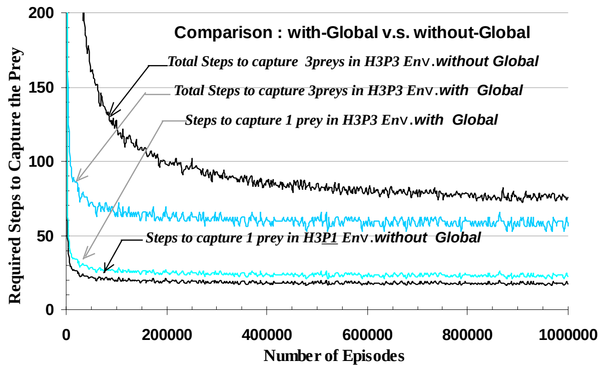


Figure 7: Performance of Learning in H3P3: Comparison with the Global-agent-Environment

of whole state spaces, because they use the value of consecutive state to learn.

4.2 Results2 : Comparison with Centralized Scheduling of Prey capture

Second, we compared with the condition in which a global agent schedules the order of prey capturing. To evaluate performance of the hunters without global knowledge, we compared with the baseline condition in which a single global agent schedules the ordering of prey capture. In this case, the global agent is given the information about the location of all the preys and hunters, then selects the target prey by Eq.(7).

$$Prey_j = \arg\min_{j \in preys} \sum_i distance(Prey_j, Hunter_i) \quad (7)$$

Then, all hunters converge on the target prey and neglect the other preys. In this case, a hunter ignores the other preys although they could be in his sight. After capturing the 1st prey, the global agent decides the next target and hunters repeat the same procedure as mentioned above.

Figure 7 shows the learning curves of the required steps to capture the 3 preys and 1 prey in both methods. The x-axis indicates the number of episodes and the y-axis indicates the average of required steps in 10 trials.

The with-global-agent condition shows more effective performance than without-global-agent to capture whole preys because the hunters' target is always consistent among them. In the with-global-agent method, the state space size of each hunter's is constant $(((Hunters-1)+1))^{15})$, regardless of number of preys. And also the acquired policy of capturing the 1st prey could reuse to capture the second and third prey. But, what we notice here is that the required steps to capture the *1 prey* in the H3P3-with-global is larger than that in the H3P1-without-global condition. This fact implies that hunters in the H3P3-with-global seem to be thrown into a kind of perceptual aliasing and to be compelled them to move unnatural way because they are concealed non-target prey from their sights. And in with-global method, the hunters could not pursue multiple preys opportunistically which is realized in the without-global-method.

To understand what caused these results, we draw out the change of the policy through their learning process as shown in Table 1. When a hunter does not see anything in its sight, the perceptual aliasing problem occurs in this toroidal triangular world. The hunter has a set of four available rules in this situation. While, the Q-value of each rule changes every episode, both of methods using Profit-sharing are significantly reinforced in the early stage due to exploitation-intensive property of the Profit-sharing; i.e. they just add credit on the successful rules after an episode. The interesting results in Table 1 is that each Profit-sharing hunter learned different policies among one another, and they seem to play its own role for capturing the prey, though their initial policies were the same as one another.

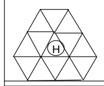
This is the worthy of notice because each hunter specified the effective actions though their initial position changes randomly in every episodes. In the *without global method*, Hunter1 and Hunter2 acquired the deterministic policy which is *Right*(99.8%) and *Straight*(95.3%) respectively. On the other hand, Hunter3 got the stochastic policy which consists of *Left*(58.5%) and *Straight*(39.6 %) after 1,000,000 episodes. Such a combination seems the best for getting out of this perceptual aliasing state. Also in the *with global method*, hunters acquired almost same deterministic policy and effective stochastic policy as the case of *without global*, but it is faster to converge on their firm resolution than *without global* case. Because the number of state spaces of *without global* case is much larger than that of *with global* one.

5. Discussion

Profit-sharing succeeds in finding an effective plan and schedule without any control knowledge. From our several experiments, we can observe some interesting behaviors as follows.

1. **Despite the absence of a global scheduling mechanism, the hunters capture the preys in a "reasonable" order** (e.g., capture closest prey first). The arrangement of hunters at each time step is determined by the independent learning of the reinforcement agents.

Table 1: Convergence Processes of Hunter's Policy when there is no other entity in a hunter's sight.

Observation 	***Hunter's ID***	Percentages of the weight (%)											
		After 50,000 Episodes				After 500,000 Episodes				After 1,000,000 Episodes			
		Stay	**Right**	**Left**	**Straight**	**Stay**	**Right**	**Left**	**Straight**	**Stay**	**Right**	**Left**	**Straight**
Profit-sharing **without Global**	***H1***	23.4	54.6	19.6	2.4	2.6	94.3	2.8	0.3	0.1	99.8	0.1	0.0
	H2	15.3	17.9	26.5	40.3	2.0	1.9	6.4	89.7	0.3	0.4	4.0	95.3
	H3	16.7	18.0	42.6	22.7	2.0	3.4	54.6	40.0	0.0	1.9	58.5	39.6
Profit-sharing **with Global**	***H1***	5.3	81.3	11.3	2.1	0.7	98.6	0.4	0.3	0.1	99.9	0.0	0.0
	H2	4.7	5.0	4.7	85.6	1.4	1.8	1.2	95.6	0.1	0.1	1.2	98.6
	H3	13.9	39.3	14.4	32.4	2.0	46.5	2.9	48.6	0.8	47.7	0.4	51.1
Monolithic **Q-learning**	***All Hunters***	***the policy is changed from beginning to end.***											

2. **Each hunter plays its own role for capturing the prey.** After learning by Profit-sharing, each hunter plays its own role for capturing the prey, even when their initial position is randomly changed at every episode. The roles of hunters are determined probabilistically. The differentiation among them depends on the seed of random numbers used in the early stages of their learning processes.

3. **He who runs after two hares will catch neither.** The larger the number of preys, the more steps are required to capture the first prey. In fact, as the number of preys increases, the number of options (i.e., preys to pursue) available to each hunter increases, thus making coordination among hunters more difficult. This fact implies that target of each hunter is scattered. Here, the proverb, "He who runs after two hares will catch neither" seems to be true.

4. **Kill two birds with one stone.** The required number of steps to capture the last prey in the multiple-prey environment is less than the number of steps required to capture the first prey in the single-prey environment. This fact implies that hunters pursuit multiple preys simultaneously. Here, the proverb, "Kill two birds with one stone" is realized.

Though Q-learning cannot be applied in non-Markovian environments in nature, it has been often used as an engine of multi-agent reinforcement learning. The first experiment shows that the method based on Q-learning oscillates between perceptual aliasing states, it fails to acquire a stable policy in the multiple-prey environment. The cause of the oscillation is not only the limitation of the sensory input but also the change of hunters' policies even with the smaller learning rate ($\alpha = 0.05$). Some control knowledge, such as hierarchical structure will be necessary to overcome this problem of Q-learning.

On the other hand, Profit-sharing can be applied in non-Markovian environments without any control knowledge. Though Profit-sharing cannot always find an optimal policy, it can find an effective policy very quickly because it reinforces the successful state-action pairs immediately after one episode. In other words, the Profit-sharing-agent needs only one successful episode to acquire its effective policy, and then its following iterations of the episode improve its policy. In contrast, Q-learning takes a long time to propagate the reinforcement throughout all the state-action pairs. This feature of Profit-sharing is appropriate for multi-agent learning systems where the agent needs to adapt as quick as possible.

The results of our experiments suggest that Profit-sharing also needs a hierarchical structure to reduce the state spaces, and to handle higher level of the agents' behaviors. However, the traits of Profit-sharing we observed here indicate that it is easier to introduce hierarchy into Profit-sharing-agent than Q-learning-agent. Especially, the trait which Profit-sharing does not require to keep the value of whole state-space in its look-up table, makes it easy to scale up to be a structured algorithm. In other words, when we use this learning result in the off-line situation, we can shrink the state space and keep only the useful part of the look-up table.

6. Conclusion

In this paper, we present a variant of the Profit-sharing algorithm, and demonstrate its effectiveness within a multi-agent domain where agreement among the agents is required without sharing information. Profit-sharing solves the problems of perceptual aliasing and concurrent learning while minimizing memory requirement. This makes reinforcement learning more amenable for multi-agent domains. While Profit-sharing is appropriate for an episodic task where the reward is only given at end of the goal, it is less suited for domains that include intermediate rewards.

We plan to combine Profit-sharing with other bottom-up approaches, such as genetic algorithms, and with top-down approaches for real world applications.

References

Arai,S.,Miyazaki,K., Kobayashi,S.: Generating Cooperative Behavior by Multi-Agent Reinforcement Learning, *Proceedings of 6th European Workshop on Learning Robots* p111-120 (1997).

Benda M., Jagannathan, V. and Dodhiawalla, R. : On Optimal Cooperation of Knowledge Sources, Technical Report BCS-G2010-28, Boeing AI Center, 1985.

Chrisman, L.: Reinforcement learning with perceptual aliasing: The Perceptual Distinctions Approach, *Proceedings of the 10th national Conference on Artificial Intelligence*, pp.183-188 (1992). Top-Down Search for Coordinating the Hierarchical Plans of Multiple Agents. *Proc. of the 3rd International Conference on Autonomous Agents,* pp252-259 (1999).

Firby, R.J., An Investigation into Reactive Planning in complex Domains, *Proceeding of 10th National Conference of Artificial Intelligence '87*, pp. 202-206 (1987).

Georgeff, M.P.: Communication and interaction in multi-agent planning, *Proc. of the 3rd national Conference on Artificial Intelligence*, pp.125-129 (1983).

Gasser, L.,Rouquette, N., Hill, R.W. and Lieb, J.:Representing and Using Organizational Knowledge in Distributed AI Systems. in Gasser, L. and Huhns, M. H. (eds.),*Distributed Artificial Intelligence, Vol.2*, Morgan Kaufmann, pp55-78(1989).

Grefenstette, J. J.: Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms, *Machine Learning Vol.3*, pp.225-245(1988).

Humphry, Mark : Action Selection using Reinforcement Learning, *Proc. of the Fourth International Conference on Simulation of Adaptive Behavior*, pp.135-144 (1997).

Hu, Junling and Wellman, Michael P.: Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm *Machine Learning: An Artificial Intelligence Approach, Vol.2*, pp.593-623. Morgan Kaufman (1986).*Proc. of the 15th International Conference on Machine Learning,* pp.242-250(1998).

Jaakkola, T., Singh,S.P. and Jordan, M.I.: Reinforcement Learning Algorithm for Partially Observable Markov decision Problems, *Advances in Neural Information Processing Systems 7* (NIPS-94), pp.345-352 (1994).

Levy, R. and Rosenschein, J.S.: A Game Theoretic Approach to Distributed Artificial Intelligence and The Pursuit Problem, Proc. of the 3rd European Workshop on Modelling Autonomous Agents in a Multi-Agent World, pp.129-146(1992).

Littman, L.Michael : Markov games as a framework for multi-agent reinforcement learning, *Proc. of 11th International Conference on Machine Learning*, pp.157-162(1994).

MacCallum, R. A.: Instance-Based Utile Distinctions for Reinforcement Learning with Hidden State, *Proc. of 12th International Conference on Machine Learning*, pp387-395(1993).

Mahadevan, S.: To discount or not to discount in reinforcement learning, *Proc. of 10th International Conference on Machine Learning*, pp205-211(1993).

Mataric, J.M. : Reinforcement Learning in the Multi-Robot Domain, *Autonomous Robots Vol.4 No.1* pp.73-83 (1997)

Miyazaki, K. and Kobayashi, S.: Proposal for an Algorithm to Improve a Rational Policy in POMDPs, *1999 IEEE International Conference on Systems, Man and Cybernetics*, p55-61 (1999).

Ono, N., Fukumoto, K. and Ikeda, O. : Collective Behavior by Modular Reinforcement Learning Animats, *Proc. of the Fourth International Conference on Simulation of Adaptive Behavior*, pp.618-624 (1997).

Rummery, G.A. : Problem Solving with Reinforcement Learning. Ph.D. thesis, Cambridge University. (1995)

Sen, S. and Sekaran, M. : Multiagent Coordination with Learning Classifier Systems, in Weiss, G. and Sen, S.(eds.), *Adaption and Learning in Multi-agent systems*, Berlin, Heidelberg. Springer Verlag, pp.218-233(1995).

Singh, S.P.: Transfer of learning by composing solution of elemental sequential tasks, *Machine Learning Vol.8*, pp.323-339(1992).

Singh, S.P., Jaakkola, T. and Jordan, M.I. : Learning Without State-Estimation in Partially Observable Markovian Decision Processes,*Proc. of the 11th International Conference on Machine Learning*, pp.284-292(1994).

Singh, S.P. and Sutton, R.S. : Reinforcement Learning with Replacing Eligibility Traces, *Machine Learning* Vol.22, pp.1-37(1996).

Stephens, L. and Merx, M. : Agent Organization as an Effector of DAI System Performance. In Miroslav Benda (ed.), *Proc. of th 9th Workshop on Distributed Artificial Intelligene*, pp.263-292(1989).

Stephens, L. and Merx, M. : The effect of agent control strategy on the performance of a DAI pursuit problem, Proc. of the 10th International Workshop on Distributed Artificial Intelligene (1990).

Stone, P. and Veloso, M. : Team Partitioned, Opaque Transition Reinforcement Learning, *Proc. of the Third Annual Conference on Autonoumous Agents*, pp.206-212.(1999)

Sutton, R.S.: Learning to Predict by the Methods of Temporal Differences, *Machine Learning, Vol. 3*, pp.9-44(1988).

Tan, M. : Multi-Agent Reinforcement Learning: Independent vs.Cooperative Agents, *Proc. of the 10th International Conference on Machine Learning*, pp.330-337(1993).

Watkins, C. J. H., and Dayan, P.: Technical note: Q-learning, *Machine Learning Vol.8*, pp.55-68(1992).

Weiss, G. : Adaptation and Learning in Multi-Agent Systems: Some Remarks and a Bibliography, In Weiss, G. editor, *Lecture Notes in Artifitial Intelligence 1042*, Springer-Verlag, pp1-21(1995).

Whitehead, S. D. and Balland, D. H.: Active perception and Reinforcement Learning, *Proc. of 7th International Conference on Machine Learning*, pp.162-169(1990).

Whitehead, S. D. *et al.*: Learning Multiple Goal Behavior via Task Decomposition and Dynamic Policy Merging, in J.H.Connell *et al* (Eds.) : Robot Learning, Kluwer Academic Press, pp.45-78(1993).