2003 International Symposium on Advanced Intelligent Systems
September 25-28, 2003, Jeju, Korea

Generating Cooperative Behavior by Multi-Agent Profit Sharing on the Soccer Game

Kazuteru Miyazaki
National Institution for Academic Degrees and University Evaluation
1-29-1 Gakuennishimachi Kodaira-city
Tokyo, 187-8587 Japan
teru@niad.ac.jp

Takashi Terada
Japan Overseas Cooperation Vulunteer
ce311261@yahoo.co.jp

Hiroaki Kobayashi
Meiji University
1-1-1 Higashimita Tama-ku
Kawasaki, 214-8571 Japan
kobayasi@isc.meiji.ac.jp

Abstract - Reinforcement learning is a kind of machine learning. It aims to adapt an agent to a given environment with a clue to a reward and a penalty. *Q-learning* [8] that is a representative reinforcement learning system treats a reward and a penalty at the same time. There is a problem how to decide an appropriate reward and penalty values. We know *the Penalty Avoiding Rational Policy Making algorithm* (PARP) [4] and *the Penalty Avoiding Profit Sharing* (PAPS) [2] as reinforcement learning systems to treat a reward and a penalty independently. Though PAPS is a descendant algorithm of PARP, both PARP and PAPS tend to learn a local optimal policy. To overcome it, in this paper, we propose *the Multi Best method* (MB) that is PAPS with *the multi-start method* [5]. MB selects the best policy in several policies that are learned by PAPS agents. By applying PS, PAPS and MB to a soccer game environment based on *the SoccerBots* [9], we show that MB is the best solution for the soccer game environment.

1. INTRODUCTION

Reinforcement learning is a kind of machine learning. It aims to adapt an agent to a given environment with a clue to a reward and a penalty. We can classify RL systems into two types. One is focusing on the optimality in Markov Decision Processes (MDPs) such as *Q-learning* (QL) [8] and *k-Certainty Exploration Method* [6]. The other is focusing on the rationality in non-Markovian environments, such as *Profit Sharing* (PS) [1, 7].

We know *the rationality theorem of PS* [7] to guarantee the rationality in non-Markovian environments. It does not support a reward and a penalty at the same time. On the other hand, QL guarantees the optimality in MDPs where there are both a reward and a penalty. However, it is difficult to design an appropriate reward and penalty values. If we set incorrect these values, the agent will learn unexpected behavior [4].

The Penalty Avoiding Rational Policy Making algorithm (PARP) [4] gives one solution for this problem. It can treat a reward and a penalty independently. It guarantees the rationality by suppressing all penalties. However, it has to memorize all rules that have been experienced and descendant states that have been transited by their rules to find all penalty rules. It is difficult to adapt on the large state space problems because of the curse of dimensionality.

Recently, we have proposed *the Penalty Avoiding Profit Sharing* (PAPS) [2] to reduce the problem by connecting PARP with PS. Though PAPS is a descendant algorithm of PARP, both PARP and PAPS tend to learn a local optimal policy.

In this paper, we propose *the Multi Best method*(MB) to overcome the above problem. It is constructed by PAPS with *the multi-start method* [5]. By applying PS, PAPS and MB to a soccer game environment based on *the SoccerBots* [9], we show that MB is the best solution for the soccer game environment.

2. THE DOMAIN

2.1. Target Environments

Consider an agent in some unknown environment. The agent senses the environment as a set of discrete attribute-value pairs and performs an action in M discrete varieties. The environment gives a reward or a penalty to the agent as a result of some sequence of action. We assume that there is a kind of a reward and apenalty in the environment. We will give the agent a reward for achievement of the purpose and a penalty for violation of restriction.

We denote a sensory input as $x, y, \cdots$ and actions as $a, b, \cdots$. A pair of a sensory input and an action is called a *rule*. We denote a rule "if x then a" as xa. A scalar weight, that represents the importance, is attached to each rule. The weight of rule xa is denoted as S_{xa}. The function that maps sensory inputs to actions is called a *policy*. We call a policy *deterministic* if and only if at most one action should be selected in each sensory input. We call a policy *rational* if and only if expected reward per an action is larger than zero.

We call a sequence of rules selected between the previous reward (or a penalty) and the current one an *episode*. For example, when the agent selects $xb, xa, ya, za, yb, (reward), xa, zb, xa$ *and* $yb, (reward)$ in figure 1, there are two episodes (xb, xa, ya, za, yb) and (xa, zb, xa, yb) as

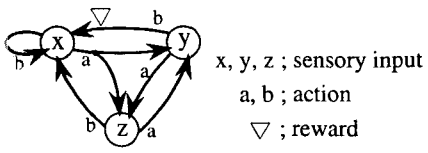

Figure 1: A sample environment.

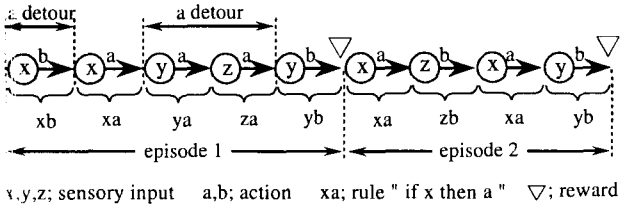

Figure 2: An example of an episode and a detour.

shown in figure 2. We call a subsequence of an episode a *detour* when the sensory input of the first selecting rule and the sensory output of the last selecting rule are the same though both rules are different. For example, an episode 1 in figure 1 has two detours (xb) and (ya, za) as shown in figure 2. The rules on a detour may not contribute to get any reward. The rule that does not exist on a detour in some episode is *rational*. Otherwise, a rule is called *irrational*. Irrational rules should not be selected when they conflict with rational rules.

In this paper, we assume that a rational rule does not change to an irrational rule. It is called *no type 2 confusion* [5]. MDPs is contained in the class.

2.2. Profit Sharing

Profit Sharing (PS) reinforces rules on an episode at once. We call a function that shares a reward (or a penalty) among rules on an episode a *reinforcement function*. The term f_i denotes a reinforcement value for the rule selected at i step before a reward (or a penalty) is acquired. The weight S_{r_i} of rule r_i is reinforced by $S_{r_i} = S_{r_i} + f_i$ for an episode $(r_{W-1}, r_i, ..., r_1, r_0)$ where W is the length of an episode.

When a reward (or a penalty) f_0 is given to the agent, we use the following reinforcement function that satisfies *the Rationality Theorem of PS* [7],

$$f_n = \frac{1}{M} f_{n-1}, \quad n = 1, 2, \cdots, W_a - 1, \qquad (1)$$

where M is the number of actions. For example, at episode 2 in figure 2, the weight of yb, xa and zb are reinforced by $S_{yb} = S_{yb} + f_0$, $S_{xa} = S_{xa} + \frac{1}{2} f_0 + (\frac{1}{2})^3 f_0$ and $S_{zb} = S_{zb} + (\frac{1}{2})^2 f_0$, respectively.

The rationality theorem of PS guarantees the rationality on the class where there is no type 2 confusion.It cannot treat both a reward and a penalty at the same time. If we want to use two type weights such as a reward and a penalty, we should use the other methods discussed in section 2.3 and 2.4.

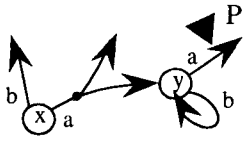

Figure 3: An example of penalty rules (xa, ya) and a penalty state (y).

procedure The Penalty Rule Judgement
begin
  Set a mark on the rule that has been got a penalty directory
  do
    Set a mark on the following state ;
      there is no rational rule or
      there is no rule that can transit to no marked state.
    Set a mark on the following rule ;
      there are marks in the states that can be transited by it.
  while (*there is a new mark on some state*)
end.

Figure 4: The Penalty Rule Judgment algorithm(PRJ); First, we set a mark on the rule that has been got a penalty directory. Second, we set a mark on the state where there is no rational rule or there is no rule that can transit to no marked state. Last, we set a mark on the rule where there are marks in the states that can be transited by it. We can regard a marked rule as a penalty rule. We can find all penalty rules in the current rule set by continuing the above process until there is no new mark.

2.3. The Penalty Avoiding Rational Policy Making algorithm

We know *the Penalty Avoiding Rational Policy Making algorithm* (PARP) [4] as a reinforcement learning system to treat a reward and a penalty at the same time. We call a rule *penalty* if and only if it has a penalty or it can transit to a *penalty state* in which there are penalty or irrational rules. For example, in figure 3. xa and ya are penalty rules, and state y is a penalty state. We call a policy that cannot have any penalty rule a *penalty avoiding policy*.

PARP is a reinforcement learning system to make a penalty avoiding rational policy. To avoid all penalties, it suppresses all penalty rules in the current rule set by *the Penalty Rule Judgment algorithm* (PRJ) as shown in figure 4. After suppressing all penalty rules, it aims to make a deterministic rational policy by PS, *the Rational Policy Improvement algorithm* [4] and so on. Furthermore, it avoids all penalties stochastically if there is no deterministic rational policy.

PRJ has to memorize all rules that have been experienced and descendant states that have been transited by their rules to find all penalty rules. Furthermore, PARP needs $O(MN^2)$ memory to suppress all penalties stochastically where M is the number of actions and N is that of

states. Therefore, in applying PARP to large-scale problems, we are confronted with the curse of dimensionality.

2.4. The Penalty Avoiding Profit Sharing

The Penalty Avoiding Profit Sharing (PAPS) [2] is a reinforcement learning system to overcome the problem of PARP. It uses two type weights of PS. One is for a reward, the other is for a penalty. It selects an action considering with *the weights of PS for a reward* if there is non-penalty rule in current state. On the other hand, if all rules are penalty rules in current state, it selects an action with *the weights of PS for a penalty*. In this case, we share a penalty by *the function (1)*, and select an action that is less reinforced in a rational rule.

It is important to classify an irrational rule from penalty rules. We can judge an irrational rule in an episode by attaching the flag for each rules. In the first, all flags go to up. Next, scanning starts from the state that gets a penalty or a reward on the episode. If the rule has been selected at the first time, the flag goes to down. As result, the rule whose flag is up is the candidate of an irrational rule.

Furthermore, PAPS uses modified PRJ to find a penalty rule. The original PRJ needs $O(MN^2)$ memory since it scans all state spaces. To reduce it, we use ***PRJ on episode*** where the scanning is restricted within each episode. In this case, a penalty rule and the mark to find a penalty state are backtracked on the episode from a penalty state. Such the mark transition on episode is continuing until there is no new penalty state. It needs only $O(MN)$ memory since it memorizes only an episode and penalty rules.

3. PROPOSAL OF THE MULTI BEST METHOD

PARP and PAPS tend to learn a local optimal policy since they are based on PS. To overcome it, we consider that PAPS is performed several times. After performing PAPS several times by *the multi-start method* [5], we make the best trial more learner. It is called *the Multi Best method* (MB).

The usual multi-start method repeats initialization and makes the best policy the final policy. It does not consider improving the final policy more than it. On the other hand, MB performs improving the final policy obtained by the multi-start method.

For example, we show the case where the geme of 100000 steps is perfomed 10 times. PS and PAPS continue updating and learning continuously in each 100000 steps. On the other hand, in MB, after playing a game 10 times for 50000 steps, a game is played 10 times for further 50000 steps using the best policy in it.

4. APPLICATION TO THE SOCCER GAME

4.1. The soccer game environment

We apply PS, PAPS and MB to a soccer game environment based on *the SoccerBots* [9]. SoccerBots simulates the dynamics and dimensions of a regulation RoboCup small size robot league game. Though SoccerBots has two teams of five robots, we use only one learning agent for our team and a keeper agent for each team as shown in figure 5.

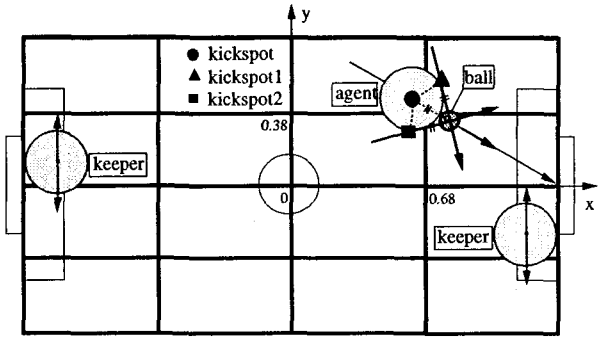

Figure 5: The soccer game environment.

Our learning agent is designed as the PS, PAPS or MB agent. The keeper agent behaves a keeper whose y-axis approaches to the ball's y-axis when it senses the ball. The keeper cannot kick a ball. Field Size is $2.740(W)m$ x $1.525(L)m$. The ball's and the robot's diameters are $0.040m$ and $0.150m$, respectively.

4.2. How to design the learning agent

The learning agent has the following four sensory inputs (S,D,A,K).

- S is the position of the learning agent ; there are 16 rectangle positions in figure 5. Each rectangle is $0.68m$ x $0.38m$.
- D is the distance from the learning agent to the ball ; $D \leq 0.6$, $0.6 < D \leq 1.2$, $1.2 < D \leq 1.8$, $1.8 < D$.
- A is the angle from the learning agent to the ball ; $0 \leq A < \frac{\pi}{2}$, $\frac{\pi}{2} \leq A < \pi$, $-\frac{\pi}{2} < A < 0$, $-\pi \leq A \leq -\frac{\pi}{2}$.
- K is the y-axis position of the keeper ; $K \leq -0.15$, $-0.15 < K \leq -0.05$, $-0.05 < K \leq 0.05$, $0.05 < K \leq 0.15$, $0.15 < K$.

The learning agent has the following four actions.

- can kick
- move to kickspot (figure 5:●)
- move to kickspot1 (figure 5:▲)
- move to kickspot2 (figure 5:■)

When the learning agent achives a goal, it receives a reward. On the other hand, if it makes own goal, it receives a penalty.

4.3. Results and Discussion

We show the results of each method in table 1. The upper of the table is the number of goals and it of the lower is own goals. It contains the averages (ave.) and the standard deviations (S.D.) of 10 games. One game is 100000 steps (actions) long. The PS, PAPS, MB agents exceed the random selection (RND) agent. The RND agent selects an action at random in every step. Especially, MB shows the best solution for this problem. It means that connecting PARP with PS and the multi-start method are useful for this problem.

Table 1: The averages and the standard deviations of 10 games in the soccer game.

		MB	PAPS	PS	RND
goals	ave.	302.1	142.5	124.0	89.5
	S.D.	4.15	24.3	30.0	3.48
own goals	ave.	15.7	23.3	33.2	33.6
	S.D.	0.633	1.85	2.54	2.67

Furthermore, we show all results of 10 games in table 2. The upper of each column is the number of goals and it of the lower is own goals. The PS agent learns an unexpected results (No.10) such that the number of goals is less than that of own goals. On the other hand, The PAPS and MB agents do not learn such results. Furthermore, the MB agent takes the best results in this table. We can see the superiority of the MB agent in the soccer game environment.

5. CONCLUSIONS

The most reinforcement learning systems treat a reward and a penalty on the same weights. There is a problem how to decide an appropriate reward and penalty values. We know PARP as a reinforcement learning system to treat a reward and a penalty independently. PARP is difficult to adapt on the large state space problems because of the curse of dimensionality. On the other hand, we know *the Penalty Avoiding Profit Sharing* (PAPS) to reduce the problem by connecting PARP with PS. Though PAPS is an descendant algorithm of PARP, both PARP and PAPS tend to learn a local optimal policy.

In this paper, we propose *the Multi Best method* (MB) to overcome the above problem. It is constructed by PAPS with *the multi-start method* [5]. MB selects the best policy in several policies that are learned by PAPS agents. By applying PS, PAPS and MB to soccer game environments based on *the SoccerBots* [9], we show that MB is the best solution for the soccer game environment.

In the future, we will extend MB to the environment where there are a few kind of rewards and penalties. Furthermore, we should apply it on the multi-agent environment [3] such that there are five robots for each teams on the SoccerBots.

6. REFERENCES

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

[2] Miyazaki, K., Saiou, J. & Kobayashi, S. Reinforcement Learning for Penalty Avoiding Profit Sharing and its Application to the Soccer Game, *ICONIP'02-SEAL'02-FSKD'02*, pp.335-339, 2002.

[3] Miyazaki, K. & Kobayashi, S. Rationality of Reward Sharing in Multi-agent Reinforcement Learning, *Journal of New Generation Computing*,Vol.91, pp.157-172, 2001.

[4] Miyazaki, K. & Kobayashi, S. Reinforcement Learning for Penalty Avoiding Policy Making. *2000 IEEE International Conference on Systems, Man and Cybernetics*, pp.206-211, 2000.

[5] Miyazaki, K. & Kobayashi, S. Learning Deterministic Policies in Partially Observable Markov Decision Processes, *International Conference on Intelligent Autonomous System (IAS-5)*, pp.250-257, 1998.

[6] Miyazaki, K., Yamamura, M. & Kobayashi, S. k-Certainty Exploration Method : An Action Selector on Reinforcement Learning to Identify the Environment, *Journal of Artificial Intelligence*, Vol.91, pp.155-171, (1997).

[7] K.Miyazaki, M.Yamamura and S.Kobayashi. On the Rationality of Profit Sharing in Reinforcement Learning, *Proc. of the 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing*, pp.285-288, 1994.

[8] Watkins, C. J. II., & Dayan, P.: Technical note: Q-learning, *Machine Learning Vol.8*, pp.55-68, 1992.

[9] http://www.teambots.org

Table 2: All results of 10 games in the soccer game.

No.		1	2	3	4	5	6	7	8	9	10
MB	goals	295	322	278	293	310	314	314	293	299	303
	own goals	14	13	18	15	17	13	16	18	18	15
PAPS	goals	237	168	115	167	178	87	77	79	41	225
	own goals	23	25	16	32	20	21	27	29	24	26
PS	goals	61	110	123	52	119	114	269	61	312	19
	own goals	32	33	32	43	36	21	20	39	32	44