

# REINFORCEMENT LEARNING SYSTEMS BASED ON PROFIT SHARING IN MULTIPLE REWARDS AND PENALTIES ENVIRONMENTS

Kazuteru Miyazaki

*National Institution for Academic Degrees and University Evaluations  
1-29-1 Gakuennishimachi Kodaira-city, Tokyo, 187-8587 Japan*

Shigenobu Kobayashi

*Tokyo Institute of Technology  
4259 Nagatsuta Midori-ku Yokohama, Kanagawa, 226-8502 Japan*

## 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 that is a representative reinforcement learning system can treat a reward and a penalty at the same time. However, there is a problem how to decide an appropriate these values. We know the Penalty Avoiding Rational Policy Making algorithm (PARP) to treat a reward and a penalty independently not to design a reward and a penalty values. Though PARP does not have the problem how to design these values, it cannot treat more than one kind of rewards and penalties. Previously, we extend PARP to such the multiple rewards and penalties environments in which there is no type 2 confusion that is a special class of a Partially Observable Markov Decision Process (POMDP). In this paper, we extend it to the class where there is a type 2 confusion that is the same as a POMDP.

## KEYWORDS

Reinforcement Learning, Reward and Penalty, Profit Sharing, the Penalty Avoiding Rational Policy Making algorithm.

## 1. INTRODUCTION

Reinforcement learning (RL) 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. Traditional RL systems are mainly based on the Dynamic Programming (DP). They aim to get an optimum policy in a Markov Decision Process (MDP). We know *Temporal Difference learning* [Sutton 88] and *Q-learning* (QL) [Watkins 92] as a kind of such the DP-based RL systems. They are often used in many cases since they can guarantee the optimality in MDPs.

The DP-based RL systems aim to optimize its behavior under rewards and penalties values that are given by the designer. However, it has difficult problem how to design appropriate reward and penalty values that fit his purpose. It seems to be required that the same effort as resolving the problem. If we set incorrect rewards and penalties values, we may get unexpected behavior [Miyazaki 00].

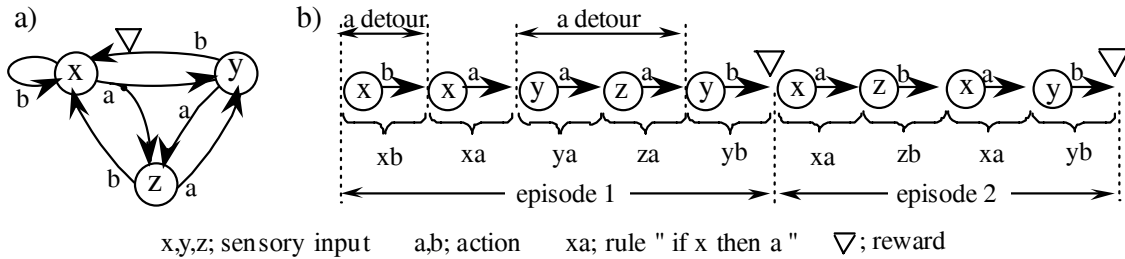
To overcome the problem, we do not assign any values for a reward and a penalty and regard them as independent signals. In the point of view, we have proved *the rationality theorem of Profit Sharing* [Miyazaki 94] and proposed the *Rational Policy Making algorithm* (RPM) [Miyazaki 98] where there is a kind of a reward in an environment. Furthermore, we have proposed the *Penalty Avoiding Rational Policy Making algorithm* (PARP) [Miyazaki 00] to treat a reward and a penalty independently. Though PARP guarantees the rationality, it cannot treat more than one kind of rewards and penalties. Previously, in the paper [Miyazaki 04], we extend PARP to such the multiple rewards and penalties environments in which there is no type 2 confusion that is a special class of a Partially Observable Markov Decision Process (POMDP). In this paper, we extend it to the class where there is a type 2 confusion that is the same as a POMDP.

## 2. THE DOMAIN

### 2.1 Properties of 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. In this paper, we treat multiple rewards and penalties environments such that there are  $r$  kinds of rewards ( $R_i ; i=1,2,\dots,r$ ) and  $p$  kinds of penalties ( $P_j ; j=1,2,\dots,p$ ). A designer that uses the RL system gives the priority among them. For example, if the avoiding of  $P_1$  is more important than the getting of  $R_1$ , and the getting of  $R_1$  is more important than the avoiding of  $P_2$ , the priority  $A(P_1) > G(R_1) > A(P_2)$  is given by the designer. We assume that the priority does not contain two or more than rewards to avoid "between two stools one falls to the ground".

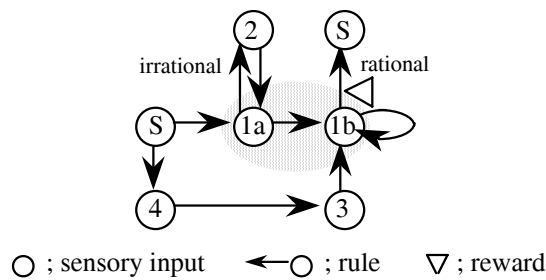
We denote a sensory input as  $x, y, \dots$  and actions as  $a, b, \dots$ . A pair of a sensory input and an action is called a *rule*. We denote a rule "if  $x$  then  $a$ " as  $xa$ . The function that maps sensory inputs to actions is called a *policy*. We call a policy *deterministic* if and only 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.



x,y,z; sensory input    a,b; action    xa; rule "if x then a"     $\nabla$ ; reward

Figure 1. a) A sample environment. B) An example of an episode and a detour.

We call a sequence of rules selected between the previous reward 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 1a), there are two episodes ( $xb, xa, ya, za, yb$ ) and ( $xa, zb, xa, yb$ ) as shown in figure 1 b). 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 b) has two detours ( $xb$ ) and ( $ya, za$ ). The rules on a detour may not contribute to get any reward. The rule that always exists on a detour is irrational. Otherwise, a rule is called rational.



$\circ$  ; sensory input     $\leftarrow \circ$  ; rule     $\nabla$  ; reward

Figure 2. An example of the type 2 confusion.

We treat the environment that is described as a POMDP. A POMDP is a generalization of a MDP that allows for incomplete information regarding the sensory input of the agent. The agent in a POMDP may sense different environmental state as the same sensory input. It is called confusion. Especially, if a rational rule on a POMDP is an irrational on the MDP that has complete information of the POMDP, we say that the environment has a *type 2 confusion* [Miyazaki 98]. Figure 2 is an example of the environment with a type 2 confusion. If the agent has complete information of the environment, it can distinguish hatched sensory inputs (1a and 1b). Otherwise, they are sensed as the same sensory input (sensory input 1).

The action to move up in the sensory input 1 has been classified to a rational rule by the agent, since it

can be regarded as a rational on the sensory input 1b. However, the rule is an irrational on the sensory input 1a.

If there is no type 2 confusion in the environment where there is a kind of a reward, we can get a deterministic rational policy by suppressing all irrational rules and selecting only rational rules [Miyazaki 94, Miyazaki 98]. Otherwise, we cannot always get it by suppressing irrational rules only [Miyazaki 02].

## 2.2 Previous Works

We know the rationality theorem of Profit Sharing (PS) and RPM to learn a deterministic rational policy in the class where there is no type 2 confusion and there is a kind of reward in it. In this paper, we use RPM. It requires 1st and 2nd memories for each sensory input. Each memory requires  $2N$ 's memory where  $N$  is the number of sensory inputs. When the agent selects an action, it describes the action in the current state of the 1st memory. If the agent gets a reward, the contents of the 1st memory are copied to the contents of the 2nd memory. After that, rational rules are memorized on the 2nd memory. If there is an action in the current state of the 2nd memory, the agent selects the action. Otherwise, it selects an action according to some exploration strategy, for example, random selection. For the class where there is no type 2 confusion and there is a kind of reward in it, RPM can get a deterministic rational policy.

RPM cannot treat a reward and a penalty at the same time. If we want to adapt it to the environments, we should use PARP. In PARP, 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. Furthermore, we call a policy that cannot have any penalty rule a *penalty avoiding policy*.

```

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

```

Figure 3. The Penalty Rule Judgment algorithm (PRJ).

PARP can 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 3. After suppressing all penalty rules, it aims to make a deterministic rational policy by PS, RPM 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. It needs  $O(MN^2)$  memory where  $M$  is the number of actions and  $N$  is that of states. Also, PARP needs the same memory to suppress all penalties stochastically.

Though PARP treat a reward and a penalty at the same time, it cannot treat more than one kind of rewards and penalties. Furthermore PARP cannot treat the environment with a type 2 confusion. In this paper, we extend PARP to the multiple rewards and penalties environments with a type 2 confusion.

## 3. EXTEND PARP TO MULTIPLE REWARDS AND PENALTIES ENVIRONMENTS

### 3.1 Preliminary

In this paper, we extend PARP to multiple rewards and penalties environments. We realize it by  $O(MN^2)$  memory that is the same as PARP. We aim to find the policy that satisfies *the priority* among  $r$  kind of

rewards and  $p$  kinds of penalties. A designer that uses the RL system gives the priority. If the designer gives the priority that the avoidance of P1 is more important than the getting of R1 and the getting of R1 is more important than the avoidance of P2, it is described by  $A(P1) > G(R1) > A(P2)$ . Remark that the priority should not contain two or more than rewards to avoid “*between two stools one falls to the ground*”.

The classification of rules should be done by the kinds of rewards and penalties since the judgments of irrational and penalty rules depend on the kinds of rewards and penalties. In the first, we use  $r$  kinds of RPMs to find an irrational rule that may change by a kind of a reward. They are denoted by  $\mathbf{RPM}_i$  for each reward  $i$  ( $i = 1, 2, \dots, r$ ). They need  $rMN$  memories where  $M$  is the number of actions and  $N$  is the number of states. It is  $O(MN)$ . It means that it memories whether each rule is a rational or an irrational for each rewards.

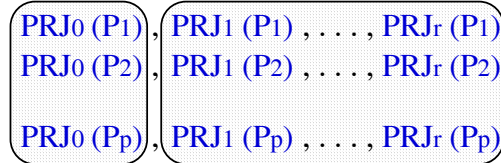


Figure 4.  $p(r+1)$  kinds of PRJs.

Next, we use  $p(r+1)$  kinds of PRJs to find a penalty rule that may change by a kind of a reward and a kind of a penalty. They are denoted by  $PRJ_i(P_j)$  for each reward  $i$  ( $i = 1, 2, \dots, r$ ) and penalty  $j$  ( $j = 1, 2, \dots, p$ ). We show them in figure 4.  $PRJ_i(P_j)$  ( $i \neq 0$ ) can be calculated by executing the penalty rule judgment algorithm about  $P_j$  after suppressing all irrational rules that are found by  $\mathbf{RPM}_i$ . Furthermore,  $PRJ_0(P_j)$  memorizes penalty states and rules when the environment has  $P_j$  without any irrational rule. They need  $p(r+1) \cdot MN$  memories to storage the result of each PRJ. It is  $O(MN)$ . It means that it memories whether each rule is a penalty or not for each rewards and penalties. On the other hand, we need  $O(MN^2)$  memory to execute each RPJ. If we want to reduce it to  $O(MN)$ , we can use *PRJ on episode* where the transition of a mark is restricted within each episode [Miyazaki 03b].

The classification of rules has meaning on the class where there is no type 2 confusion. In the first, we treat the class where there is no type 2 confusion since PARP cannot treat the type 2 confusion environments. After that, we extend it to the type 2 confusion environments.

### 3.2 There is no type 2 confusion

When there are  $p'$  ( $p' \leq p$ ) kinds of penalties and  $R_i$  ( $i = 1, 2, \dots, r$ ), the best policy is to get  $R_i$  and to avoid all  $p'$  penalties. It means that we select a rational rule about  $R_i$  that is derived by  $\mathbf{RPM}_i$  except for all penalty rules by  $PRJ_i(P_1)$ ,  $PRJ_i(P_2)$  and  $PRJ_i(P_{p'})$  as shown in figure 5.

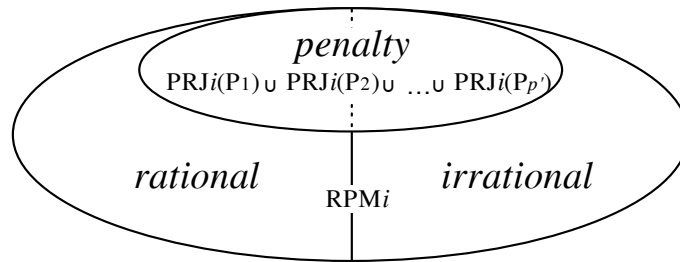


Figure 5. The relation between rational, irrational and penalty rules.

If we cannot select the best policy, we should give up to get  $R_i$  or to avoid  $P_j$  that is the less important in the priority. For example, we consider the case that the priority  $A(P1) > G(R1) > A(P2)$  is given by the designer. In the first, the agent selects a rational rule as referring to  $\mathbf{RPM}_1$  in non-penalty rules of  $PRJ_1(P1)$  and  $PRJ_1(P2)$ . If the agent cannot select a rational rule, it gives up to avoid  $P2$  since it is the less important in  $A(P1) > G(R1) > A(P2)$ . In this case, it selects a rational rule as referring to  $\mathbf{RPM}_1$  in non-penalty rules of  $PRJ_1(P1)$ . Furthermore, if it cannot select any rule, it gives up to get  $R1$ . In this case, it selects a rule only considering to avoid  $P1$

Last, we consider the case that there are two or more than rational rules in same sensory input. If  $R_i$

remains in the priority, the agent selects a rule based on  $RPM_i$ . Otherwise, the agent selects an arbitrary rule that can select in the sensory input. If there is no rule that can select in the sensory input, the agent avoids a penalty stochastically as same as original PARP.

### 3.3 Extend to the type 2 confusion environments

We can get the policy by the method described in section 3.2. We preserve  $R_i$  or  $P_j$  that is rejected by getting the policy. We do not need to change the policy, if the priority about  $R_i$  or  $P_j$  that is not rejected by getting the policy has been maintained even if there is a type 2 confusion. Otherwise, we should investigate whether there is a type 2 confusion or not.

In this paper, we realize it by borrowing an idea of PS-r\* [Miyazaki 03a]. In the first, we memory the descendant states that can be transited by each rule in executing the policy. We call it  $ST(\text{the policy})$ . Furthermore, we call the memory of the descendant states by random walk  $ST(\text{rand})$ . We compare  $ST(\text{rand})$  and  $ST(\text{the policy})$  by *chi-square goodness-of-fit test*. We select an action by random selection on the state where the result of the comparison is coincident and recalculate PRJ.

We continue the above process until we cannot find a state where there is no type 2 confusion. The worst policy is random walk in the environment that does not have the state where there is no type 2 confusion. It is the same property as PS-r\*.

## 4. NUMERICAL EXAMPLES

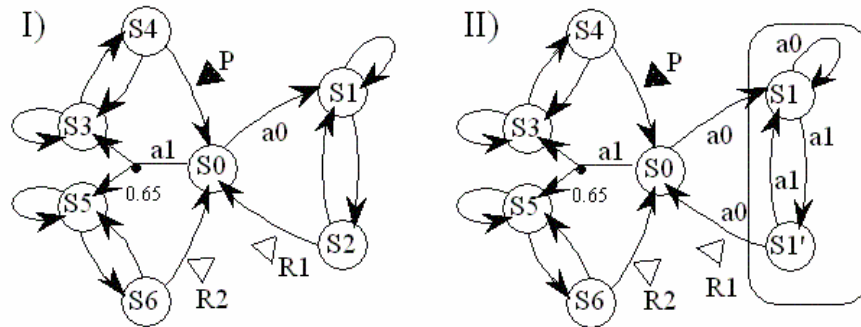


Figure 6. An environment of 2 rewards (R1,R2) and a penalty (P).

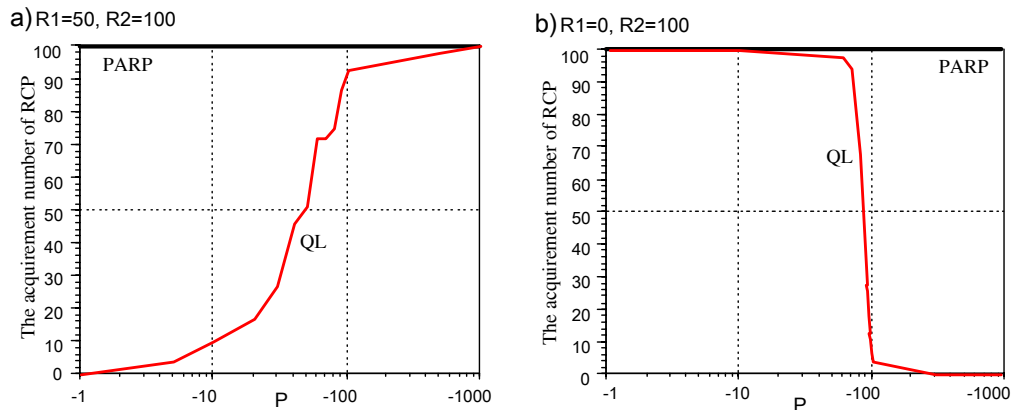


Figure 7. The acquirement number of RCP in figure 6.

In the first, we compare the proposed method with QL and original PARP in the environment of figure 6 I). It has two rewards (R1 and R2) and a penalty (P1). Figure 7 a) and b) are the results of QL and PARP in ( $R_1 = 50, R_2 = 100, -1 \leq P \leq -1000$ ) and ( $R_1 = 0, R_2 = 100, -1 \leq P \leq -1000$ ), respectively. In this environment, if the agent selects the action  $a_0$  in all sensory inputs, it can get R1 without P1. It is the only penalty avoiding rational policy. We call the policy the *right circle policy* (RCP). On the other hand, if the

agent selects the action  $a_1$  in all sensory inputs, it gets  $R_2$  and  $P_1$ . It is called the *left circle policy* (LCP). The vertical axes of both figures show the acquirement number of RCP in different 100 trials.

In this environment, we can consider four type priorities;  $A(P_1) > G(R_1)$ ,  $A(P_1) > G(R_2)$ ,  $G(R_1) > A(P_1)$  and  $G(R_2) > A(P_1)$ . Only the priority  $G(R_2) > A(P_1)$  requires LCP. The other priorities require RCP. PARP learns always RCP. Therefore PARP cannot fit on the priority  $G(R_2) > A(P_1)$ . Though QL can learn RCP or LCP, it is difficult to design appropriate  $G(R_1)$ ,  $A(P_1)$  and  $A(P_2)$  values for each priority. For example, if we set ( $R_1 = 50$ ,  $R_2 = 100$ ,  $P = -1000$ ), QL always learns RCP even if it should learn LCP in  $G(R_2) > A(P_1)$ . In this case, if we set ( $R_1 = 50$ ,  $R_2 = 100$ ,  $P = -1$ ), QL can always learn LCP. However, it is very difficult to find the combination of reward and penalty values before learning. On the other hand, our proposed method can always learn an appropriate policy for each priority. The average number of steps to learn the policy was 133.11 (S.D.=85.36) in different 100 trials.

Next we apply to the environment of figure 6 II) that has a type 2 confusion in the sensory inputs  $S_1$  and  $S_1'$  since they are sensed as the same sensory input. In this case, we can estimate the number of actions to learn it by the following equation,

$$n = \frac{1}{2} \left( \frac{u(\alpha) + u(\beta)}{\sin^{-1} \sqrt{\pi_1} - \sin^{-1} \sqrt{\pi_2}} \right)^2,$$

where  $\alpha$  is the significant level,  $1-\beta$  is the power of test and  $\pi$  is the probability to test the difference by *chi-square goodness-of fit test*. If we set that the maximum value of the difference of between probabilities is 0.05,  $n=2095.5$ . On the other hand, we can learn the environment by average 134.11 actions and S.D. is 85.36 on 100 different trials. It means that our method is useful on the environment where there is a type 2 confusion.

## 5. CONCLUSIONS

The most reinforcement learning (RL) systems treat a reward and a penalty on the same weights. There is a problem how to decide an appropriate these values. We know PARP as a RL system to overcome the problem. However PARP cannot treat more than one kind of rewards and penalties. Previously, we extend PARP to such the multiple rewards and penalties environments in which there is no type 2 confusion that is a broader class than a Markov Decision Process. In this paper, we extend it to the class where there is a type 2 confusion that is the same as a Partially Observable Markov Decision Process. In the future, we should apply it on the multi-agent environments, and so on.

## REFERENCES

- [Miyazaki 04] Miyazaki, K. & Kobayashi, S. Reinforcement Learning in Multiple Rewards and Penalties Environments, *Joint 2nd International Conference on Soft Computing and Intelligent Systems and 5th International Symposium on Advanced Intelligent Systems*, 2004.
- [Miyazaki 03a] Miyazaki, K. & Kobayashi, S. An Extension of Profit Sharing to Partially Observable Markov Decision Process: Proposition of PS-r\* and its Evaluation, *Journal of the Japanese Society for Artificial Intelligence*, Vol.18, No.5, pp.286-296, 2003. (in Japanese).
- [Miyazaki 03b] Miyazaki, K., Terada, T. & Kobayashi, H.: Generating Cooperative Behavior by Multi-Agent Profit Sharing on the Soccer Game, *4th International Symposium on Advanced Intelligent Systems*, pp.166-169, 2003.
- [Miyazaki 02] Miyazaki, K. & Kobayashi, S. Comparison with profit sharing and random selection in POMDPs. Joint first International Conference on Soft Computing and Intelligent Systems, pp.206-211, 2002.
- [Miyazaki 00] Miyazaki, K. & Kobayashi, S. Reinforcement Learning for Penalty Avoiding Policy Making. *2000 IEEE International Conference on Systems, Man and Cybernetics*, pp.206-211, 2000.
- [Miyazaki 98] 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.
- [Miyazaki 94] Miyazaki, K. & Yamamura, M. and Kobayashi, S. On the Rationality of Profit Sharing in Reinforcement Learning, *3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing*, pp.285-288, 1994.
- [Sutton 88] Sutton, R. S. Learning to predict by the methods of temporal difference, *Machine Learning*, Vol.3, pp.9-44, 1998.
- [Watkins 92] Watkins, C. J. H., & Dayan, P.: Technical note: Q-learning, *Machine Learning*, Vol.8, pp.55-68, 1992.