Proposal and Evaluation of Detour Path Suppression Method in PS Reinforcement Learning

Daisuke SHIRAISHI *, Kazuteru MIYAZAKI **, and Hiroaki KOBAYASHI *

Abstract: Profit sharing (PS) is well known as a kind of reinforcement learning. In a PS method, a reward is generally distributed with a geometrically decreasing function, and the common ratio of the function is called a discount rate. A large discount rate increases the learning speed, but a non-optimal policy may be learned. On the other hand, a small discount rate improves the performance of the policy, but the learning may not proceed smoothly because of the shallow learning depth. In this paper, in order to cope with these problems, we propose a method that reinforces both the detour path and the non-detour path with different discount rates. Finally, this method is applied to a maze problem and an altruistic multi-agent environment to confirm its effectiveness.

Key Words: reinforcement learning, profit sharing, detour path, multi-agent environment

1. Introduction

Recently, robots are developing to the human living level. The advances are appealing since robots will be able to do various tasks for our labor shortage problem accompanying low birthrate and aging society.

However, the human living environment is filled with uncertainty. For example, the information obtained from the environment by the robot may contain many distractions, and it is also essential to cooperate with humans and other robots in real time. Generally, it is very difficult for us to design such behavior in advance; robots can give us a preview of the future. Robots are required to adapt to the environment based on their experience by themselves.

Reinforcement learning (RL) is a method for the robot to adapt to the environment. RL is a type of machine learning that adapts to the environment through trial and error [1]. RL is a method to acquire an appropriate policy with a reward. RL has the possibility to discover solutions beyond our imaginations. Q-learning (QL) [2], Sarsa [1],[3] and profit sharing (PS) [4]–[6] are representative RL methods; QL is guaranteed to acquire the optimal policy in Markov decision processes (MDPs) [2], but it does not so in the non-Markovian environment.

On the other hand, PS is an exploitation-oriented learning method and thus aims to be not an optimal but a rational policy by strongly enhancing learners’ experience [7]. PS has some rationality even in a kind of non-Markovian environment [5],[6], and its effectiveness can also be seen in multi-agent environments [8]–[11]. In addition to these papers, many studies exist on PS [9],[12]–[18].

In the PS method, generally, the learning progresses by distributing rewards using a geometric decreasing function. The common ratio of this function is called a discount rate. A large discount rate increases learning speed, but unsuitable rules may be learned. On the other hand, a small discount rate does not cause the above problem but often reduces the learning speed because distributed rewards decrease so rapidly so that learning does not progress in a long episode.

In this paper, in order to solve this problem, a new reward distribution method is proposed. In this method, we first judge whether there is a detour path or not in the episode, and then we distribute the reward separately to detour and non-detour paths to suppress the reinforcement of detour paths and prioritize the reinforcement of non-detour path.

In the PS method, generally, the learning progresses by distributing rewards using a geometric decreasing function. The common ratio of this function is called a discount rate. A large discount rate increases learning speed, but unsuitable rules may be learned. On the other hand, a small discount rate does not cause the above problem but often reduces the learning speed because distributed rewards decrease so rapidly so that learning does not progress in a long episode.

In this paper, in order to solve this problem, a new reward distribution method is proposed. In this method, we first judge whether there is a detour path or not in the episode, and then we distribute the reward separately to detour and non-detour paths to suppress the reinforcement of detour paths and prioritize the reinforcement of non-detour path.

After important terms are explained in Section 2, we describe the rationality theorem [5],[6] and an extended rationality theorem [19] about PS in Section 3. We then propose a detour path suppression method (DPSM) that reinforces detour paths and a non-detour path with different discount rates, respectively, in Section 4. In Sections 5 and 6, the effectiveness of the proposed DPSM is shown through maze problems and a multi-agent environment. The last section, Section 7, summarizes the paper and speculate on future work.

2. Reinforcement Learning

2.1 Environment in Reinforcement Learning

Let us consider an agent in an unknown environment. After perceiving sensory input from the environment, the agent selects and executes an action. Time is discretized by one input-action cycle called a step.

The agent perceives the state input \( s_t \) from the environment as the observation \( o_t \) at the time \( t \). If there is no restriction on the observability of the agent, i.e., \( o_t = s_t \), the environment is called a complete perception environment. If there is some restriction on the observability, i.e., \( o_t \neq s_t \), the environment is called an incomplete perception environment.

The agent decides the action \( a_t \) based on the state \( s_t \) and the evaluation value (defined later). The pair of a state \( s_t \) and an
action \(a_t\) selected in the state is called a \textit{rule} and described as \textit{rule}(\(s_t, a_t\)) or simply \((s_t, a_t)\). The agent changes the state \(s_t\) to the next state \(s_{t+1}\) by applying the rule. If the agent receives a reward \(r_t\) at the state \(s_{t+1}\), then the evaluation value is updated based on the reward. In this paper, so we assume that no penalty corresponds to a negative reward, and the number of types of a reward is one. This was the same assumption in the previous PS papers [4]–[6]. If the probability of the state transition in the environment depends only on the state and the action, this state transition has Markov property, and the process is designated as a Markov decision processes (MDP).

A function that maps states to actions is called a \textit{policy}. If the reward acquisition expected value of a policy is positive, that policy is called a \textit{rational policy}; an \textit{optimal policy} is a policy that can maximize the amount of rewards. Reinforcement learning aims to acquire a policy that gives maximum rewards with the least action.

A series of rules that begins with a reward state or an initial state and ends with the next reward state is called an \textit{epis ode}. If one episode contains rules of the same state but is paired with different actions, the partial series from one state to the next is called a \textit{detour path}, and a rule in a detour path is called a \textit{detour rule}. For example, the episode \(((S1, 1), (S2, 2), (S3, 3), (S4, 4), (S2, 2))\) shown in Fig. 1 has a detour path \(((S2, 2), (S3, 3), (S4, 4))\) because different actions, \(\gamma\) and \(\beta\), were selected in the same state \(S2\), and \((S2, 2), (S3, 3)\) and \((S4, 4)\) are detour rules. A rule always existing in a detour path is called an \textit{ineffective rule}, and otherwise called an \textit{effective rule}. Note that a detour rule is not always an ineffective rule, but an ineffective rule is always a detour rule.

![Fig. 1 Example of detour path and detour rules.](image)

### 2.2 Reinforcement Learning Methods

#### 2.2.1 Profit sharing

PS learns a rational policy by propagating a reward backward in an episode when a reward is given. If a reward \(R\) is given at time \(N+1\), and the corresponding episode is \(((s_1, a_1), (s_2, a_2), \ldots, (s_t, a_t), \ldots, (s_N, a_N))\), then the amount of rewards (the evaluation value) of the rule \((s_t, a_t), Q(s_t, a_t)\), will be updated as follows:

\[
Q(s_t, a_t) \leftarrow Q(s_t, a_t) + f(N-t), \quad t = N, N-1, \ldots, 1, \quad (1)
\]

where \(f(0) = R\). The function \(f(k)\) to propagate a reward is known as a \textit{reinforcement function}. In this paper, we use a geometrically descending function:

\[
f(i) = \lambda^i R, \quad (2)
\]

where \(\lambda (0 < \lambda < 1)\) is the \textit{discount rate}.

#### 2.2.2 Q-learning

Each time the agent selects an action, QL learns by updating the \textit{Q-value}, i.e., a function that represents the evaluation value of the rule. When the agent selects the action \(a_t\) in the state \(s_t\), receives a reward \(r_t\), and transits to the next state \(s_{t+1}\), \(Q(s_t, a_t)\), the Q-value of \(r(s_t, a_t)\), is updated by the following equation:

\[
Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a'))\), \quad (3)
\]

where \(\alpha\) is the \textit{learning rate} and \(\gamma\) is the \textit{discount rate} \((0 < \alpha, \gamma < 1)\).

QL is guaranteed to learn the optimal policy in the Markov decision processes; it learns the optimal policy if the agent selects an action that has the maximum Q-value in the state after the Q-value has converged by reducing the learning rate \(\alpha\) according to a previously proposed theorem [2]. However, this theorem does not guarantee any rationality before the Q-value converges.

In this paper, we focus on PS, and our proposed method is compared with QL.

#### 2.3 Rule Selection Methods

In RL, the rule selection method is also an important factor determining the character of learning. There are various rule selection methods, so here are typical rule selection methods.

##### 2.3.1 Random selection

Random selection is a method to randomly select the rule with equal probability from available rules without considering their evaluation values.

##### 2.3.2 Greedy method

The so-called greedy selection method chooses the rule with the highest evaluation value among the candidate rules. If the evaluation value of each rule is an appropriate value, it is possible to select the optimal policy. However, this method is not effective if the reliability of evaluation values of rules is low. Therefore, it is better to use this method after learning has converged.

##### 2.3.3 \(\varepsilon\)-greedy method

An \(\varepsilon\)-greedy method performs random selection with the probability \(\varepsilon\) \((0 < \varepsilon < 1)\) and the greedy method with the probability \(1 - \varepsilon\). For large \(\varepsilon\), the probability of selecting a random rule increases, and for small \(\varepsilon\), the probability of selecting the rule with the highest evaluation value increases.

It is effective to use a large \(\varepsilon\) in the initial learning and reduce \(\varepsilon\) closer to 0 as learning progresses.

##### 2.3.4 Soft-max method

A Soft-max method selects a rule according to the ratio of evaluation values. There are several types, but we use the following \textit{roulette selection} in this paper.

In the roulette selection method, selection probability is determined according to the ratio of the evaluation value. The probability \(P(a|s)\) of selecting an action \(a\) in a state \(s\) is defined as follows:

\[
P(a|s) = \frac{Q(s, a)}{\sum_{a_j \in \mathcal{A}} Q(s, a_j)}, \quad (4)
\]

where \(N_{act}\) is the number of rules available in this state. Thus, the higher the evaluation value is, the higher the selection probability becomes.

These are well-known rule selection methods. Since each
3. How to Suppress No-Reward Acquisition Rules

In a case where there are ineffective and effective rules in the same state, the ineffective rules should be suppressed, and the effective rules should be preferentially enhanced. As shown in Section 2.2.1, the following theorem gives a reinforcement function that can suppress reinforcement of ineffective rules.

### 3.1 Rationality Theorem \[5],[6]\n
If the reinforcement function satisfies the following condition, ineffective rules will not be enhanced over effective rules.

\[
W \sum_{j=i}^{W} f(j) < f(i-1) \quad (\forall i = 1, 2, ..., W),
\]

where \(W\) is the maximum episode length and \(L\) is the maximum number of effective rules in the state. A geometric decreasing function satisfies the condition, if

\[
f(i) = \lambda f(i-1) \quad (\forall i = 1, 2, ..., W)
\]

where \(\lambda \leq 1/(L+1)\).

The value of \(L\) cannot be known in advance, but it is sufficient to set the maximum number of rules available in each state minus 1. The simplest and representative geometric decreasing function is shown in (2).

### 3.2 Extended Rationality Theorem \[19]\n
There is another theorem \[19\] that suppresses detour rules. Though ineffective rules cannot be judged from one episode, detour rules can be so judged. Furthermore, if we suppress detour rules, ineffective rule suppression will be performed efficiently because ineffective rules are contained in detour rules. The extended rationality theorem assures that the reinforcement function satisfying the following condition suppresses the detour rules:

\[
\begin{align*}
L \sum_{j=i}^{W} f(j) \cdot d(j) &< f(i-1) & \text{if } d(i) = 1, \\
f(i) &\leq f(i-1) & \text{if } d(i) = 0, \\
& \quad i = 1, 2, ..., W,
\end{align*}
\]

where \(W\) is the maximum episode length, \(L\) is the maximum number of effective rules in the state, and

\[
d(i) = \begin{cases} 
1 & \text{if the rule is a detour rule}, \\
0 & \text{if the rule is not a detour rule}.
\end{cases}
\]

An example of functions satisfying the extended rationality theorem is as follows:

\[
\begin{align*}
f(i) &= \lambda f(i-1) & \text{if } d(i) = 1, \\
f(i) &\leq f(i-1) & \text{if } d(i) = 0, \\
& \quad i = 1, 2, ..., W.
\end{align*}
\]

The value of \(\lambda\) is also set as \(\lambda \leq 1/(L+1)\) for the rationality theorem.

4. Proposal of Detour Paths Suppression Method

#### 4.1 Detour Path Suppression Method

In this paper, we suppress detour rules in the same way as the extended rationality theorem does. In this proposed method, we use different reinforcement functions for detour rules and the other rules in an episode, respectively. We show an example of reward distribution in Fig. 2, where \(f_0\) and \(f_1\) are reinforcement functions for non-detour rules and detour rules, respectively.

We classify rules in the episode (the first row of Fig. 2) into detour rules (the third row) and other rules (the second row) before reward distribution. Since the latter rules contain no detour rules, there is no ineffective rule. Hence, an arbitrary non-monotonically-increasing function can be used as the reinforcement function. Here, the discount rate of this function is called a non-detour discount rate and expressed as \(\lambda_0\). For example, the reward \(R\) is distributed to non-detour rules \(S1, S2, S3, S4, S5\) with the discount rate \(\lambda_0\). In principle, \(\lambda_0\) is required to be \(0 < \lambda_0 \leq 1\), and the larger \(\lambda_0\) is used, the faster the learning speed becomes. But if we use \(\lambda_0 = 1\) and there are a plurality of effective rules, makes it difficult to select a rule that can most efficiently obtain a reward. In this paper, we conduct experiments for \(\lambda_0 = 1\) from the viewpoint of learning speed but note that \(\lambda_0\) is a parameter related to the tradeoff between reward acquisition and learning speed.

For detour rules, the reward allotted to the rule immediately after the detour path is distributed to the rules in the detour path using a reinforcement function that satisfies the rationality theorem. The discount rate of this function is called a detour discount rate and is represented by \(\lambda_1\). For example, for the detour rules \(S2, S3, S4\), the rule immediately after the detour path is \(S2\), and the allotted reward \(\lambda_1^2 R\) is distributed to the rules \(S2, S3, S4\) with the discount rate \(\lambda_1\). Distributing rewards using this method, we can suppress the reward given to the detour paths and acquire a rational policy.

Figure 3 shows the transition of decrease in the reward value of the proposed method and the methods based on the rationality theorem and the extended rationality theorem. The horizontal axis represents the action distance from the last rule on the episode, and the vertical axis represents the decrease rate of distributed rewards. From the figure, it can be seen that the proposed method has a slower decrease in reward value and a longer learning horizon than the other methods do.

4.2 Detour Paths Discrimination Procedure

To apply the above method, first of all we have to find out detour paths in an episode. Here, we describe a method to determine detour paths in an episode. From the definition of the detour path, a detour rule in an episode can be seen in the following algorithm:

1. \(i = 1, j = N, d(k) = 0 \quad (k = 1, 2, ..., N)\), where \(N\) is the episode length.
2. if \(s_i = s_j\) and \(a_i \neq a_j\), then go to 4.
3. \(j = 1\). If \(j > i\) then to 2 else, then go to 5.
4. \(d(k) = 1 \quad (k = i, i + 1, ..., j - 1)\).
5. \(i = i + 1, j = N\). If \(i > W\) then go to 2, else end loop.
This algorithm can discriminate as to whether the $i$th rule is in the detour path or not; if $d(i) = 1$, the $i$th rule is a detour rule, and if $d(i) = 0$, it is a non-detour rule. However, this algorithm cannot work when there are multiple detour paths and overlap one another. We consider the problem in the next section.

4.2.1 Consideration of duplicate detour paths

Figure 4 shows an example in which there are two detour paths $\{(S_2, \gamma), (S_3, \alpha), (S_4, \beta)\}$ and $\{(S_4, \beta), (S_2, \beta)\}$ that overlap each other. If the above detour path discrimination algorithm is used, all the rules of both detour paths will be judged as detour rules. However, if all these rules are discarded, no rule chain exists from $(S_1, \alpha)$ to $(S_4, \gamma)$. And if we discard only one of the two detour paths, there will be no detour path in the remaining episode.

First, if we consider the later path $\{(S_4, \beta), (S_2, \beta)\}$ as a detour, an effective rule in a certain state, for example, $(S_2, \gamma)$, may be farther from the target state than a detour rule competing with it, for example, $(S_2, \beta)$. Depending on the value of the discount rate, the reward value assigned to the detour rule may be larger than the reward value given to the effective rule. Therefore, this procedure is not suitable for learning.

Next, if we consider the earlier path $\{(S_2, \gamma), (S_3, \alpha), (S_4, \beta)\}$ as a detour, no effective rule in a particular state is farther from the target state than detour rule competing with it, so this procedure is suitable for learning.

Therefore, in this paper, we decide only the earliest detour path among duplicate ones as a detour path.

4.2.2 Consideration of multiple detour path

In Fig. 5, we show an example in which there are two detour paths and they are multiple. Even in this case, both are detour paths by definition. A detour path included in another is called a multiple detour path: If a larger detour path is determined, a multiple detour path contained in it is also fixed. At first glance, this seems to be reasonable. However, in an environment with autoregressive rules, it creates a problem. A simple example of an environment with an autoregressive rule is shown in Fig. 6. If the episode shown in Fig. 7 is given, the autoregressive rule of $S_3$ will be very strongly enhanced because rewards will be added to the rule many times. Note that continuing selection of an autoregressive rule will form a multiple detour path, and it is highly probable that the rules in the multiple detour path will be rules that do not contribute to reward acquisition. Therefore, the rules in multiple detour paths will not receive any rewards.

Thus, in this paper, we use the following algorithm as the detour path discrimination method.

4.3 Extended Detour Paths Discrimination Procedure

The difference from the above algorithm is discrimination of detour paths in an episode with duplicate and/or multiple detour paths.

1. $i = 1, j = N, d(k) = 0 \ (k = 1, 2, \ldots, N)$, where $N$ is the episode length.
2. if $d(i) > 1$ then go to 6.
3. if $s_i = s_j$ and $a_i \neq a_j$ then go to 5.
4. $j = 1$. If $j > i$ then to 3, otherwise go to 6.
5. $d(k) = 1$ ($k = i, i + 1, \ldots, j - 1$).
The discount rate (d.r.) of PS was set to 0.95 in preliminary experiments. The discount rate (d.r.) of PS with DPSM for non-detour rules is 0.95 that has an advantage in terms of learning speed and is the same as the discount rate of PS with DPSM for non-detour rules. The initial evaluation value is set to 10, and the reward when achieving that goal is 100. The action selector for the learning stage is the softmax method for PS-based methods, and the $\epsilon$-greedy method ($\epsilon = 0.2$) for QL. The action selector method for the evaluation stage is the greedy method for all methods. The upper limit number of action selections is 2000, so the agent continues to act until it reaches the goal – or the action selection number exceeds 2000.

### Table 1: Experiment parameters.

| Method         | discount rate (d.r.) | non-detour d.r. | detour d.r. |
|----------------|----------------------|-----------------|-------------|
| PS             | 0.95                 |                 |             |
| PS with E-RT   |                      | 1               | 0.20        |
| PS with DPSM   | 0.95                 | 0.20            |             |
| learning rate  |                      |                 | d.r.        |
| QL             | 0.2                  | 0.99            |             |

### 5.2 Results and Discussion

The experiment was carried out 30 times with different random seeds. Figures 9 to 14 show results of the four methods for each size maze. Among these, Figs. 9, 11 and 13 show the results of the learning stage for each action selector. Figures 10, 12 and 14 are for the evaluation of previously learned policies. The horizontal axis represents the learning number of times, and the vertical axis represents the number of actions in one trial.

The proposed method features high learning ability in mazes of all sizes. The learning speed for the size of $45 \times 45$ is considerably faster than the other methods. In PS and extended rationality PS, reward values distributed to rules of states near the goal. The sizes of the mazes are $25 \times 25$, $35 \times 35$, and $45 \times 45$ with no outer wall.

In this experiment, the PS method, the PS method satisfying extended rationality theorem (PS with E-RT), and the proposed method (PS with DPSM) and QL are all used. Table 1 shows the parameters used in the experiments of each method. They were decided by preliminary experiments. Especially in PS, for long episodes such as the maze environment used in this experiment, the function satisfying the rationality theorem takes a long time to learn near the start point S. Therefore the discount rate of PS was set to 0.95 that had achieved the fastest learning speed in preliminary experiments. The discount rate (d.r.) for the detour rule is 0.2 that satisfies the rationality theorem.

Fig. 6 Environment with recursive rule.

Fig. 8 An example of maze ($35 \times 35$) with the shortest route.

To create a maze, a hole drilling method http://studio.s1.xrea.com/prog/maze.html of a maze creation algorithm was used. After that, the walls are randomly removed.

---

2 To create a maze, a hole drilling method http://studio.s1.xrea.com/prog/maze.html of a maze creation algorithm was used. After that, the walls are randomly removed.

3 Though the upper limit that satisfies the theorem is 0.25, we have confirmed that there is not much difference in the experimental results between 0.2 and 0.25.
start decrease in size when the episode becomes longer, which leads to slow progress in learning. In QL, the identification of the environment requires more time, and the advance of learning becomes slower as the environment becomes larger. On the other hand, learning can progress regardless of the length of the episode in the proposed method since non-detour rules that contribute to the reward directly have been given priority for reinforcement. These results show that the proposed method has higher learning efficiency in a single agent maze problem.

Table 2 shows the average value of the number of trials to the target state and the standard deviation (in parentheses) at the 5000th learning step for the greedy methods. We checked the difference between each item with $t$-test at the significance level of 5%, and then it was shown that there was a significant difference in items except for PS ERT and PS DPSM in the 25 $\times$ 25 maze.

Table:<br>

| Maze Size | PS   | PS ERT | PS DPSM | QL    |
|-----------|------|--------|---------|-------|
| 25 $\times$ 25 | 62.7 (1.31) | 66.7 (3.11) | 67.6 (4.36) | 64.8 (2.71) |
| 35 $\times$ 35 | 76.3 (2.17) | 82.2 (5.02) | 85.5 (5.67) | 74.6 (1.28) |
| 45 $\times$ 45 | 116.9 (2.46) | 20.6 (4.62) | 26.1 (5.72) | 2000.0 (0.00) |

In the QL of the 45 $\times$ 45 maze, learning did not finish, and in the QL of the 25 $\times$ 25 maze, it may be partially in a local solution. (Extending learning number of times may achieve the optimality.) With regard to PS, there was a tendency for more optimal policies to be obtained in the order of PS DPSM, PS ERT, and PS. For PS, the effect of using a discount rate of 0.95 that does not satisfy the theorem is also conceivable. In this paper, we focused on learning speed, but in the future, we will study further on the tradeoff between optimality and learning speed.

6. Evaluation of DPSM in Altruistic Multi-Agent Environment

6.1 Setting

We use the environment shown in the paper [20] to verify the effectiveness of DPSM in a multi-agent environment. The agents aim to obtain rewards in the environment simultaneously if possible in an environment where more than one agent performs learning at the same time. DPSM is compared with the proposed method in the paper [20] (see Appendix for details) by using the environment shown in Fig. 15.
Figure 15 corresponds to the case of three agents in environments. Each agent is located in one of the squares and can be perceived vertically and horizontally from neighboring squares on the agent. The thick line is a wall that prevents the perception of the square from the other side of the wall. As a result, all agents at hatched squares perceive the same input. There are three possible perceptions: [there is nothing, there are other agents, there is a wall] on each square. That is, there is an incomplete perception environment; one agent cannot distinguish the other agents. Each agent (i = 1, 2, 3) is located in S_i at the time of starting the learning. An agent selects an action from [up, down, left, right] movement after obtaining a sensory input. Transition to the wall is not allowed, so the agent remains in the original square, and multiple agents can occupy the same square. Each agent (i = 1, 2, 3) aims to move to the target state G_i. If an agent i transits to the target state G_i, a reward is given to all agents, and that agent is returned to the square of the initial position S_i. Each square except for the hatched squares and the target states is perceived as a different input. As a result, the agent distinguishes 22 kinds of squares. The number of possible sensory inputs are 704 = 22 × 2^5 since other agents are not distinguished even if they exist in the square.

As an action selection method, we use the ε-roulette strategy where the upper limit of the number of action selection times is decided; the upper limit of the number of times is 100,000 in this paper, which calculated by 

$$\text{R} = 10^3 \frac{1}{\text{R}}$$

generating a random number between 0.0 to 1.0. If the value of ε is zero or less, we set ε = 0.0. If the value of ε is larger than the random number, we use roulette selection for evaluation values. Otherwise, we use random selection. For this ε, the ratio of roulette selection will exceed one of random selection after 25,000 actions, and only roulette selection will be used after 50,000 actions until 100,000 actions. The action selection will be executed in the order of agents 1, 2, and 3. When all three agents finish selecting an action, the number of selected actions is increased by one.

If there is only one agent in Fig. 15, the perceptual aliasing problem [21] will occur, since an agent in the hatched squares perceives the same sensory input. On the other hand, in multi-agent learning, it may be possible to reduce the perceptual aliasing problem if the other agent moves properly; such behavior is likely to be derived by an indirect reward, and the method using an indirect reward has been previously proposed in the paper [20]. In this paper, we evaluate the performance of DPSM for this method changing the reward value while the initial evaluation value is kept to 10.0.

### 6.2 Results and Discussion

The results on the case of three agents are shown in Table 3 for the reward value R from 10^2 to 10^6. The experiment was carried out 100 times with different random seeds. The table shows the average value of the reward acquisition number of times and the standard deviation (in parentheses). The first row of the table corresponds to the results of the method proposed in the paper [20]. Subsequent rows are the results for DPSM instead of PS in the method of the paper [20].

The experiment in DPSM was conducted with λ_0 in the range of 0.3 to 0.9 in increments of 0.2 while λ_1 = 1/4 since the performance may change widely depending on the value of the non-detour discount rate λ_0. Subsequent rows are the results for DPSM instead of PS in the method of the paper [20]. The experiment in DPSM was conducted with λ_0 in the range of 0.3 to 0.9 in increments of 0.2 while λ_1 = 1/4 since the performance may change widely depending on the value of the non-detour discount rate λ_0.

The performance of the paper [20] significantly varies depending on the reward value R, because it uses PS as a basic learning method. In the case of PS, learning at the start point of the episode does not progress easily for a small reward, since the distributed reward decreases rapidly due to the discount rate. This means that it is difficult to learn desired behaviors for small reward values.

The method combined with DPSM, compared with the case of PS, results in behaviors in which the reward acquisition frequency of each agent is similar, and also the standard deviation

![Fig. 15 Three agents environment.](image-url)
is small, for each $\lambda$. In addition, the difference in the reward value seen in the method of the case of PS is reduced. This is considered to be that the speed of learning of DPSM worked effectively in multi-agent learning. There are cases where the method of the case of PS is larger if simply looking at the reward acquisition times. Note that we are aiming for the difference in the number of rewards acquisition times for each agent to be similar, so we should not limit our attention only to the magnitude of the reward acquisition times.

7. Conclusions

In this paper, in order to improve learning speed while satisfying rationality, we proposed a distribution method that could suppress detour rules. In addition, we also considered detour paths in cases of duplication. Furthermore, by adopting the concept of multiple detour paths, we created it a more robust discrimination method. We showed a discrimination algorithm that can distinguish a detour path from one episode. In order to compare the proposed method with the conventional method, numerical experiments were conducted using the maze problem, and the effectiveness of the proposed method was shown. By using the proposed method, learning efficiency improved compared with the conventional method.

In the future, we aim to combine DPSM with deep learning. We know DQNwithPS [22] that is combined with a deep Q-network (DQN) [23],[24] and PS, and DQNbyPS [25] that is a kind of DQN where an indirect reward is generated by PS. In general, PS can be replaced by DPSM. Therefore we propose DQNwithDPSM and DQNbyDPSM soon. Verification of the effectiveness of these methods is one of the most important tasks in the future, and we will also apply our method to multi-agent environments, such as those control for a team of quadrotors, the keepaway task [26],[27], and so on.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 17K00327.

References

[1] R.S. Sutton and A.G. Barto: Reinforcement learning: An introduction, A Bradford Book, MIT Press, 1998.
[2] C.J.C.H. Watkins and P. Dayan: Technical note: Q-learning, Machine Learning, Vol. 8, No. 3-4, pp. 55–68, 1992.
[3] G.A. Rummery and M. Niranjan: On-line Q-learning using connectionist systems, Technical Report CUED/F-INFENG/TR-166, 1994.
[4] J.J. Grefenstette: Credit assignment in rule discovery systems based on genetic algorithms, Machine Learning, Vol. 3, No. 2-3, pp. 225–245, 1988.
[5] K. Miyazaki, M. Yamamura, and S. Kobayashi: A theory of profit sharing in reinforcement learning, Transactions of the Japanese Society for Artificial Intelligence, Vol. 9, No. 4, pp. 580–587, 1994 (in Japanese).
[6] K. Miyazaki: Exploitation-oriented learning XoL: A new approach to machine learning based on trial-and-error searches, Multi-Agent Applications with Evolutionary Computational and Biologically Inspired Technologies: Intelligent Techniques for Ubiquity and Optimization, pp. 267–293, IGI Global, 2010.
[7] K. Miyazaki and S. Kobayashi: Exploitation-oriented learning PS-#, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 13, No. 6, pp. 624–630, 2009.
[8] S. Arai, K. Miyazaki, and S. Kobayashi: Generating cooperative behavior by multi-agent reinforcement learning, Proc. 6th European Workshop on Learning Robots, pp. 143–157, 1997.
[9] P. Zhou and B. Hong: A modular on-line profit sharing approach in multiagent domains, International Journal of Intelligent Technology, Vol. 1, No. 2, pp. 131–137, 2008.
[10] T. Watanabe, K. Miyazaki, and H. Kobayashi: A new improved penalty avoiding rational policy making algorithm for keep-away with continuous state spaces, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 13, No. 6, pp. 678–682, 2009.
[11] K. Miyazaki, K. Furukawa, and H. Kobayashi: Proposal of PSwithEFP and its evaluation in multi-agent reinforcement learning, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 21, No. 5, pp. 930–938, 2017.
[12] S. Kato and H. Matsuoka: A theory of profit sharing in dynamic environment, Proc. PRICAI ’00, pp. 115–124, 2000.
[13] H. Kawai, A. Ueno, and S. Tatsumi: A reinforcement learning method using reward acquisition efficiency for POMDP environment, Transaction of the Japanese Society of Artificial Intelligence, Vol. 23, No. 1, pp. 1–12, 2008 (in Japanese).
[14] M. Usui, H. Nakano, and A. Miyauchi: A profit sharing reinforcement learning method using a memory-based dynamic reinforcement function, Proc. 2008 International Symposium on Nonlinear Theory and its Applications (NOLTA’08), 2008.
[15] S. Koujaku, K. Watanabe, and H. Igarashi: Adaptive profit sharing reinforcement learning method for dynamic environment, Proc. 10th International Conference on Machine Learning and Applications and Workshops, pp. 462–465, 2011.
[16] Z. Su, J. Jiang, C. Liang, and G. Zhang: A distributed algorithm for parallel multi-task allocation based on profit sharing learning, Acta Automatica Sinica, Vol. 37, No. 7, pp. 865–872, 2011.
[17] K. Suzuki and S. Kato: Hybrid learning using profit sharing and genetic algorithm for partially observable Markov decision processes, Proc. 20th International Conference on Network-Based Information Systems (NBIS-2017), pp. 463–475, 2017.
[18] K. Suzuki and S. Kato: An evolutionary based approach to profit sharing for POMDP environments, Proc. 6th IEEE Global Conference on Consumer Electronics (GCCE 2017), pp. 735–737, 2017.
[19] W. Uemura and S. Tatsumi: About the reinforcement function for profit sharing, Transactions of the Japanese Society for Artificial Intelligence, Vol. 19, No. 4, pp. 197–203, 2004 (in Japanese).
[20] K. Miyazaki: A study of an indirect reward on multi-agent environments, Procedia Computer Science, Vol. 88, pp. 94–101, 2016.
[21] L. Chrisman: Reinforcement learning with perceptual aliasing: The perceptual distinctions approach, Proc. 10th National Conf. Artif. Intell., pp. 183–188, 1992.
[22] K. Miyazaki: Exploitation-oriented learning with deep learning: Introducing profit sharing to a deep Q-network, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 21, No. 5, pp. 849–855, 2017.
[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller: Playing Atari with deep reinforcement learning. Proc. NIPS Deep Learning Workshop 2013, arXiv preprint arXiv:1312.5602, 2013.
[24] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis: Human-level control through deep reinforcement learning, Nature, Vol. 518, pp. 529–533, 2015.
[25] K. Miyazaki, N. Kodama, and H. Kobayashi: Proposal and evaluation of an indirect reward assignment method for reinforcement learning by profit sharing, Proc. Intelligent Systems Conference 2018 (IntelliSys2018), pp. 187–200, 2018.
[26] P. Stone, R.S. Sutton, and G. Kuhlmann: Reinforcement learning toward RoboCup soccer keepaway, *Adaptive Behavior*, Vol. 13, No. 3, pp. 165–188, 2005.

[27] K. Miyazaki, H. Muraoka, and H. Kobayashi: Proposal of a propagation algorithm of the expected failure probability and the effectiveness on multi-agent environments, *Proc. SICE Annual Conference 2013*, pp. 1067–1072, 2013.

[28] R. Watanabe, F. Maruyama, Y. Nagata, and S. Tojo: Reinforcement learning method for multi-agent systems by generalized reciprocity, *Proc. 16th Annual Conference of Japanese Society for Artificial Intelligence*, pp. 1–3, 2002 (in Japanese).

[29] H. Ueda, T. Tanizawa, K. Takahashi, and T. Miyahara: Acquisition of reciprocal altruism in a multi-agent system, *The IEICE Transactions on Information Systems*, Vol. J88-D-I, No. 9, pp. 1278–1286, 2005 (in Japanese).

Appendix A Method Aiming at Uniformly Rewarding Each Agent in Multi-Agent Environment [20]

A.1 Basic Concept

We have described a specific learning method using multiple weight tables in which all agents aim to acquire a uniform reward. Each agent has a weight table in a number equal to the number of agents. Therefore, in selecting the action to be used, it will be necessary to clear some conflict resolutions among them.

As one conflict resolution method, though we can consider how to integrate multiple weight tables to one weight table, a method of selecting only one weight table is here adopted for an action selection from various multiple weight tables.

Section A.2 describes the method of determining the agent, called an altruistic agent, to perform the action selection using the weight tables that have been enhanced by the reward that was obtained by the previous agent, rather than the weight table that has been strengthened by the reward itself. Section A.3 describes the procedure after reward acquisition.

Learning of altruistic behavior is an essential issue in multi-agent learning, and is here reported in the papers [28],[29]. Though these papers pay some attention to the fact that altruistic behavior has been acquired as a result of learning, this paper focuses on providing a framework for learning efficiently altruistic behavior.

A.2 How to Determine the Altruistic Agent

Only when there is no previous altruistic agent, do we make a selection and determination of an altruistic agent. Though we can consider the cases where multiple agents perform the altruistic behavior for each other agent, this paper treats only one agent is a target for an altruistic agent in order to deal with a more simple case.

Specifically, determine the altruistic agent by the following method. First, find an agent with less than the top reward acquisition; this is the agent that can be used as a target of the altruistic behavior. If the minimum of the agent reward acquisition number is more than one, that agent will be the altruistic target.

All agents other than the agent that has previously been the altruistic target can become the agent to perform the altruistic behavior. In other words, all agents other than the agent in which the number of reward times is the minimum select an action using a weight table that is updated when the agent with the minimum number of reward acquisition times has been rewarded. Though it is conceivable that only one agent will have the maximum of the number of reward acquisition times to help the agent with the minimum number of rewards, this will be discussed in numerical experiments.

A.3 Procedure after Reward Acquisition

If the agent that carried out the action obtains a reward, all weights in the table that corresponds to the agent that had obtained a reward will be updated.

We use PS to update weight tables. Reward value is assigned using the same values for all agents. Reinforcement interval is the same as the value of the agent that had obtained a direct reward. Initialization of the episode is carried out only for the agent that had been obtained a direct reward.

Finally, if the agent that has been obtained a direct reward is the same as the altruistic target, altruistic behavior for the agent will be cleared.

Daisuke Shiraishi (Student Member)
He received his B.E. and M.E. degrees from Meiji University, Japan, in 2017 and 2019, respectively. His research interests include reinforcement learning.

Kazuteru Miyazaki (Member)
He received his B.E. degree from Meiji University, Japan, in 1991, and his M.E. and D.E. degrees from Tokyo Institute of Technology, Japan, in 1993 and 1996, respectively. In 2000, he joined National Institution for Academic Degrees and Quality Enhancement of Higher Education, where he is currently an Associate Professor of the Research Department. His research interests include reinforcement learning. He is a member of ISAI, IPSJ, JSME, RSJ, IIEE, and Japanese Association of Higher Education Research.

Hiroaki Kobayashi (Member)
He received the B.E., M.E., and Ph.D. degrees in precision engineering in 1971, 1973, and 1977, respectively, all from Kyoto University. From 1976 he was with the School of Science and Engineering at Meiji University, Japan. From 2018, he is a Professor emeritus of Meiji University. From 1988 to 1989, he was a postdoctoral fellow in Computer Information Science at the School of Engineering and Applied Science, University of Pennsylvania. His research interests are in the area of intelligent control of robotic systems including machine learning. He is a member of IEEE.