Improvement of Agent Learning for a Card Game Based on Multi-channel ART Networks

Kenta Nimoto*, Kenichi Takahashi, Michimasa Inaba
Graduate School of Information Sciences Hiroshima City University, Hiroshima, Japan.

* Corresponding author. Email: nimoto@cm.info.hiroshima-cu.ac.jp
Manuscript submitted July 15 10, 2015; accepted October 28, 2015.
doi: 10.17706/jcp.11.4.341-352

Abstract: The 3-channel fuzzy adaptive resonance theory network FALCON (Fusion Architecture for Learning, COgnition, and Navigation) is recognized as an effective method for combining reinforcement learning with state segmentation, in which learning targets the relationships between percepts, actions, and rewards. It has been shown that FALCON is effective in making a playing agent for the card game Hearts, although the agent was unable to beat a rule-based agent. This study proposes new learning methods for training FALCON, in order to create a stronger agent. Using percepts chosen with an emphasis on penalty points, actions to choose which specific card to play were selected, with feedback determined by the penalty points. The learning rate for updating weights was changed so that its value was determined based on the penalty points. Separate strategies were adopted for the lead player and the other players, and multiple FALCONs were employed to improve adaptation to the game situation. Experiments demonstrated that our approach is superior to a rule-based approach.

Key words: FALCON, hearts, multi-channel adaptive resonance theory networks, reinforcement learning.

1. Introduction

Many studies have been carried out in the field of artificial intelligence, and the technology has been shown to be useful [1]-[3]. Reinforcement learning has attracted particular attention as an effective method for learning and for deriving control rules as robots autonomously adapt themselves to the environment.

When we model a realistic environment with an agent system, the perceptual inputs that the agents perceive are presented in a variety of forms such as words, symbols, and continuous real numbers. To apply reinforcement learning efficiently in such environments, it is necessary to construct an appropriate state space of percepts. These can be roughly divided into two groups: off-line and on-line methods. Adaptive Resonance Theory (ART) is one of the on-line methods [4], [5]. In this study, we conducted learning experiments using FALCON (Fusion Architecture for Learning, COgnition, and Navigation) [6], [7], which is an extension of ART to machine learning. FALCON is an on-line method proposed by Ah-Hwee Tan. It can discretize a state space and learn action rules at the same time by simultaneously learning relations among percepts, actions, and rewards.

Two-person perfect-information games such as “go” and chess have attracted most attention in studies of artificial intelligence. Games played by more than three players, and including uncertain information, are called multi-player imperfect-information games. In general, imperfect-information games are more complex than perfect-information games, because the player lacks information about the other players, and stochastic factors are also involved. Studies have showed that FALCON is effective in both perfect- and

imperfect-information games [8], [9]. In this study, we report experiments using the card game Hearts to evaluate the effectiveness and applicability of FALCON to multi-player imperfect-information games.

Morishita has shown that FALCON can successfully learn to play Hearts [7]. However, it was unable to beat a rule-based agent [10]. Thus, we proposed a new learning method to make FALCON a stronger agent. Particularly, we chose percepts that emphasize penalty points, which significantly affect the outcome of the game, and chose a set of actions that can specifically identify a unique card to play. In addition, we used penalty points for feedback in learning and changed the learning rate in response to the penalty point. Our method also used multiple FALCONs. We experimentally examined whether these methods could improve the performance of an agent player or not.

2. Reinforcement Learning with FALCON

FALCON is an extended fuzzy ART machine learning method originally proposed by Ah-wee Tan [6]. In FALCON, an agent derives action rules by alternately executing a selecting phase (selecting an optimal action by percepts) and a learning phase (updating selecting action rules based on rewards given after the action is performed).

2.1. Architecture of FALCON

The architecture of FALCON is shown in Fig. 1. FALCON has an architecture in which a sensory field (SF), a motor field (MF), and a feedback field (FF) are connected at a cognitive field (CF). When an autonomous agent has M sensors, M neurons that receive inputs $s_i \in [0,1] (i = 1, ..., M)$ from the M sensors are built in at SF. The motor field receives a vector $A = (a_1, ..., a_K)$ corresponding to actions chosen by the agent, where K is the number of actions. Elements a_k s of the vector A are set as follows: $a_k = 1$ if action k is chosen, and $a_k = 0$ otherwise. FF has two neurons; one receives reward r and the other receives $1 - r$, where $r \in [0,1]$ is the reward that the agent receives.

The cognitive field CF has L neurons. Neuron n_j $(j = 1, ..., L)$ in CF is connected to neurons in SF, MF, and FF with weighting vectors $W^s{}_j = (w^s{}_{1j}, ..., w^s{}_{Mj})$, $W^m{}_j = (w^m{}_{1j}, ..., w^m{}_{Kj})$ and $W^f{}_j = (w^f{}_{1j}, w^f{}_{2j})$, respectively, where elements $w^y{}_{xj} \in [0,1]$ of weighting vectors indicate the degree of relations between n_j and neurons in SF, MF, and FF, respectively. FALCON learns the relations among percepts, actions, and rewards by updating weighting vectors $W^s{}_j$, $W^m{}_j$, and $W^f{}_j$, respectively.

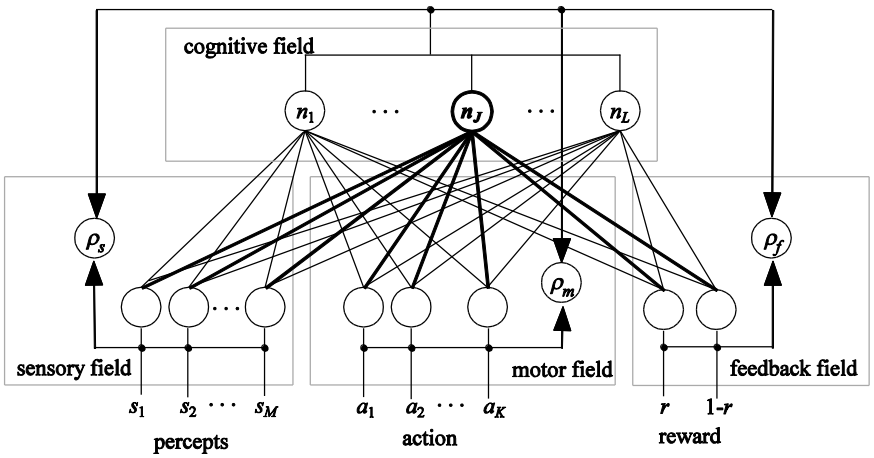


Fig. 1. The architecture of FALCON.

2.2. Choice and Learning of Actions

In the selecting phase, the choice strengths, which equal the degrees of relativity between the action and

weight vectors, are computed by

$$T_j = \gamma_s \frac{\left\| S \wedge W_j^s \right\|}{\alpha_s + \left\| W_j^s \right\|} + \gamma_m \frac{\left\| N \wedge W_j^m \right\|}{\alpha_m + \left\| W_j^m \right\|} + \gamma_f \frac{\left\| R \wedge W_j^f \right\|}{\alpha_f + \left\| W_j^f \right\|} \tag{1}$$

Operator $\wedge$ in (1) is the fuzzy AND operator which is defined as $Y \wedge Z = (\min(y_1, z_1), \ldots, \min(y_M, z_M))$ for M-dimensional vectors Y and Z. Norm $\|Y\| = \Sigma y_m$. Parameters γ_s, γ_m, and γ_f are nonnegative real numbers that satisfy $\gamma_s + \gamma_m + \gamma_f = 1$. Parameters α_s, α_m, and α_f are nonnegative real numbers. The neuron which has the largest choice strength shown in (1) is selected. Let the neuron denote n_J. An action is chosen according to the weighting vector $W^m{}_J$ connected to n_J; action k, whose weighting value is the largest, i.e., $maxarg_k (w^m{}_{kJ})$ in $W^m{}_J$, is usually chosen.

In the learning phase, if the feedback from its environment is positive, then the system learns to associate the percept vector S, the action vector A, and the reward vector R with one another; otherwise, it dissociates them. The similarity between S and $W^s{}_J$, the similarity between A and $W^m{}_J$, and the similarity between R and $W^f{}_J$ are computed by:

$$\frac{\left\| S \wedge W_J^s \right\|}{\left\| W_J^s \right\|} \leq \rho_s \tag{2}$$

$$\frac{\left\| A \wedge W_J^m \right\|}{\left\| W_J^m \right\|} \leq \rho_m \tag{3}$$

$$\frac{\left\| R \wedge w_J^f \right\|}{\left\| w_J^f \right\|} \leq \rho_f \tag{4}$$

If all three of (2), (3), and (4) hold, then the weighting vectors are updated according to (5), (6), and (7). ρ_s, ρ_m, and ρ_f are thresholds for similarities. β_s, β_m, and β_f are learning ratios that satisfy $0 \leq \beta_s, \beta_m, \beta_f \leq 1$. In general, the values of β_s, β_m, and β_f are 1, and we call this quick learning. When β_s, β_m, β_f <1, the weight vectors $W^s{}_J$, $W^m{}_J$, and $W^f{}_J$ are modified step by step, based on current values.

$$W_J^s = (1 - \beta_s) W_J^s + \beta_s (S \wedge W_J^s) \tag{5}$$

$$W_J^m = (1 - \beta_m) W_J^m + \beta_m (A \wedge W_J^m) \tag{6}$$

$$W_J^f = (1 - \beta_f) W_J^f + \beta_f (R \wedge W_J^f) \tag{7}$$

If any one of (2), (3), or (4) does not hold, then a category whose choice intensity is the largest out of the categories in which all of (2), (3), and (4) hold is chosen, and the weighting vectors of the category are updated by (5), (6), and (7). When there exists no category that satisfies the condition that all of (2), (3), and (4) hold, a new category is generated, and its weighting vectors are set as $W^s{}_{new} = S$, $W^m{}_{new} = A$, and $W^f{}_{new} = R$.

3. Hearts

3.1. Rules

Hearts is normally played by four players, using the standard 52-card deck. The higher card in the suit

wins, in the following strength order: *A* (high), *K*, *Q*, …, 4, 3, and 2. There is no superiority or inferiority between suits. Each player gets 13 cards and must play a card from his/her hand at his/her turn. A trick starts when a player plays a card, followed by each of the other players in a clockwise direction. A game is completed after 13 successive tricks have been played. In each trick, the card played by the first player is called a leading card, and that player is called the lead player.

The objective of Hearts is to hold the fewest penalty points at the completion of the game. Penalty points are based on the cards held, as follows: ♠*Q* is worth 13 points, and each other ♥ is 1 point.

In this research, two general rules of Hearts were not applied: (1) The exchange of cards before the 1st trick. (2) Shooting the moon and its variants.

3.2. Rule-Based Agent

Our learning agent played the game against rule-based agents [10] in order to compare their performance. The rule-based agent determined a playing card with rules extracted from gnome-hearts. When two random agents play the game against two rule-based agents, the average penalty ratio of the rule-based agent is 0.14 and that of the random agent is 0.36. The random agent determines a playing card randomly, following the game rules. Under these conditions, the agent whose penalty ratio is lower than 0.25 is stronger than the other agent. Thus, the rule-based agent wins an overwhelming victory against the random agent. The average penalty ratio is the ratio of penalty points obtained by each agent up to a total of 26 points in any one game.

4. Proposed Method

4.1. Determining Percepts

In FALCON, learning proceeds by associating percepts with actions. According to Tan [6], the complexity of learning is largely determined by the dimensions of the percepts; it is difficult for the system to learn efficiently when there are too many dimensions of percepts. Thus we selected seven kinds of percepts defined by 10 dimensions related to the penalty cards; ♥ and ♠*Q*. We list them in Table 1.

Table 1. Proposed Percepts

Percept	Definition of Value
Whether there is even one opponent player who has a stronger than the strongest in our agent's hand.	In case the condition is held, the value is 1; otherwise, it is 0.
Whether there is even one opponent player who has a weaker than the strongest in our agent's hand.	
Whether there is even one opponent player who has a stronger than the weakest in our agents' hand.	
Whether there is even one opponent player who has a weaker than the weakest in our agent's hand.	
Bit string to indicate the state of *Q*.	The value is 11 when our agent has the card, 10 when opposing players have it, 01 when that card is played in the current trick, and is 00 when it has been played.
Bit string to indicate the state of *K*.	
Bit string to indicate the state of *A*.	

4.2. Setting Actions

The actions for the learning agent were chosen so that only one playing card will be determined by the relative strength of the cards. We chose 29 actions based on the rules used by the rule-based agent. We list

them in Table 2. As already noted, the neuron n_J whose choice strength is the largest is chosen. Then, an action is chosen according to the weighting values w^m_{kJ} connected to n_J. Action a_k in Table 2 is executed if w^m_{kJ} is the largest and the if-condition for a_k is satisfied, otherwise, the action with the next largest weight is examined.

Table 2. Proposed Actions

No.	Actions
a_1	If the leading card is ♥, then play the strongest ♥ in weaker cards than the strongest ♥ playing in the current trick.
a_2	If the leading card is ♥, then play the strongest ♥.
a_3	If the leading card is ♥, then play the weakest ♥.
a_4	If the leading card is ♠, then play the strongest ♠ in weaker cards than the strongest ♠ playing in the current trick.
a_5	If the leading card is ♠, then play the strongest ♠.
a_6	If the leading card is ♠, then play the weakest ♠.
a_7	If the leading card is ♦, then play the strongest ♦ in weaker cards than the strongest ♦ playing in the current trick.
a_8	If the leading card is ♦, then play the strongest ♦.
a_9	If the leading card is ♦, then play the weakest ♦.
a_{10}	If the leading card is ♣, then play the strongest ♣ in weaker cards than the strongest ♣ playing in the current trick.
a_{11}	If the leading card is ♣, then play the strongest ♣.
a_{12}	If the leading card is ♣, then play the weakest ♣.
a_{13}	If there is no card where suit is the same as that of the leading card, then play ♠Q.
a_{14}	If there is no card where suit is the same as that of the leading card, then play the strongest ♥.
a_{15}	If there is no card where suit is the same as that of the leading card, then play a card according to follow rules: When the leading card isn't ♠, play a stronger card of ♠K and ♠A if there is even one of ♠K and ♠A in hands and ♠Q has been played. Otherwise, the agent determine the suit which has the strongest card in all of suits and has the least the number of cards, and play the strongest card from that suit. If there are some of such cards, play a card according to the following order: ♣, ♦, ♠, and ♥.
a_{16}	If the agent is a lead player, then play the strongest ♥.
a_{17}	If the agent is a lead player, then play the weakest ♥.
a_{18}	If the agent is a lead player, then play ♥ randomly without the strongest and weakest one.
a_{19}	If the agent is a lead player, then play the strongest ♠.
a_{20}	If the agent is a lead player, then play the weakest ♠.
a_{21}	If the agent is a lead player, then play ♠ randomly without the strongest and weakest one.
a_{22}	If the agent is a lead player, then play the strongest ♦.
a_{23}	If the agent is a lead player, then play the weakest ♦.
a_{24}	If the agent is a lead player, then play ♦ randomly without the strongest and weakest one.
a_{25}	If the agent is a lead player, then play the strongest ♣.
a_{26}	If the agent is a lead player, then play the weakest ♣.
a_{27}	If the agent is a lead player, then play ♣ randomly without the strongest and weakest one.
a_{28}	If the agent is a lead player, then play the weakest ♠ in weaker cards than ♠Q.
a_{29}	If the agent is a lead player, play the strongest ♠ in weaker cards than ♠Q.

4.3. Changing the Way of Updating Weights

The player played a card determined by FALCON, based on the percepts at the time of the player's turn. Feedback was obtained from the environment after performing the action, and the weights were adjusted.

Thus, the weights reflected the degree of relations between categories and vectors, namely percepts, actions, and rewards.

We determined the feedback according to the penalty points the agent obtained in a trick: the feedback was positive when the learning agent obtained no penalty points, otherwise the feedback was negative. We consider this action by the learning agent to be an improvement, as the penalty points obtained by the learning agent are lower and those obtained by the opponents are higher. Thus, the weights were updated at a learning rate determined by the penalty points our agent and the opponent players obtained. The parameter values of the learning rates in (5), (6), and (7) were determined as follows:

$$\beta_s = 1 \tag{8}$$

$$\beta_m = \begin{cases} 0.01 & (if\ x_i^m = 1) \\ 0 & (if\ w_{iJ}^{m(\text{old})} < pv) \\ 0.01*(1-\dfrac{pv}{w_{iJ}^{m(\text{old})}}) & (\text{otherwise}) \end{cases} \tag{9}$$

$$\beta_f = 0.001 \tag{10}$$

where x_i^m is the action vector, $w^{m(old)}_{ij}$ is the pre-update weight of the category J, and pv is the penalty variable defined by:

$$pv = 0.5*(1-\frac{sp}{16}) \tag{11}$$

where sp represents the penalty points obtained by the winner in a single trick, and 16 is the largest possible number of penalty points in a trick.

4.4. Using Multiple FALCONs

We predicted that it would be more effective to use separate strategies for the lead player and the other players. Our method therefore allowed the use of independent FALCONs for the lead player and the others. We further considered that it would be useful to divide the game into groups of tricks such as an opening, middle, and endgame, and making FALCONs learn each of these groups separately. Thus, we divided the game into groups of tricks, and used independent FALCONs for each of them, as well as using separate strategies for the lead player and the others.

5. Experiments

We conducted experiments using the card game Hearts to evaluate the performance of the methods proposed in Sec. 4. In the experiments, we used the following parameter values for (12)–(15):

$$\alpha_s = \alpha_m = \alpha_f = 0.001 \tag{12}$$

$$\beta_s = 1.0,\ \beta_m = \text{variable},\ \beta_f = 0.001 \tag{13}$$

$$\gamma_s = 1.0,\ \gamma_m = 0.0,\ \gamma_f = 0.0 \tag{14}$$

$$\rho_s = 1.0,\ \rho_m = 0.0,\ \rho_f = 0.5 \tag{15}$$

Those values were selected through preliminary experiments so that the penalty points obtained by the learning agent would become smallest.

5.1. Experimental Results for Each Method

In this subsection, we adapt the proposed methods to the learning agent, and show each experimental results.

5.1.1. For updating weights

We conducted experiments to examine whether changing the learning rates is effective or not. We used three rule-based agents as opposing players to the learning agent, and measured the average penalty ratio obtained by the learning agent through 10 simulation runs for 5000 games. Fig. 2 depicts the average penalty ratio obtained by the three learning agents using FALCON: F_{murai} is the method described in [11], F_{base} is an agent that adopts the settings for percepts, actions, and feedback, and F_{weight} is an agent that adopts the proposed method of determining the value of the learning rate, in addition to F_{base}. In the experiments, each of the agents played a game against three rule-based agents. We see from Fig. 2 that the average penalty ratio of F_{weight} is lower than that of F_{base}. Thus, we conclude that these methods are useful for improving the performance of the learning agent. However, note that none of the learning agents can beat the rule-based agent, because the penalty ratio of the rule-based agent is lower than 0.25.

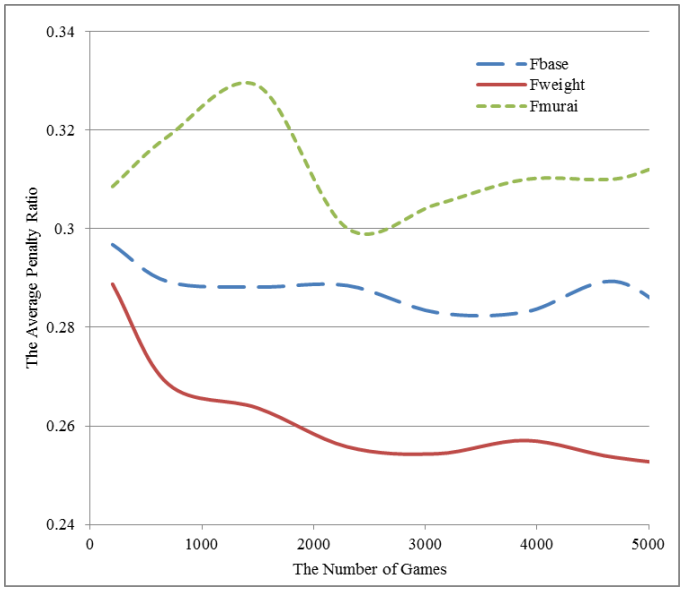


Fig. 2. The average penalty ratios of changing rates.

5.1.2. For using multiple FALCONs

We also conducted experiments to examine whether using separate strategies for the lead player and the other players is effective, using the proposed methods described in Section 5.1.1 for the learning agent. We used three rule-based agents as opposing players and measured the average penalty ratio of the learning agent through 10 simulation runs for 10000 games. Fig. 3 shows the average penalty ratio obtained by the two learning agents, where $F_{separate}$ is the agent that adopts separate strategies for the lead player and the others, in addition to F_{weight}. We can see from Fig. 3 that the average penalty ratio of $F_{separate}$ was lower than that of F_{weight}. Thus, we can conclude that this method is useful in improving the performance of the learning agent. We further conducted experiments to determine the number of FALCONs to use, and when

to divide a game, again using three rule-based agents as opposing players. We measured the average penalty ratio obtained by the learning agent through 10 simulation runs for 10000 games after training the learning agent on 50000 games. Table 3 lists the number of FALCONs used in the experiments, number of ways to divide the game, and the average penalty ratios obtained by the learning agent. The second column of Table 3 is the comma-separated list of the number of tricks making up each group (e.g., the element of third row "4, 4, 5" indicates that the game is divided into three groups as follows: the first group consists of the 1st–4th tricks, the second group consists of the 5th–8th tricks, and the third group consists of the 9th–13th tricks). We see from Table 3 that the method using seven FALCONs, shown in the 9th row, had the smallest average penalty ratio. We conclude that it is optimal to use seven FALCONs, with separate strategies for the lead and other players.

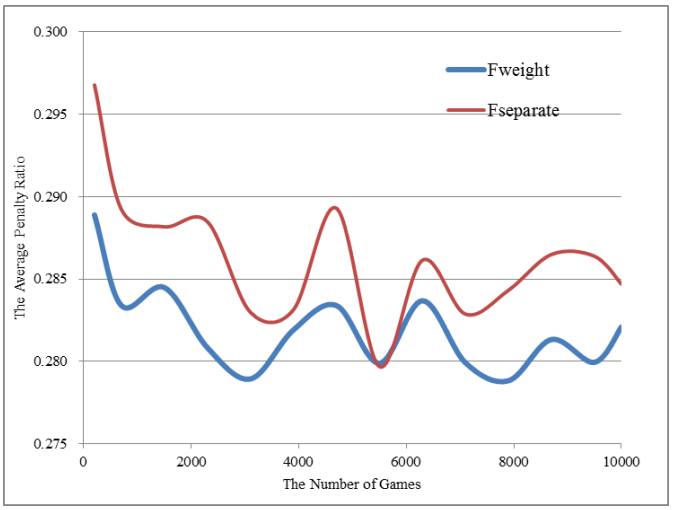


Fig. 3. The average penalty ratios of using separate strategies for the lead player and the others.

Table 3. The Average Penalty Ratios Obtained by the Agent Using Multiple FALCONs

The Number of FALCONs	The Way of Setting Groups (Tricks)	The Average Penalty Ratio
2	(1 group)	0.2430
4	7,6	0.2446
6	4,4,5	0.2490
8	4,3,3,3	0.2395
8	3,3,3,4	0.2388
10	1,3,5,3,1	0.2404
10	2,3,3,3,2	0.2392
10	3,3,3,2,2	0.2407
14	1,2,2,3,2,2,1	**0.2380**
26	(13 groups)	0.2400

5.2. Experimental Results against Rule-Based Agents

We compared rule-based agents, used as opposing players, with the learning agent. Fig. 4 shows the average penalty ratio of the learning agent using one FALCON, against three rule-based agents, through 10 simulation runs for 50000 games. We see from Fig. 4 that the average penalty ratio of the learning agent decreased as learning progressed, and that the average penalty ratio of the learning agent was smaller than that of the rule-based agents after about 8000 games, converging to under 0.24. This demonstrates that our proposed methods are sufficiently effective to beat a rule-based agent, and that a learning agent using our

methods can successfully learn rules and win games.

Fig. 4. The average penalty ratio of our learning agent and the rule-based agent.

5.3. Experimental Results of Changing the Way of Evaluation

In the previous subsection, we reported experiments to evaluate the performance of a learning agent using penalty ratios. Conversely, winning of a game of Hearts is generally based on the following rule: the winner is determined by a big game consisting of a series of small games. In small games, the penalty points held at the end of a game by each player are recorded and accumulated. Small games are played repeatedly, until the accumulated penalty points of a player exceed 100, and the ranking of players in a big-game is determined by the ascending order of the accumulated penalty points. In this subsection, we report experiments based on this rule.

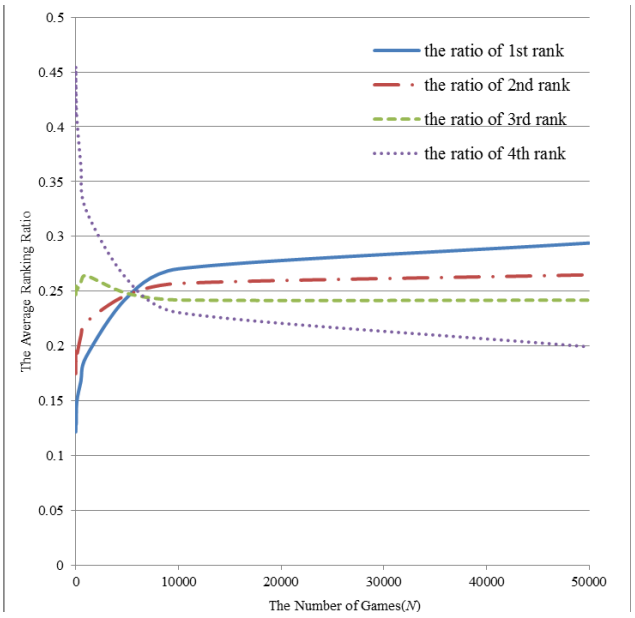


Fig. 5. The average ranking ratio of the learning agent.

In these experiments, the learning agent played against three rule-based agents for a predetermined number of games (denoted by N) for training, then for 1000 big games. We changed the value of N from 0 to 50000 and compared the ranking ratio obtained by the learning agent, after learning. The ranking ratio was the ratio of the order of ranking obtained by each agent across 1000 big-games. Fig. 5 plots the ranking ratios obtained by the learning agent through 10 simulation runs.

We see from Fig. 5 that the average ranking of 1st rank increases with learning, while that of 4th rank decreases. The ratio of 1st rank becomes the largest after about 5000 learning games. Thus, the learning agent shows good performance when evaluated in this way.

5.4. Experimental Results of Changing the Opponents

In these experiments, we changed the type of opponent used to train the learning agent, in order to examine the relationship between the type of opponent and the efficiency of learning. We used a random agent and a Monte-Carlo agent, in addition to the rule-based agent.

The Monte-Carlo agent determined a playing card using the UCT algorithm developed for Monte-Carlo simulation [12] in FALCON. The strength of the Monte-Carlo agent changed with the number of simulations. The Monte-Carlo agent chose a playing card based on the action with the best evaluation, after running the simulations for a predetermined number of games. The action was chosen from the actions described in Section 4. Our preliminary experiments indicated that the Monte-Carlo agent showed no incremental improvement after the number of simulations exceeded 250, when two Monte-Carlo agents were playing against two rule-based agents. When the number of simulations was 250, the average penalty ratio obtained by the rule-based agent was 0.27, and that by the Monte-Carlo agent was 0.23. The Monte-Carlo agent was therefore stronger than the rule-based agent. The relative strength of the random agent, the rule-based agent, and the Monte-Carlo agent ran in the following descending order: the Monte-Carlo agent, the rule-based agent, then the random agent.

In these experiments, the learning agent played against the three combinations of opponent players, as shown in Table 4, for a predetermined number of training games (denoted by N), then against three rule-based agents for 10000 games. We varied the value of N from 10 to 100000, and compared the average penalty ratios obtained by the learning agent, after learning was completed. Fig. 6 plots the average penalty ratio obtained by the learning agent through 10 simulation runs.

Table 4. List of Combinations of Opponent Players against the Learning Agent for the Experiments

Case	Opponent Players		
(a)	random agent	random agent	random agent
(b)	Monte-Carlo agent	Monte-Carlo agent	Monte-Carlo agent
(c)	rule-based agent	random agent	Monte-Carlo agent
(d)	rule-based agent	rule-based agent	rule-based agent

We can observe from Fig. 6 that the average penalty ratio of the learning agent for combination (d) was the best and that for (a) was the worst. We can also see a difference between the penalty ratios for (c) and (b). The best results were obtained when the learning agent was trained by the rule-based agents. In case (a), where the learning agent was trained by random agents, the random agents were weak and the authors believe that the learning agent was unable to derive an appropriate strategy because the wins and losses were achieved randomly. In contrast, in case (b), where the learning agent was trained by Monte-Carlo agents, it appears that the learning agent was able to deduce rules to avoid getting penalty points, but not rules to ensure that the opponents got penalty points. This is because the Monte-Carlo agent was too strong to allow the learning agent to win, and thus insufficient data was acquired on winning strategies. In case (c), because the weakest random agent always lost, the learning agent could deduce useful rules from the other players. As a result, the penalty ratio was the second largest among the four cases. We conclude that effective learning can be achieved when the learning agent is trained by an agent which is neither too strong nor too weak.

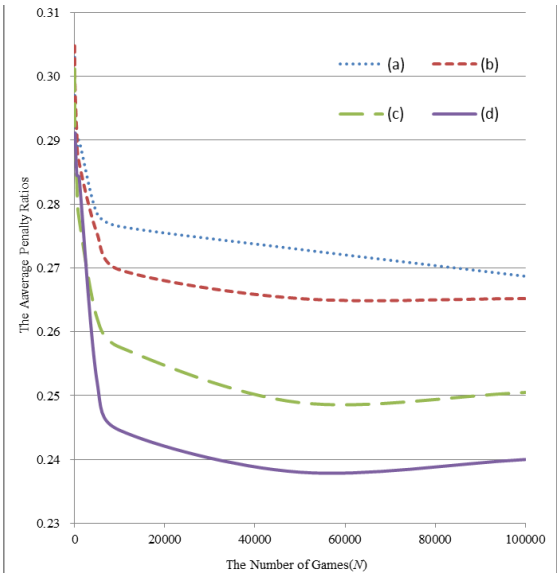


Fig. 6. The penalty ratio of the learning agent when the agent is trained in the combinations listed in Table 4.

As Ikeda has noted, it is necessary that, in training game learners, the teacher also loses some games, in order to give the learner a chance to develop winning strategies [13]. This might also apply to our experiments, in which the performance of the learning agent was changed by the strength of the opponents it was playing against.

6. Conclusion

In this study, we proposed the following six methods to efficiently train an autonomous agent in the game of Hearts using FALCON: (1) Determining percepts, (2) Setting actions, (3) Setting feedback, (4) Changing learning rates, (5) Using separate strategies for the lead player and the others, and (6) Using multiple FALCONs. Our experiments showed that these proposed methods can reduce the penalty rate, enabling the learning agent to beat a rule-based agent.

Experiments in which different combinations of opposing players were used to train the learning agent showed that the learning agent cannot learn effectively when the opponents are too strong or too weak.

A future task is to further improve the effectiveness of learning. The authors believe that changing the value of the learning rate is the most effective among the proposed methods, based on the large improvements in performance seen in the experiments. It should be possible to improve the performance of the learning agent by developing a better way of updating the weight vector, for example by modifying the weight vector based on an evaluation of the entire game. This should enable the learning agent to deduce a more suitable set of rules by which to play Hearts. In addition, in this study, percepts and actions were determined by trial and error. More efficient methods should be developed.

Acknowledgment

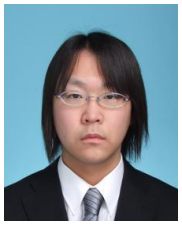
This research is in part supported by Hiroshima City University Grant for Special Academic Research (General).

References

[1] Hoshino, J. (2008, January). Editor's introduction to "Game AI"(<Special Issue> Game AI). *Journal of Japanese Society for Artificial Intelligence*, *23(1),* 43.

[2] Tanaka, H., & Tokunaga, T. (2003, December). Talking to Robot: From a viewpoint of artificial intelligence (<Special Feature> Robotics Based on AI Technology). *IPSJ Magazine*, *44(12)*, 1247-1252.

[3] Nakamura, T. (2009, May). Computer go(<Special Issue>Mind Games). *Journal of Japanese Society for Artificial Intelligence*, *24(3)*, 341-348.
[4] Hiroaki, U., Takeshi, N., Yo, N., Kazuki, M., Kenichi, T., & Tetsuhiro, M. (2006). State space segmentation for acquisition of agent behavior. *Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology* (pp. 440-446). IEEE Computer Society.
[5] Hiroaki, U., Takeshi, N., Naoki, H., Hideaki, K., Kenichi, T., & Tetsuhiro M. (2005). Fuzzy q-learning with the modified fuzzy ART neural network. *Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology* (pp. 308-315).
[6] Tan, A.-H. (2004, July). FALCON: A fusion architecture for learning, cognition, and navigation. *Proceedings of International Joint Conference on Neural Networks* (pp. 3297-3302).
[7] Hitomi, M., Hiroaki, U., & Kenichi, T. (2013). Multi-agent reinforcement learning based on multi-channel ART networks. *Proceedings of Int. Conf. Agents and Artificial Intelligence* (pp. 461-464).
[8] Xiao, D., & Tan, A.-H. (2008, December). Scaling up multi-agent reinforcement learning in complex domains. *Proceedings of International Conference on Web Intelligence and Intelligent Agent Technology* (pp. 326-329).
[9] Wang, D., Subagdja, B., Tan, A.-H., & Ng, G.-W. (2009). Creating human-like autonomous players in real-time first person shooter computer games. *Proceedings of Twenty-First Innovative Applications of Artificial Intelligence Conference* (pp. 173-178).
[10] Stichting Lone Wolves. (2004-2015). Lone wolves-web, game, and open source development. Retrieved August 22, 2015, from https://launchpad.net/ubuntu/+source/gnome-hearts
[11] Kaname, M., & Kenichi, T. (2013). *Learning Experiments for a Card Game Using a FALCON.* Graduation thesis, Hiroshima City University, Japan.
[12] Kocsis, L., & Szepesvari, C. (2006). Bandit based montecarlo planning. *Proceedings of European Conference on Machine Learning* (pp. 283-293).
[13] The digital brain which lets human "win easily". (2015, February 20). *Asahi News*.



Kenta Nimoto received the BSc from Hiroshima City University in 2014. He is currently a graduate student in Graduate School of Information Sciences at Hiroshima City University. His research interests include artificial intelligence and machine learning.

Kenichi Takahashi received his M.E. in information engineering from Nagoya Institute of Technology in 1979 and the doctor of engineering in information engineering from Nagoya University in 1986. Since 1994, he has been a professor of Hiroshima City University. His research interests include artificial intelligence, pattern information processing and machine learning. He is a member of IEEE, IEICEJ, and IPSJ.

Michimasa Inaba received the Ph.D. degree from Nagoya University in 2012. He is currently an assistant professor in Graduate School of Information Sciences at Hiroshima City University. His research interests include artificial intelligence, data mining, and natural language processing. He is a member of IEEE, IEICEI, JSAI, and IPSJ.