Applying GA for Reward Allotment in an Event-driven Hybrid Learning Classifier System for Soccer Video Games

Yuhki Inoue and Yuji Sato, Member, IEEE

Abstract—This paper describes our study applying GA to search the reward values for reinforcement learning in a soccer video game using learning classifier systems. In particular, we report the result of promotion of efficiency by dividing searched space and searching the divided space alternately. We have already proposed that acquiring algorithms by using the event-driven hybrid learning classifier system. Moreover, we have proposed that using GA for setting the reward values which have no index for setting. As the result, a probability that the reward values can be set to appropriate value for learning was obtained. In prior studies, certain rewards became searching candidate. Meantime, if the kind of the success rewards increase, setting of success rewards is difficult in practical time because the search space of reward values become spread. To address this problem, in this paper, we propose that applying the technique of dividing the searched space, and searching divided space alternately with exchange information. By comparison with the technique of searching the reward values all at once, we show a possibility that this technique have effect to improve efficiency of learning the reward values.

I. INTRODUCTION

With the growth of the Internet, video games which play through the Internet grow. Many people that have a skill of various levels play a same video game. Meantime, the video game environment has begun to change due to the explosive growth of the Internet. Because of the Internet, it is becoming easily to participate in the video games for many people, and the number of game users is increasing dramatically. Game user knowledge is also diversifying ranging from children to adults playing levels. By the above developments, a problem happens in development of the video game. As a taking place problem, there is a following problem. First, an algorithm life has been short due to shift of the player’s skill. Second, a single algorithm cannot possibly satisfy all users, and as the number of users increase, differences in strategies that users prefer and excel in can no longer be ignored. The need is therefore felt for simultaneous support of multiple strategy algorithms. Meantime, it is common in the production of the video games for human designers to explicitly specify the decision-making algorithms to be used by game agents. It is also common to use IF-THEN type of production rules as a format for describing these algorithms. Because production rules of this type make it relatively easy to describe algorithms at design time and to understand them during maintenance. In view of good consistent of IF-THEN type, as a means for addressing the above problems, taking a soccer video game as an example of a video game, we have already proposed a learning scheme [1] that considers a hybrid system and events when applying a classifier system [2] to the acquisition of decision-making algorithms by the soccer in game agents. And we have proposed that using the success rewards that are differentiated for each play based on a consideration of the different roles to each player [3, 4]. By contrast, there is a problem that indicator for setting the reward values is nothing, and the values are set by trial-and-error based on the designer’s experiment. To address above problem, we have already proposed that using the genetic algorithm to set the reward values [5]. In above proposition, we have searched only the six reward values because of a problem of the search space width, and we pointed to a possibility that the rewards that are effective for the agent learning are found by GA. Meantime, in previous method, there is a problem that if increase the types of reward values which are search candidate, searching isn’t finished in impractical time. To address this problem, in this paper, we propose that applying the technique of dividing the searched space, and searching divided space alternately with exchange information. Because, it would appear that the optimal reward values of each action have dependence relationship, but some of this interdependent relationship is not deeply. Section 2 presents an overview of the conventional soccer video game and conventional study on which this study is based. In Section 3, we propose applying partition search using genetic algorithms to the search for the success rewards in reinforcement learning. In section 4, we present an evaluation method and the results of our tests. In Section 5, we express investigation of the results, and finally we conclude with summary.

II. A SOCCER VIDEO GAME AND PROBLEMS

A. Overview of a soccer video game

The type of a soccer video game that we deal with here is a software-driven soccer video game with soccer as its theme in which two teams battle for the most points. Figure 1 shows a typical game scene targeting the area around the current position of the ball. In the soccer video game, each team has 11 players. There are 22 players in total. In the soccer video game, the game users of all 22 people that control player of the video game have not arrived. The shortage is made up by player which a computer controls. In this paper, the movements of each team player are controlled by computer.
B. A problem with the conventional technique

In the soccer video game, the agents choose the action according algorithm which game designer programmed. The algorithm to control agent’s action programmed as a set of control rules in IF-THEN (condition-action) format. Figure 2 shows an example of a rule written in IF-THEN format and corresponding scene. The rule states “If the ball is right in front of me while I am in front of the goal and if two players of the other team are between me and goal, then pass the ball to an unmarked player on my team. The algorithm written in IF-THEN format has several advantages. First, it is easy to write the algorithm. Second, if the new rule is added, it is easy to understand conventional rule and add the new rules.

On the other hand, many peoples go in for the video game and the game environment has been complex that it made problems. First, the increasing number of users means that the differences in strategies that users prefer and excel in can no longer be ignored and that multiple strategy algorithms must be simultaneously supported. Second, the appearance of users with advanced techniques has generated a need for decision-making algorithms under even more complicated environments. And finally, as the Internet makes it easy for new users to appear one after another, it must be possible to provide and maintain bug-free programs that support such complex decision-making algorithms in a time frame much shorter than that in the past. For all these reasons, the algorithm is becoming ephemeralization. And, in conventional decision-making system, the algorithm is hard to change dynamically for the complicated environment.

C. Hybrid Decision-making System

We studied the equipping of game programs with machine learning functions as an approach to solving the above problems. This is because incorporating machine learning functions in an appropriate way will enable the system to learn the game player’s strategy and to automatically evolve a strong strategy of its own. It will also eliminate worries over program bugs and significantly reduce the resources required for development and maintenance.

A number of techniques can be considered for implementing machine learning functions such as Q-learning [6], genetic algorithms (GAs) and neural networks, and we decided, in particular, on incorporating functions for acquiring rules based on classifier systems. We came to this decision considering the many examples of applying evolutionary computation to the acquisition of robot decision-making algorithms [7-9] in world of the robot soccer games such as RoboCup [10, 11], and considering the compatibility between the IF-THEN production-rule description format and classifier systems and the resulting ease of program migration. In classifier system, similarly conventional decision-making system, written IF-THEN format. Therefore Hybrid decision-making system can be made only modify classifier system. At the same time, the bucket brigade algorithm [12-17] used as a reinforcement learning scheme for classifier systems needs time to obtain an effective chain between classifiers. Figure 3 shows the basic idea of the hybrid decision-making system using a classifier system. This hybrid system has two different kinds of classifier. One classifier is achieved by learning in game and other classifier (privileged classifier) is written that is a minimum of action by the game designer. Privileged classifiers are set to the highest possible value and are not targeted for updating by learning. On the other hand, learning of classifier takes long time. As a result, hybrid decision-making system is improved by three ways. First, the proposed system adds an event analysis section and creates a table that records event frequency for each game player. Second, the classifier discovery section using genetic algorithms targets only actions while conditions are generated by adding new classifiers in accordance with the frequency of actual events. Third, the system updates the strength of classifiers by the bucket brigade algorithm starting with high-frequency events. Figure 4 show the event driven hybrid learning classifier systems that is add above improvement.

![Figure 1. Example of a typical game scene targeting the area around the current position of the ball.](image)

![Figure 2. Example of a rule written in IF-THEN format and corresponding scene.](image)
D. Applying GA to search for reward values

By improvement previously described, the agent’s learning performance has been improved. On the other hand, there is a problem how to set the reward values. The problem is that no indicator for setting the reward values and the reward values must be set by trial and error based on the experience. The reward values must be set in mind multiple factor of agent’s role and all that. If the reward values are set to the inappropriate value, there is the capability that the agent’s learning performance remains low. To address above problem, we had proposed using GA to search for rewards by focusing on the characteristics of GA for being able to get on with reinforcement learning even when no specific training data is available. As a result, the reward values that advance the learning performance are fined, and it became possible automate the work of setting the reward values. On the other hand, if the number of reward values which are targeted for searching increase, there was a problem that searching efficient is down because the search space broaden out and that efficient learning become difficult in practical time.

III. APPLYING GA FOR SEARCHING THE REWARD VALUES

A. Basic concept and our proposal

With the aim of efficient setting the appropriate rewards value used for learning agent’s action, to address above problem, in this paper we propose dividing a number of rewards used for learning of agent’s action into two groups and searching appropriate rewards value of two groups in alternate shifts by GA. The reason for dividing, because it would appeared that the search space can be made small and the divided search space can be searched efficiently, by dividing to two groups in recognizing the importance of dependence between the rewards. And it would appear that the whole of the rewards are improved to advance learning efficiency of agents as each groups affect agent learning by searching the rewards value in alternate shifts in each groups.

In proposal, a number of rewards are divided two groups and searched. In searching, the appropriate values are searched by GA in one group (scope of searching). Other group (scope of keeping) is kept the values while the scope of searching is searched. In scope of keeping, the rewards value is kept the value searched before. After searching appropriate rewards value in a few generations, each scope is switched from the searching to the keeping and the keeping to the searching. Figure 5 shows the flow of the searching. In figure 5, the rewards are split between α part and β part. α part is the scope of searching. β part is the scope of keeping. After searching rewards in a few generations, switch from α part to the keeping and from β part to the searching, go ahead with searching.

Figure 3. Basic idea of the hybrid decision-making system using a classifier system

![Figure 3](image-url)

Figure 4. The configuration of the event-driven hybrid learning classifier systems

![Figure 4](image-url)
B. Designing GA to the search for rewards

The chromosome definitions are shown in figure 6. The leftmost group consists of the rewards in pass actions. The second group from the left consists of the rewards in get ball actions. The third group from the left consists of the rewards in dribble actions. The last group consists of the rewards in get goal actions. An individual is composed of the chromosome which is defined above rewards. The values of the rewards are expressed as integers. Table 1 shows the success rewards for each type of play that were used in the soccer video game. The contexture of reinforcement learning based on the winning rate is shown in figure 7. Next, the flow of the searching the reward values with GA is expounded. In applying GA for searching the reward values, the group of the rewards is selected for searching value. In this paper, the scope of searching is PASS&GETBALL or DRIBBLE&GETGOAL. In the scope of keeping, the reward values were kept. The reward values which were set by before searching are used for the reward values which not selected to the scope of searching. On the other hand, in the scope of searching, parent entities which are used for uniform crossover are selected based on the fitness value by roulette selection. The fitness value is the average of the winning rate in sets of 200 games 3 times. Uniform Crossover is performed between selected individuals based on the crossover rate. Figure 8 shows the example of crossover. In figure 8, the reward values which are searched in DRIBBLE and GETGOAL. The reward values for dribbling by midfielder and for getting goal by forward are swapped over between two parent entities based on mask bits. In mutation, the reward values which are applied mutation are selected based on the mutation rate. Mutation is applied to the current values which are transformed according to Equation (1).

\[ x_{n+1} = x_n \pm y \]  (1)

where \( y \) is integer number randomly in the range from 5 to 20.

![Figure 5. The flow of the searching.](image)

![Table 1. The success reward for each play that were used in initial value (FW: forward, MF: midfielder, DF: defense)](table)

![Figure 6. The chromosome definitions.](image)

![Figure 7. The contexture of reinforcement learning based on the winning rate.](image)

![Figure 8. An example of a crossover.](image)
A. Experimental method

The tests involved playing matches between two teams in the soccer environment. The event-driven hybrid classifier system was used for HOME team. An algorithmic decision-making system was used for AWAY team. The searched reward values are used for the event-driven hybrid classifier system. The intended reward values of searching were PASS, DRIBBLE, GETBALL and GETGOAL of each position. The search range was divided into two groups. One group is PASS & DRIBBLE, other group is DRIBBLE & GETGOAL. Figure 9 show the flow of the experiment.

Phase 1: Four individuals which has the reward values of pass, dribble, get ball and get goal of each position are made ready. The initial reward values are set to random integers.

Phase 2: The game is performed in sets of 200 games 3 times. The fitness is the average of the winning rate of 3 times. The reward values are searched until 30 generations as the search range is switched.

Phase 3: In each 1st 10th 20th 30th generation, the game is performed in sets of 200 times with the use of the individual that has a highest fitness, and the average of the winning rate of 10 times out read.

Phase 4: the experiment of phase 1 to 3 is performed 3 times. The average of 3 times is used as the result.

From the preliminary experimental result, the searching range is exchanged each 5 generations. The crossover rate and mutation rate were set to 0.6 and 0.3. The data used for the result is the HOME team data. We used three different algorithms that were used for AWAY team. One was algorithm A which aims to strike a balance between attack and defense. One was algorithm B which places more emphasis on attacking play. One was algorithm C which place more emphasis on defensive play. We used these algorithms. The winning rate is defined by Equation (2) below:

\[ R_w = \frac{N_w}{N_t - N_d} \]  

(2)

where, \( N_t \), \( N_d \), and \( N_w \) are total number of matches, number of draws, and number of wins respectively.

B. Experimental results

1. Evaluation results using algorithm A

Figure 10 show the results obtained when playing against algorithm A. When GA was used to perform learning of the reward values, the winning rate was increased and the increase in the winning rate occurred earliest as searching goes. When using the initial reward values, the winning rate of about 60% was achieved around 50 matches, after which no increase in the winning rate was observed. In contrast, when using the reward values of 10 learning generations, the winning rate of over 60% was achieved around 20 matches and initial rise is fast than using the initial reward values. When using the reward values of 20 learning generations, the winning rate was achieved 65% around 30 matches and about 70% ultimately. When using the reward values of 30 learning generations, the increase in the winning rate occurred earliest, the winning rate was achieved about 70% around 30 matches and over 70% around 70 matches.

2. Evaluation results using algorithm B

Figure 11 show the results obtained when playing against algorithm B. When GA was used to perform learning of the reward values, the winning rate was increased and the increase in the winning rate occurred earliest as searching goes. In 20 and 30 learning generations, the winning rate over 60% around 20 matches. When using initial and 10 learning generation’s reward values, the winning rate achieved about 55%. In contrast, when using the reward values of 20 learning generations, the winning rate over 60% about 30 matches. When using the reward values of 30 learning generations, the winning rate was achieved about 65% around 20 matches and about 70% ultimately.

3. Evaluation results using algorithm C

Figure 12 show the results obtained when playing against algorithm C. When using the reward values of the initial values, the winning rate wasn’t achieved 60%. When using the reward values of 10 and 20 learning generations, the winning rate ultimately converged on about 70%. In contrast, the increase in the winning rate of 20 learning generations occurred earlier than 10 learning generations, where the winning rate was achieved 70% and was up 10% with 10 learning generations around 20 matches. When using the reward values of 30 learning generations, it have no advantage over 20 learning generations in the increase in the winning rate, but the winning rate ultimately achieved more than 70%.

4. Using conventional technique for searching the rewards

Figure 13, 14, 15 show the results using the reward values that were searched by the conventional technique that search the values all at once. In the case of playing against algorithm A, the winning rate not over 70%. The increase in the winning rate was not improved. In the case of playing against algorithm B, when using the reward values of 30 generations, the winning rate was achieved about 70% but was not exceeded 70%. When using 10 and 20 generations, the winning rate was achieved about 60%. In the case of playing against algorithm C, the winning rate didn’t rise though forward searching. When using 10, 20, 30 generations, the winning rate was achieved about 60%.
Figure 10. The winning rate for the case of using proposal technique for searching the rewards values. (vs. Algorithm A)

Figure 11. The winning rate for the case of using proposal technique for searching the rewards values. (vs. Algorithm B)

Figure 12. The winning rate for the case of using proposal technique for searching the rewards values. (vs. Algorithm C)

Figure 13. The winning rate for the case of using conventional technique which search the values all at once for searching the rewards values. (vs. Algorithm A)

Figure 14. The winning rate for the case of using conventional technique which search the values all at once for searching the rewards values. (vs. Algorithm B)

Figure 15. The winning rate for the case of using conventional technique which search the values all at once for searching the rewards values. (vs. Algorithm C)
Table 2 The number of shoot for the case of using the rewards values of 30th generation. (vs. Algorithm A)

<table>
<thead>
<tr>
<th></th>
<th>Number of Shoot</th>
<th>Number of Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>proposal technique</td>
<td>408.6</td>
<td>105.4</td>
</tr>
<tr>
<td>conventional technique</td>
<td>276.1</td>
<td>63.0</td>
</tr>
</tbody>
</table>

Table 3. The number of shoot for the case of using the rewards values of 30th generation. (vs. Algorithm B)

<table>
<thead>
<tr>
<th></th>
<th>Number of Shoot</th>
<th>Number of Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>proposal technique</td>
<td>371.6</td>
<td>91.9</td>
</tr>
<tr>
<td>conventional technique</td>
<td>329.3</td>
<td>77.8</td>
</tr>
</tbody>
</table>

Table 4. The number of shoot for the case of using the rewards values of 30th generation. (vs. Algorithm C)

<table>
<thead>
<tr>
<th></th>
<th>Number of Shoot</th>
<th>Number of Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>proposal technique</td>
<td>326.7</td>
<td>102.5</td>
</tr>
<tr>
<td>conventional technique</td>
<td>253.1</td>
<td>59.6</td>
</tr>
</tbody>
</table>

V. DISCUSSION

From Figure 10 through 12 show the result obtained when the event-driven hybrid learning classifier system using the searching reward values going against algorithm A, B and C. When the proposal technique was used to search the reward values, the winning rate was upped and the increase in the winning rate occurred earlier as searching the reward values progressed. For these reason, it would appear that the reward values were improved and the efficient learning was performed. From Figure 10, it shows the result which playing against algorithm A. When the reward values of the 30th generation were used, the increase in the winning rate occurred earliest in comparison to other generations. For this reason, it would appear that the reward values were set to efficient values for learning by using the proposal technique. The winning rate was stabilized, and the winning rate of over 70% was achieved ultimately. It would appear that the learning was stabilized. Figure 13 show the result for the case of using the conventional technique which searches the reward values all at once. When the conventional technique was used to search the reward values, the winning rate was not increased and there is no different in the increasing in the winning rate of each generation. It would appear that the learning efficiency was down because each reward values could not strike a balance. When the proposal technique was compared with the conventional technique in 30th generation, there is a difference in the winning rate. The winning rate of using the proposal technique is higher 10% than using the conventional technique around 30 matches. The winning rate of 20th generation in the proposal technique is higher than the winning rate of 30th generation in the conventional technique. The proposal technique showed the higher winning rate than the conventional technique at less generation number of searching. It would appear that the searching speed of the proposal technique is faster than that of the conventional technique. For these reason, it would appear that the proposal technique is available. From Figure 11, it shows the result which playing against algorithm B. The increasing in the winning rate of 20th and 30th generations occurred earlier by comparison with 10th generation. For this reason, it would appear that the reward values were set to the effective value for agent’s learning. When the reward values of 30th generation were used, the winning rate was higher than 20th generation around 30 matches. For this reason, it would appear that the reward values of 30th generation were set to the more effective values than 20th generation. Figure 14 show the result for the case of using the conventional technique for searching the reward values. Compared to the case of using the conventional technique, when the proposal technique was used to search the reward values, the increase in the winning rate occurred earlier than the conventional technique in 30th generation. It would appear that the reward values are set to the effective values for learning. From Figure 12, it shows the result which playing against algorithm C. The increase in the winning rate of 20th generation occurred earlier than 30th generation. On the other hand, the ultimately winning rate of 30th generation was higher than 20th generation. Therefore, it would appears that the reward values of 30th generation were effective for stable learning and that the reward values of 20th generation were set slightly destabilizing reward values. Figure 15 shows the result for the case of using the conventional technique for searching the reward values. Compared to using the conventional technique, when using the proposal technique for searching the reward values, the increase in the winning rate occurred earlier than the conventional technique, the winning rate of over 65% was achieved around 30 matches. On the other hand, when using the conventional technique, the winning rate was not increased. There is a 10% difference between the conventional technique and the proposal technique in 30th generation. The winning rate of the conventional technique is about 65% in 30th generation. On the other hand, the winning rate of the proposal technique is about 70% in 20th generation. The proposal technique showed the higher winning rate than the conventional technique at less generation number of searching. It would appear that the searching speed of the proposal technique is faster. Therefore, it would appear that the proposal technique is effective in algorithm C. Table 2 through 4 show the number of shoot and goal for the case of using the reward values of 30th generation which were searched by the conventional technique and the proposal technique with algorithm A, B and C. In each algorithm, when the proposal technique was used to search the reward values, the number of shoot is large. For this reason, it would appear that the reward values were set to the effective values for learning, and that the flows of the action to the point of shoot were learned effectively.
VI. CONCLUSIONS

In this paper, we have studied a technique applying GA for setting the reward values which used for reinforcement learning. In particular, we proposed that dividing the many reward into two groups and searching groups alternately. We have evaluated the validity of this technique in test based on the soccer video game. As a result, we have been able to confirm that the winning rate of the team used that the reward values was raised and the learning efficiency was improved when the reward values were searched by the proposal technique. When the proposal technique was compared with the technique which searches the values all at once, the proposal technique has set the values that the winning rate is raised and the increasing in the winning rate occurred early in less time. For this reason, if there are many reward values and the search space is large, it would appear that the proposal technique which is dividing the reward values into groups in recognizing the importance of dependence and using GA to search the reward values is available, it may eventually become possible automate the laborious and tedious work of setting the reward values.

REFERENCES


