Vol.2012-GI-27 No.4
2012/3/2

Design and implementation of Bonanza Method for the Evaluation in the Game of Arimaa

Kanjanapa Thitipong, Komiya Kanako, Yoshiyuki Kotani

Department of Computer and Information Sciences,
Tokyo University of Agriculture and Technology, Naka-chou 2-24-16, Koganei, Tokyo 184-8588, Japan

***Abstract* – Since the bonanza method had been discovered and proved to be required to make a strong Japanese chess program, the idea of applying it to the other 2-player perfect information game like Arimaa is proposed. In order to create a strong Arimaa program, the study aims to create the evaluation function that enables the accurate positional evaluation by using supervised learning algorithm called bonanza.**

INTRODUCTION

Since the computer chess program had already reached the state of the art by beating the world champion human player in 1997, the new game called "Arimaa" founded in 2002 by Omar Syed was claimed to be the new way more challenge in the research of AI game. Like chess, Arimaa, the 2-player zero-sum perfect information game, was designed to be easy to play for human but hard for the computer. A move in arimaa is the combination of 4 piece movement called steps. Due to Arimaa piece can only move to the orthogonally adjacent positions, Arimaa piece move quite slow compared to chess. Therefore, this leads Arimaa deeper in strategy and wider in game tree search space. There are 6 types of piece ordered by the strongest piece which are Elephant, Camel, Horse, Dog, Cat, and Rabbit. The goal is to bring a weakest piece, (a rabbit), to the opposite side of the board. The game also ends when a player loses all his rabbit or when he repeatedly makes the same move for three times. Stronger piece may pull or push the weaker piece into trap and perform capturing.

Figure 1: Example of Arimaa Position

Practically, with the high branching factor (16,000 possible move per position), Arimaa can be considered that it has the extremely large search space. That means it is impossible to do a full-width search within a limited time period. Fortunately, there was some study on tacking the Arimaa high branching factor effectively by using move-ordering heuristics [1]. Since 95% of expert moves were arranged in the top first 5% of the possible move list, the quality of alpha-beta search was proved to be more effective. For the evaluation of Arimaa position, the variety of technique such as hand-coded evaluation, reinforcement learning was applied to deal with the complicated in the evaluation. However, to evaluate position accurately, there still exist the way to improve the evaluation using the supervised machine learning technique called "bonanza", which aims to evaluate the position like the way human does.

I. RELATED WORK

A. Monte-Carlo Tree Search based Arimaa program

Inspired from the highly effective in Computer Go, the Monte-Carlo Tree Search (MCTS) was managed to be used as the search algorithm in Arimaa by Tomas Kozelek [2]. The "as-is" Monte-Carlo playouts used in Computer Go was proved to provide a weak Arimaa player. Besides, the standard improvement in Go such as UCT-RAVE, UCB-tuned still provided no effect to the strength of the Arimaa program. Since the MCTS approach tends to provide the average strength program compared with the Alpha-Beta approach. That means the role of evaluation function was proved to be important. So, we chose to focus on the study of the evaluation function in Arimaa in order to create a strong Arimaa program.

B. Alpha-Beta Pruning Tree Search based Arimaa program

Most of the strong Arimaa program is based on the Alpha-Beta search algorithm. Like the winner of 2011 Computer Arimaa Championship, bot_sharp, was also the Alpha-Beta engine plus the standard improvement effectively used in Chess like the move ordering technique [1]. Move ordering enables the search to search on the expert-like move first, which means that it optimized the time usage in searching the principal variations. Bot_sharp evaluation function was tuned by using the reinforcement machine

learning technique. The 4 different algorithms which are TD, TD-Leaf, Treestrap, and Rootstrap were used to perform weight training. The work had proved that the reinforcement technique with the Rootstrap weight training method did very well in Arimaa and enabled the best Arimaa program. Besides, it is also proved that there still have many ways to improve the level of game play for the computer Arimaa. Likewise, we offered the hypothesis that the supervised learning technique also provides the good evaluation for Arimaa.

B. The bonanza method

In the field of Japanese chess AI development, the bonanza method, invented by Kunihito Hoki[3], was considered to be indispensable for the good program. Bonanza is the name of the supervised machine learning method that performs the evaluation function weight training by using the expert human game play (game record) as the training instance. The idea of the bonanza technique is to find the objective function J' that can define how advantages the move is, compared with the expert human move.

$$J'(P_0, P_1, \dots, P_{N-1}, v) = \sum_{i=0}^{N-1} l(P_i, v) \quad \text{- (1)}$$

Given an amount of sample data record which provides total N-1 number of recorded positions and the weight (v), the objective function J' returns minimum value when minimax search results agree with the records. The function $l(P, v)$shown below describes the actual activities of objective function J'.

$$l(P, v) = \sum_{m=0}^{M} T[Eval(P_m, v) - Eval(P_{m=0}, v)] \quad \text{- (2)}$$

Given the position P, for each of the possible moves $P_{m=0,1,2,..,M}$ the difference between evaluation value of the position P_m and the evaluation value of the record position $P_{m=0}$is used as the input to the function T. Where T represents the monotonic increasing function, in this case, the sigmoid function is used. Then, given the function l, the gradient $\frac{\partial J(P_0,\dots,v)}{\partial v}$ is calculated to be used in the weight training process.

$$\frac{\partial J(P_0,\dots,v)}{\partial v} = T[x] * T[1 - x] * (g(P_m) - g(P_{m=0})) \quad \text{- (3)}$$

Where $x = Eval(P_m, v) - Eval(P_{m=0}, v)$. The gradient is then used as the update value in the weight training process.

$$v^{new} = v^{old} - h\,\text{sign}[\frac{\partial J(P_0,\dots,v)}{\partial v}] \quad \text{- (4)}$$

Where *h* is the learning rate, and the sign(x) function returns the sign of x.

The bonanza method had become the "must have" and proved to be useful in the modern Japanese chess program. In the other game designed to be challenge for the computer to play like Arimaa, the effectiveness of bonanza method against the Arimaa evaluation function will be shown in this work.

II. PROPOSED METHOD

The idea to apply the bonanza method to improve the evaluation function of Arimaa game was purposed for the attempt to create a strong Arimaa program. In order to deal with the difficulty in evaluation position in Arimaa, we suggest the evaluation function that all the weights were automatically trained based on the bonanza method which is hypothesized to be able to create a good evaluation function.

In this work, we started from the small set of feature. Only the piece position and the piece type had been used as the feature vector. The actual feature vector is as shown below

weight [piece type][x position][y position]

The piece in Arimaa consists of 6 types and each has the different value depended on the piece strength. The position in Arimaa consists of 64 positions (8x8). Therefore, Total 384 features were used to create the evaluation feature.

With these simple set of features, the program is expected to be able to evaluate how advantage to move the piece to a position, and the material advantage of a given position.

III. IMPLEMENTATION

A. Game Record Preparing

The game records from the Arimaa website [4] were downloaded and pre-processed to be able to use in evaluation function weight training. The online game records since the game was invented are available for download. The game record with rating more than 1800 was assumed to be a good game and being used as the training instance.

B. Arimaa Game Engine

In this work, the program was developed based on the program called "Bot_Faerie [5]" available on Arimaa website to be used as a starter Arimaa game engine. Bot_Faerie was written in C. It consists of most of the useful features such as Generate possible moves, simple Alpha-Beta Pruning Search, and a simple evaluation function (material value). Besides, it also keep the position hash value in order to avoid the search at the previously search position. Therefore, it would be a right choice to realize evaluation function weight training with the bonanza method based on the Bot_Faerie engine.

C. Weight Training

Before the weight training was performed, the initial weight was set to the initial piece values which are 20000 for Elephant, 5000 for camel, 3000 for horses, 1800 for dogs, 1500 for cats, 1000 for rabbits. In this work, the update was done by the policy

$$v^{\text{new}} = v^{\text{old}} + h[\frac{\partial J(P_0, \ldots, v)}{\partial v}]$$

Where h is the learning rate which better be set to the small number. In this work it was set to 10. And, the actual weight updates was used as the weight updates instead of the sign function used in the original work by Kunihito Hoki. The weight was trained by using 700 recorded positions from several game records. The result of bonanza weight training was shown in the table below.

TABLE I
WEIGHT FOR RABBIT PIECE

881	977	941	879	990	1009	1082	1062
1133	902	1201	1066	1091	924	867	839
991	990	846	801	1035	1057	1089	993
972	1974	837	1227	993	993	860	1142
1029	986	1044	913	1086	987	986	1005
874	1020	930	871	989	868	1001	1001
1118	997	1007	1102	995	1157	1014	1006
1008	1227	835	1014	912	1020	1011	1056

TABLE II
WEIGHT FOR CAT PIECE

1500	1500	1474	1498	1500	1490	1500	1500
1500	1309	1779	1469	1472	1571	1480	1500
1500	1500	1467	1502	1495	1392	1501	1500
1500	1500	1502	1500	1505	1600	1500	1500
1500	1409	1594	1517	1459	1558	1353	1500
1500	1498	1512	1485	1629	1521	1500	1500
1500	1509	1464	1508	1507	1456	1505	1500
1500	1500	1508	1502	1500	1500	1500	1500

TABLE III
WEIGHT FOR DOG PIECE

1795	1796	1800	1800	1800	1800	1790	1793
1761	1785	1910	1794	1804	1793	1810	1829
1761	1720	1864	1814	1795	1802	1805	1629
1812	1800	1778	1829	1803	1794	1826	1935
1749	1645	1894	1790	1820	1800	1806	1802
1818	1846	1807	1822	1776	1815	1716	1755
1784	1840	1801	1799	1804	1800	1802	1906
1800	1800	1800	1800	1800	1800	1800	1800

TABLE IV
WEIGHT FOR HORSE PIECE

3000	3000	3005	3000	3000	2998	3000	3000
2994	2742	3102	2899	2998	3008	2735	3000
2748	3331	2977	3059	2996	3065	3221	3009
3080	3025	2958	3027	3011	2771	2993	3015
2978	3026	2973	3079	3132	2881	3037	2998
3036	2859	3003	2905	2998	3064	2779	3109
2862	2970	3000	3109	2994	3000	3127	2984
3000	3155	3028	2990	3000	3000	3000	3002

TABLE V
WEIGHT FOR CAMELPIECE

5000	5000	5000	5000	5000	5005	5000	5000
5000	4997	5000	4880	4875	4987	5013	4840
4994	5022	4983	5104	5122	4875	4988	5158
4998	5074	4992	4920	4950	5089	5098	5000
5000	4984	5063	5008	5088	4998	4929	5000
4997	4887	5008	4990	4788	4901	5067	5003
5000	5005	4993	5005	5132	5003	5183	5000
5003	5003	5002	5000	5000	4997	5000	5000

TABLE VI
WEIGHT FOR ELEPHANT PIECE

20000	20000	20000	20000	20000	20000	20000	20000
20000	20000	20000	19575	20024	19993	19996	20000
19898	20028	19888	20441	20039	19935	20030	19997
20000	20145	20070	20067	19908	19812	20121	20007
20000	19925	20014	19889	20050	19950	20088	19908
20000	20000	20006	20079	19903	20044	19818	20093
20000	20045	19957	20099	20107	19930	20046	20000
20000	20000	20000	20000	20001	20000	20000	20000

The weight training result for showed in TABLE I did not clarify the trend to move rabbit since it is generally used as the wall to defense the goal for the long time in a game. So, the trained weight reveals the trend for the rabbit is not to move forward. And, this can lead to the poor effectiveness in attacking the goal.

From TABLE II and III, Cats and dogs piece were likely to be a defense pieces because of their weakness. The trained weights showed the trend that cats and dogs value was relatively high when they are next to the trap. This maybe because players usually use cats and dogs to grant controls the trap of their own side to prevent capturing threat when the strong pieces are attacking.

Horses, Camels, and Elephant were classified as the attacker class due to the strength. The result of weight training from TABLE IV,V,VI may not obviously showed how the pieces were likely to be moved but it can be considered that those attacking pieces were played in the center of the board. The values of weight on the edges of the board were not changed compared to the center.

Since the weight training process consumes an enormous amount of time until finish, with the specified period of time, only few data were used for the weight training. So, the limited data amount leads to the result which may not reveal some significant meaning and still far from our expectation. However, we believed that with the bonanza method, the increasing amount of data trained will lead to the better evaluation function. Also, the more pattern or strategy can still achievable.

D. Match Test

After the weight training was finished, the locally arranged Arimaa game was performed in order to test whether how the evaluation really does and how it affects the game play.

The original Bot_Faerie was selected to be a competitor to our developed program. The match was performed locally with no time

limit for thinking. On the Arimaa website, Bot_Faerie with the alpha-beta-Search and the piece material evaluation function was rated as the average beginner player level.

At the start of the match, Bot_Faerie pretended to make moves that scatter the piece to many directions. The pieces were move into the opponent side by greatly distance. On the other hand, our own developed program were not pretended to advance pieces toward the opponent side, instead the pieces move like a group.

Figure 2: The early game state

Figure 3: The middle game state

Figure 4: The late game state

Figure 5: The end game state

Unfortunately, due to our bonanza evaluation function was way too early to be used the match result in the lost to Bot_Faerie. See in the figure 4, the late game state of our developed program was not really changed compared to the start position. Besides, the rabbit pieces always stayed in own side and are not going to attack the goal as the enemy did. Despite the poor attack, our own developed program had done well on the defense against the opponent rabbits. Still, with only defense, it cannot surpass the Bot_Faerie at this time. This maybe because the weight was not appropriated trained yet. It required more game training. We think that given a certain amount of recorded data trained, bonanza method can be able to evaluate position more accurately and can surpass Bot_Faerie in the future.

IV. CONCLUSION

The idea of the bonanza method known as the state of the art machine learning algorithm in Japanese chess field, was studied the possibility to apply to capable with the game of Arimaa which required the more sophisticated evaluation function due to its deep in strategy. The idea is to find the evaluation function that can evaluate the quality of a move compared to the move the human expert does. Using a set of feature, (in this work, piece position, type) the weight of the evaluation function were trained based on the game recorded of the professional player. The process of training consumes time and certain amount of data in order to create a good evaluation function.

Using the bonanza method, the well-trained Arimaa program was expected to be able to determine whether moving a piece to given position is advantage or not. This includes the material advantage and position advantage.

With the limited time constraint, our Arimaa program using bonanza supervised machine learning algorithm for the evaluation was unable to win against the test program. However, with the certain amount of game record trained we think our own developed program can surpasses the test program and given the preferable result using the designed feature.

There are many ways to improve our own developed program. For example, the feature used in the program can be extended to the 2 pieces relationship, 3 pieces relationship. With the same set of train data, considering relation between pieces has the higher expectation to be way to improve the evaluation.

Some search improvement like probcut [6], etc. can be added into the program in order to reduce the time in searching positions by pruning the sub tree that does not affect the minimax value.

V. REFERENCE

[1] David Jian Wu, "Move Ranking and Evaluation in the game of Arimaa," Harvard College, Massachusetts, BA thesis 2011.

[2] Tomas Kozelek, "Method of MCTS and the game of Arimaa," Charles University of Prague, Czech Republic, MS thesis 2009.

[3] Kunihito Hoki, "Optimal control of minimax search result to learn positional evaluation," *11th Game Programming Workshop 2006*, pp. 78-83, 2006.

[4] Arimaa Game Archive, Retrieve 2012/01/09, From "www.arimaa.com/arimaa/"

[5] Arimaa Sample C Bot (bot_faerie), Retrieve 2012/01/09, From "www.arimaa.com/arimaa/"

[6] Michael Buro, "Improving Heuristic Mini-Max Search by Supervised Learning," *Artificial Intelligence*, vol. 134(1-2), pp. 85-89, 2002.