Adaptation of Game AIs using Genetic Algorithm: Keeping Variety and Suitable Strength

Kokolo Ikeda, Yu Tanaka, Simon Viennot, Nguyen Huy Quoc and Yohei Ueda

***Abstract*—Advancement of artificial intelligence (AI) technology have made computer game players stronger than any human players in many games. However, it is still difficult to create game AIs that have a suitable strength for the average human player, while having various strategies and avoiding unnatural moves. We propose a method for generating "a group of rival AIs" which satisfy these requirements by using a Genetic Algorithm. In this paper, this method is applied to Othello game and evaluated through some experiments and questionnaires.**

***Index Terms*—genetic algorithm, variety maintenance, natural behavior, entertainment.**

I. Introduction

ADVANCEMENT of artificial intelligence technology have made computer-game-players (we call just AIs) stronger than any human players in many games. For example in 1997, Deep Blue has beaten the World Chess Champion, Garry Kasparov. And more recently, an AI has beaten the one-time master in Shogi (Japanese Chess). So we can say that the "strength" of AI is now sufficient for almost all games.

However, programs that can beat champions are too strong for most players, so AIs are often intentionally weakened for average players. But simple methods such as shallowing the depth of search or adding random values to the evaluation function are often harmful to the player's delight, because such weakened moves are sometimes **unnatural**.

Furthermore, even if different levels of AIs are obtained by using those methods, they might have a similar strategy. As a result, players cannot study **various strategies** and might feel bored when playing continuously against a given AI. It costs a lot of labor to create a set of AIs that have different strategies, therefore automation of making various AIs is needed.

Now, we advocate the following requirements, in order to make AIs friends or rivals of human players, instead of just strong enemies: (a) Adjusting the strength: players can play with AIs that have similar strength, (b) Playing natural moves: players do not feel unnaturalness in AI moves, even if the strength of AI is well adjusted, (c) Having various strategy: the AI can play with many strategies, (d) Teaching people: players can learn strategies and techniques from AIs.

In this paper, we mainly discuss about the first three requirements. Recently, such purposes instead of pure strength, are increasing their importance [1].

II. Purpose : Generating Rivals of Human Players

The purpose of this paper is to generate "rival" AIs to beginner human players. For this goal, we consider that the following three requirements should be satisfied.

- (Req-a) Strength of the AIs is almost the same as the target player
- (Req-b) Players feel that the moves of the AIs are natural
- (Req-c) AIs use various features and strategies

Respective to the requirements, we introduce three component methods in this paper. The total system enables users to play with AIs of the same level, and furthermore it builds several AIs with different strategies, so that players can continue to play with high motivation.

- (Comp-a) A method to measure the player's strength from game records, and imitate him
- (Comp-b) An Input-output model that removes unnatural moves from legal moves
- (Comp-c) An algorithm that generates various AIs whose strengths are suitable to the player

The proposed system can be applied widely to most games, and in this paper we experimented with the game of Othello. The reason is that the rules of Othello are simple, a lot of people know them, strong AI is already available[2][3], and this kind of systems have not been researched well.

III. Approaches

A. Approach for Whole System

The purpose of this research is to construct a system that generates AIs with nearer strength and different characteristics. Figure 1 is the whole image of the system, it mainly consists of two parts :

1) Player's game record is evaluated by a **tester AI**, and an **agent AI** which has the same strength is built.
2) **Individual AIs** are optimized by using a genetic algorithm (GA) whose evaluation function is calculated from both the *winning ratio* against the agent AI and *varieties* of parameters.

B. Building an Agent AI :

At the first step (1), an AI that has the same strength as the target player is constructed, based on the component technology (Comp-a). This AI is used as an agent of human player, and used to measure the strength of individual AIs.

- (1-1) Take some game records of the player, for example games against an arbitrary AI.
- (1-2) Tester AI evaluates the moves of the records, based on the difference of evaluation value between the player's move and the tester AI's best move. In other words, it evaluates how wrong the player's moves are.

Japan Advanced Institute of Science and Technology, Asahidai 1-1, Nomi, Ishikawa, Japan e-mail: kokolo@jaist.ac.jp

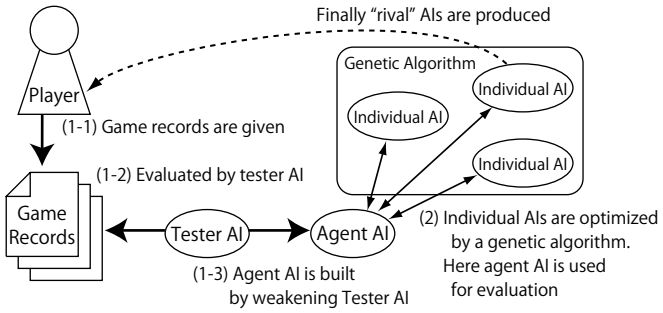

Fig. 1. The whole image of the proposed system

- (1-3) From the result of (1-2), agent AI is built by weakening the tester AI.

Tester AI has to be stronger than the target players. The detail of this algorithm will be mentioned at section IV.

C. Optimizing Individual AIs :

At the next step (2), individual AIs are generated by using a GA, so that they have almost the same strength as the agent AI and have various strategies. Finally several individual AIs are selected to be matched. This step is based on two component technologies, (Comp-b) input-output model that removes unnatural moves from legal moves, and (Comp-c) algorithm that generates various AIs whose strength are suitable to the player.

A GA is used as (Comp-c). GA is one of the stochastic optimization methods, and its signature is keeping many solutions as individuals, this helps evaluating how various the strategies are. As the evaluation function of the GA, two different measures are employed and mixed, how the individual's strength is similar to the agent AI's, and how the individual's parameters is far from the others. An individual AI consists of the model (Comp-b) and parameters, and the parameters are optimized by the GA.

By this combination of components, it is expected that the three requirements are satisfied. The detail of this algorithm will be mentioned at section V.

IV. Building Agent AI

In this section the details of component (Comp-a) are discussed. Limiting the depth of the search, or adding random values to the evaluation function, are common ways to weaken a given original AI for beginner players. But such methods cannot control well the strength of AI. For example the difference between depth-n search and depth-$(n+1)$ search is too big, or the strength with randomized evaluation functions is unstable.

A. Evaluation and Imitation of Players

Then, in our research, "weakness" is defined by the averaged difference of evaluation values between the best move and the actual move, and it is utilized for evaluating and imitating the strength of the target player. The weakness μ is measured by a sufficiently strong AI, called the **tester AI**, and then the weakness is imitated by a controlled AI, called the **agent AI**. The main components (such as the state evaluation function or the depth of search) of the agent AI are same as the tester AI, but the agent AI purposefully select a bad move so that the weakness is imitated. The concrete procedures is as follows.

1) At move t in the player's records, the best move a_t* is computed by the Tester AI
2) Both the best move a_t* and the actual move a_t of the player are evaluated, and the difference of evaluation values $v(a_t*) - v(a_t)$ is calculated.
3) Weakness is defined by the average of these differences, from the first move to the 20th move, $\mu = \frac{1}{20}\sum_{t=1}^{20} v(a_t*) - v(a_t)$. When two or more games are available, μ is defined by the average of them.
4) After μ is computed, the agent AI is controlled so that its weakness is almost μ.
   a) For example, assume that μ is 20.
   b) All the legal moves are evaluated, and the differences (badness) of the evaluation values to the best move are calculated, for example $\{0, 5, 40, 50\}$. If this is the first move, the move with badness 5 (nearest to 20) is selected.
   c) Assume that the differences of the next move is $\{0, 15, 30, 50\}$. In this case, the move with badness 30 is selected instead of 15, because the average with the previously selected difference (17.5 instead of 10) is nearest to the target $\mu = 20$.

B. Configuration of Tester AI

In this paper, the following AI is used as the tester AI (and also as the agent AI by weakening):

- Search algorithm: minmax tree search without $\alpha\beta$, for attaining accurate value
- Depth of search: 5
- Parameters of the evaluation function: parameters of Expert level of Reversi-in-C# [1]

This tester AI is clearly stronger than standard beginners. In a preliminary experiment, we obtained 20 wins per 20 games.

C. Validation of the Proposed Method

The purpose of building an agent AI is to imitate the strength of a given player. In order to validate the robustness of the proposed method, we employ various kinds of AIs in the place of various human players, and build an agent AI for each of them. The experiment is successful if a target player and its corresponding agent AI have almost the same strength.

We employed five different AIs as target players:

- $\alpha\beta_4$: Reversi in C# Intermediate, depth 4 $\alpha\beta$ search
- $\alpha\beta_3$: Reversi in C# Intermediate, depth 3 $\alpha\beta$ search
- UCT_1 : Monte-Carlo Tree Search (MCTS) with UCT [7], 1 sec per move
- UCT_{10} : MCTS with UCT, 10 sec per move
- UCT+ : MCTS with UCT+[4], 10 sec per move

200 games were done for each pair, between a target player and its agent AI. Table I shows the results. From the results,

[1] www.codeproject.com/Articles/4672/Reversi-in-C

we can say that the agent AI and the target have almost the same strength in each case.

TABLE I
THE WEAKNESS μ OF EACH TARGET PLAYER, AND THE RESULTS OF 200 GAMES OF THE AGENT AI TO THE TARGET

target player	weakness μ	win-loss (rate) of the agent AI
$\alpha\beta_4$	25.12	111-85 (0.57)
$\alpha\beta_3$	28.98	96-100 (0.49)
UCT_1	36.68	114-87 (0.57)
UCT_{10}	26.51	118-80 (0.59)
UCT+	21.47	97-99 (0.49)

V. OPTIMIZATION USING A GENETIC ALGORITHM

In this section, the components (Comp-b) and (Comp-c) are discussed in details.

A. Input-output model and tactical parameters

In this research, $\alpha\beta$ method with a fixed depth and some heuristics is used as the input-output model of individual AI. The evaluation function of a board state is constructed from a linear sum of difference of "number of stones", "openness", "number of legal moves"and "number of defined stones (the stones cannot be taken by opponents)". These concepts are relevant to the game of Othello and also used in Reversi in C#.

Each individual of the GA has its own weight for each term of the sum, except that the weight of "number of stones" is fixed to 100 for all individuals. It implies that each individual has 3 free parameters.

Two simple heuristics that prevent unnatural moves from legal moves (Comp-b) are also introduced: "if the AI can take the corner, then always take" and "prevent the opponent from taking the corner at the next turn if possible". These heuristics, specific to the game of Othello, were carefully selected by interviewing 10 beginner players. The effectiveness of these heuristics is confirmed in the experiment of section VII-B.

B. About the genetic algorithm

The GA (Comp-c) employed in this research is MGG-best2 [5]. The procedure of MGG-best2 is as follows (see also Figure 2).

1) A population of N_{pop} individuals (with 3-dimensional parameters) is generated. Each parameter is randomly selected from the range $[1, 10000]$.
2) Two individuals are selected randomly from the population as parents.
3) A crossover operator is applied to the parents. N_{child} individuals are obtained as children. Details of this crossover are in section V-C.
4) The fitness values of the parents and the children are calculated. Details of the fitness calculation are in section V-D.
5) The two best individuals of the family are selected and returned to the group, in place of the parent individuals.
6) Repeat N_{cycle} times from step 2.

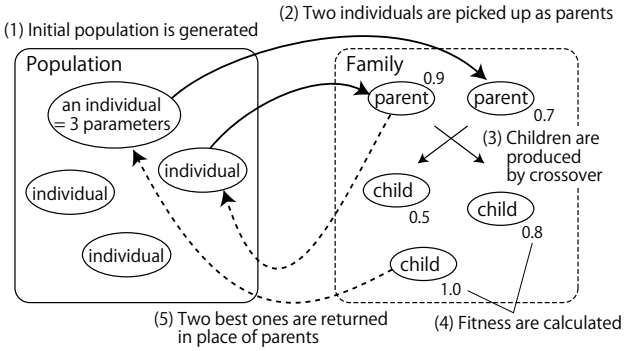

Fig. 2. MGG-best2 consists of five steps

C. BLX-α method

BLX-α [6] is employed as the crossover operator. In BLX-α, a child is created by referring to parents chromosome coordinates and a parameter α. Figure 3 shows an example of this crossover on 2-dimensions.

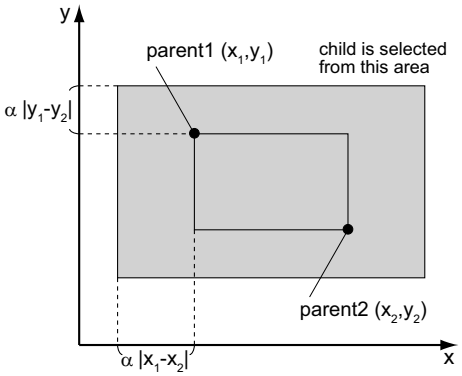

Fig. 3. BLX-α, child is generated in the gray (hyper-)rectangle

When the parents chromosome coordinates are (x_1, y_1) and (x_2, y_2), then the values max_{cx}, min_{cx}, max_{cy}, min_{cy} can be defined by the following equations.

$$max_{cx} = max(x_1, x_2) + \alpha|x_1 - x_2| \quad (1)$$
$$min_{cx} = min(x_1, x_2) - \alpha|x_1 - x_2| \quad (2)$$
$$max_{cy} = max(y_1, y_2) + \alpha|y_1 - y_2| \quad (3)$$
$$min_{cy} = min(y_1, y_2) - \alpha|y_1 - y_2| \quad (4)$$

Then, the child's chromosome coordinates x is selected randomly from $[min_{cx}, max_{cx}]$, and y is selected randomly from $[min_{cy}, max_{cy}]$. The actual crossover used for this paper is a variant of this procedure, where the crossover is computed on the logarithm of the coordinates instead of directly on the coordinates themselves. First, the logarithm of the coordinates is computed, then the crossover is computed, and finally the children coordinates are obtained by exponentiation. In practice, it leads to better results of the GA.

If α is 0 then all children are inside their parents coordinates, but if α is larger than 0 then a child can be outside of its parents coordinates. It means that this method holds both crossover and mutation at once. In the experiment, we fixed $\alpha = 0.5$.

D. Fitness

The design of the fitness function is an essential part to achieve the goal. In our case, the natural moves (Req-b) are already achieved by two heuristics, then measurement about the strength closeness (Req-a) and about strategy variety (Req-c) must be included in the fitness function.

Equation (5) is the equation to calculate the fitness of each AI. f_{wr} is a fitness concerning the winning ratio, f_{sim} is a penalty concerning the parameters variety, and $\beta(>0)$ is a fixed constant weight to balance f_{wr} and f_{sim}. The AI becomes better with bigger f_{wr} and smaller f_{sim}.

$$f = f_{wr} - \beta f_{sim} \tag{5}$$

The fitness concerning the winning ratio is given by equation (6). It shows how similar the strength of an individual AI is to the agent AI. The best value of f_{wr} is 1.0 when $wr = 0.5$, and the worst value of f_{wr} is 0.0 when $wr = 0.0$ or $wr = 1.0$.

$$f_{wr} = 4 \times (0.5 - |0.5 - wr|)^2 \tag{6}$$

On the other hand, the penalty concerning the parameters variety is given by equation (7). n is the total number of different individuals in the population and (x_i, y_i, z_i) are the parameters of individual i. The penalty for an individual i gets bigger when its parameters are closer to other individuals.

$$f_{sim} = \frac{1}{n} \cdot \sum_{j \neq i} \frac{1}{d^2(x_i, x_j) + d^2(y_i, y_j) + d^2(z_i, z_j)^2} \tag{7}$$

$$\text{where} \quad d^2(w_i, w_j) = (log_{10} w_i - log_{10} w_j)^2$$

VI. Experiment for Confirming the GA Behavior

Before the experiment with human subjects, we perform a series of experiments that confirm the behavior of the proposed GA. The settings are as follows.

- Number of individuals (N_{pop}): 20
- Number of children (N_{child}): 10
- Number of games (each individual against the agent AI, for evaluating f_{wr}): 20
- Number of generations (N_{cycle}): 150
- Weight β : one of 0.01, 0.001, 0.0001

What we want to know is (1) whether the transition and convergence of the GA is reasonable or not, (2) whether the final population of individual AIs is sufficiently varied or not, and (3) whether AIs with different parameters actually select different moves. Three values $0.01, 0.001, 0.001$ are tested for the weight β, and one of them is employed in the final experiment.

A. Fitness convergences using different β

In this subsection we discuss about the behavior of GA, transition and convergence of fitness values. This optimization problem is multi-objective optimization in fact, concerning f_{wr} and f_{sim}, and β is the balancing parameter. Figure 4 shows the transition graph of f_{wr}, and Figure 5 the transition graph of f_{sim}. 20 trials are done for each β and their average is shown.

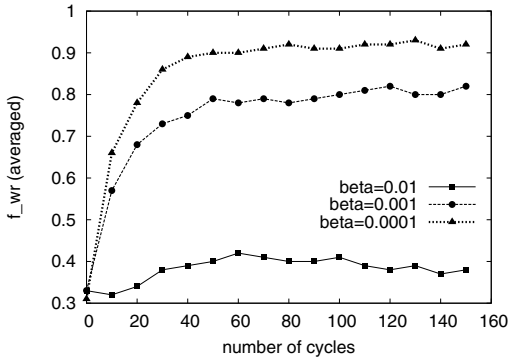

Fig. 4. Averaged transition of f_{wr}, with $\beta = 0.01, 0.001, 0.0001$

From Figure 4, we can see that the performance of f_{wr} is stably improving in early generations in the two cases $\beta = 0.001$ and $\beta = 0.0001$, finally reaching about 0.8 and 0.9 respectively. These values mean that the winning ratio of the individual AIs to the agent AI is in the range from 0.45 to 0.55. It confirms that the indivual AIs obtained with these values of β have almost the same strength as the agent AI.

On the other hand, in the case $\beta = 0.01$, f_{wr} is stagnating around 0.4. This means that the winning ratio of the individual AIs is out of $[0.32, 0.68]$, and we can conclude that this parameter 0.01 is too big. Bigger (better) f_{wr} is gained by lower β, which is a reasonable trend.

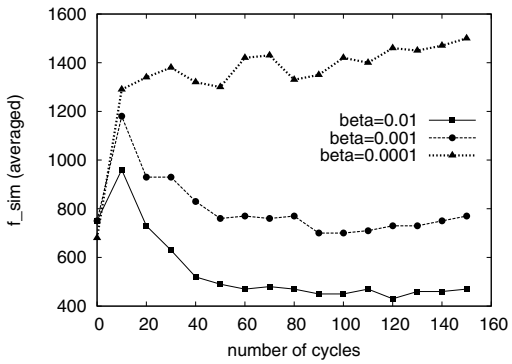

Fig. 5. Averaged transition of f_{sim}, with $\beta = 0.01, 0.001, 0.0001$

Figure 5 shows that the similarity f_{sim} is stably decreasing in early generations (except the first one), in the two cases $\beta = 0.01$ and $\beta = 0.001$. Lower (better) f_{sim} is gained by higher β, though it is difficult to say how better $\beta = 0.01$ is, compared to $\beta = 0.0001$. This trend is also reasonable.

From these results, we can say that β is working well to balance the two fitness functions, about strength and about similarity.

B. Distribution of populations using different β

In this subsection we discuss about the parameters distribution of the final individuals. Considering that our purpose is to

generate good rivals for a target player, the parameters should be widely distributed, with a strength near to the player.

Figure 6 is a quasi-3D plot of the distribution of parameters of good individual AIs gained by one GA trial, in the case $\beta = 0.001$. Here "good individual" means that its winning ratio is in $[0.4, 0.6]$, a condition that was satisfied by 12 out of 20 individuals. The value plotted is the winning ratio of the individuals to the agent AI, evaluated in 200 games (which is more accurate than the evolution process). The parameters are widely distributed, except the first parameter, and the number of individuals is enough.

Figure 7 shows the result in the case $\beta = 0.01$. Here, only 2 individuals satisfy the winning ratio condition. This β is too high, though the parameters of the two individuals are varied. Lastly, Figure 8 shows the result in the case $\beta = 0.0001$. Though almost all the individuals satisfy the winning ratio condition, the parameters have converged to a small area, except for one individual. It means that β is too low.

From these result, $\beta = 0.001$ is the best value, at least in these settings.

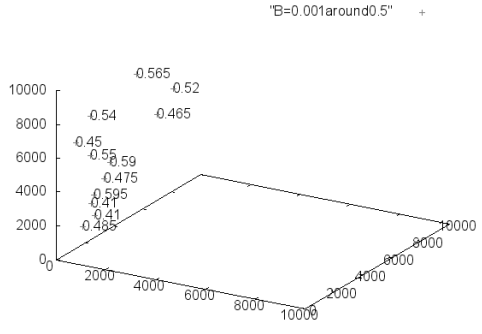

Fig. 6. Distribution of gained parameters, $\beta = 0.001$

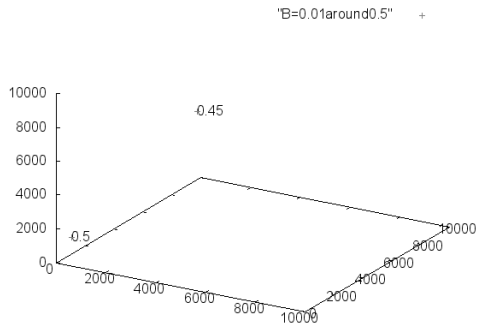

Fig. 7. Distribution of gained parameters, $\beta = 0.01$

C. Move difference of AIs

In the last subsection, we showed that various good parameters can be gained by a GA, in the case $\beta = 0.001$. However, it is still unclear whether these AIs with different parameters actually select different moves or not, and whether players can feel so or not.

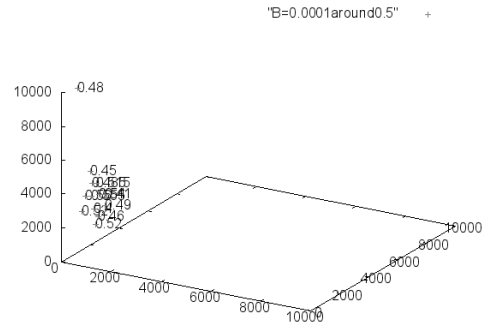

Fig. 8. Distribution of gained parameters, $\beta = 0.0001$

Here, we picked up three indivuals with parameters as different as possible, from the experiment with $\beta = 0.001$. The method to select these three individuals from the population is as follows:

1) Remove bad individuals whose winning ratio is out of the range $[0.4, 0.6]$.
2) Choose the pair (AI_1, AI_2) with the longest Euclidean distance.
3) Choose the individual AI_3 which maximizes the sum of Euclidean distances to AI_1 and AI_2.

The result of this selection is shown on Figure 9.

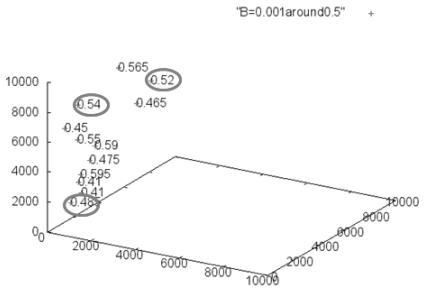

Fig. 9. Selection of three different AIs from population

Table II shows the actual parameters (weights for features) and winning ratios of the three selected individuals. They have almost the same strength, but are clearly different from each other, at least in the meaning of the parameters.

TABLE II
PARAMETERS (WEIGHTS) OF SELECTED AIS AND THEIR WINNING RATIO TO THE AGENT AI

AI	(weight of stones)	weight of openness	weight of legal moves	weight of defined stones	winning ratio
AI 1	(100)	1183	6130	7417	0.52
AI 2	(100)	612	1378	8009	0.54
AI 3	(100)	692	693	1844	0.485

To investigate how they can select different moves, we prepared 300 different boards from self-play, and the answers from the three individuals are compared. Table III shows that

the mismatch ratios between pairs of AIs. Around 30% of moves are different. We also confirmed that expert players can understand the charateristics of each AI, such as a tendency to increase the number of legal moves.

However, our target is not expert players but beginner or intermediate players. In the next section, a blind test is conducted to verify if such players can also feel the difference between the AIs.

TABLE III
MISMATCH RATIOS BETWEEN TWO AIS

	AI 1	AI 2	AI 3
AI 1		0.253	0.353
AI 2	0.253		0.317
AI 3	0.353	0.317	

VII. EXPERIMENT USING HUMAN SUBJECTS

The goal of this paper is to generate rival AIs that satisfy (Req-a) their strength is almost the same as the target player, (Req-b) players feel that their moves are natural, and (Req-c) their features and strategies are varied. In section V-A we introduced two heuristics for Req-b, but it is not verified. In the previous section, we verified Req-a and Req-c through a series of experiments using only AIs, but it is not verified with human players.

Then, in this section, we evaluate our system through an experiment using 18 novice/intermediate human players. The procedure is as follows.

1) Three game records are taken from each player, by playing against Reversi-in-C# Beginner level
2) An agent AI is built for each $player_i$. Then the strengths are different each other.
3) Individual AIs are optimized by using GA and the agent AI of the target player. Three final individuals A_i, B_i, C_i are selected with the method of section VI-C.
4) Step 3 is repeated after disabling the heuristics of section V-A about natural moves, and three individuals a_i, b_i, c_i are selected again.
5) the 6 AIs are coupled in 8 pairs, $(A_iA_i), (B_iB_i), (C_iC_i), (A_iB_i), (B_iC_i), (C_iA_i), (a_ib_i), (c_ic_i)$, and the pairs are sorted randomly.
6) $Player_i$ plays against two AIs of a pair in a row, and answer to a questionnaire. 5 questions are asked:
   - How strong is the former AI of the pair for you ? How about the latter? (five level rating)
   - How many unnatural moves the former AI selected? How about the latter? (four level)
   - Do you think the two AIs are the same, or different? (two level)
7) Playing and answering is repeated for all 8 pairs.

A. Strength

In order to check that the strength of the AIs was almost the same as their target players, we divide the 18 human players in 2 groups by using their weakness μ. $\mu = 20.9$ in the case of the strongest player, and $\mu = 55.6$ in the case of the weakest player. The averaged μ of top 9 players is 26.6 and the one of bottom 9 players is 40.9.

Figure 10 shows the histogram of answers about the strength of 12 AIs with heuristics, evaluated by the top 9 players. For those players, the given opponent AIs were a bit weak, but there are not many negative answers like "too strong" or "too weak", and the total winning ratio of the players is desirable, 51.1%.

Figure 11 shows the histogram of answers after evaluation by the bottom 9 players. For those players, the given opponent AIs were a bit strong. The total winning ratio of this group of players is not good, 36.3%. Checking the history of games showed that the performance of the human players was bad in the latter half of the experiment. A possible explanation is that they were tired by playing 16 games in a row. Surprisingly, there are not many negative answers like "too strong", even in this case.

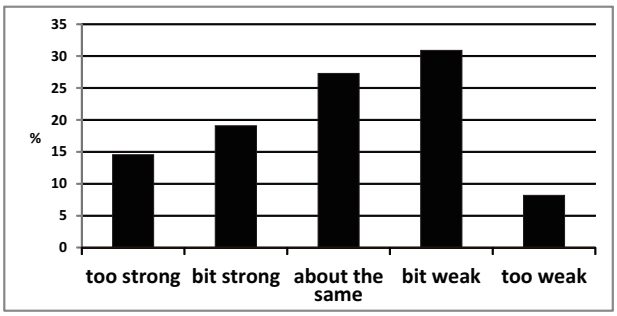

Fig. 10. Questionnaire result : How strong is this AI for you ? (top 9 players, total 108 games)

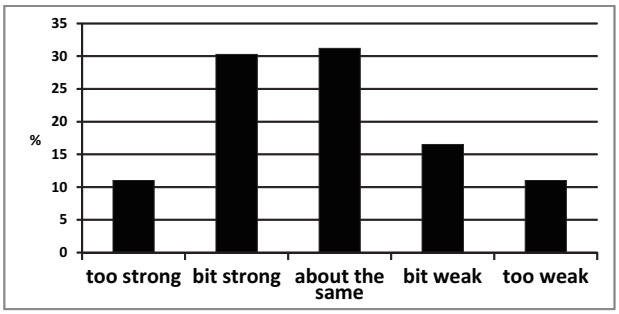

Fig. 11. Questionnaire result : How strong is this AI for you ? (bottom 9 players, total 108 games)

B. Natural moves

A total of 216 games were played against AIs with heuristics, and 72 games against AIs without heuristics. Players were not informed whether the opponent is using heuristics or not, and were asked to evaluate the number of unnatural moves.

Figure 12 shows the histogram of answers about AIs with heuristics, and Figure 13 about AIs without heuristics. It is clear from the figures that the number of moves considered

unnatural by the players increase when the heuristics are disabled. It shows that the proposed heuristics inhibits correctly unnatural moves.

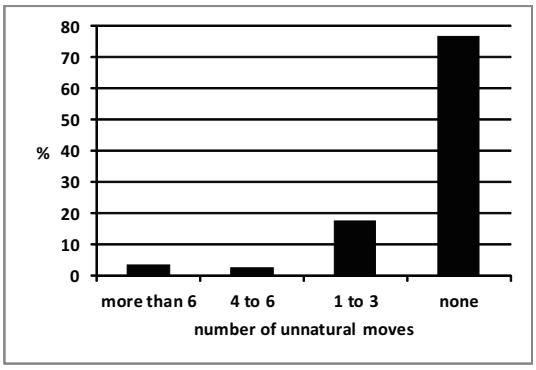

Fig. 12. Questionnaire result : How many unnatural moves were selected ? (AIs with heuristics, total 216 games)

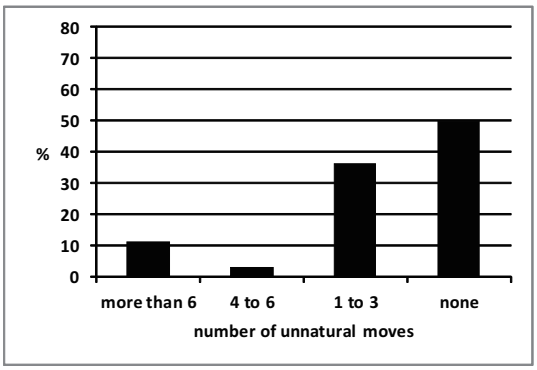

Fig. 13. Questionnaire result : How many unnatural moves were selected ? (AIs without heuristics, total 72 games)

C. Various Strategies

The question "Have these opponents various strategies ?" was possible to evaluate the AIs, but we considered it too ambiguous. Then we asked a binary question, "Are the two AIs the same, or different ?". The players were informed that 4 of the 8 pairs were the same. Only the 6 pairs using the heuristics about natural moves are used in the statistics.

Figure 14 shows the number of players that identified correctly or incorrectly the same or different AIs. For example, 3 players identified correctly half of the similar pairs, and 6 players identified correctly 4 of the 6 pairs. If the players answered randomly, the averaged total number of correct answers would be 54 in 108 pairs. However, the actual number was 65. This is significantly bigger than the random case, with over 97% confidence.

VIII. Conclusion

In this paper, we defined three requirements for constructing "rival" AIs of a target player, (1) same strength, (2) natural moves and (3) various strategies, and proposed an approach to generate such AIs by using a genetic algorithm (GA). At

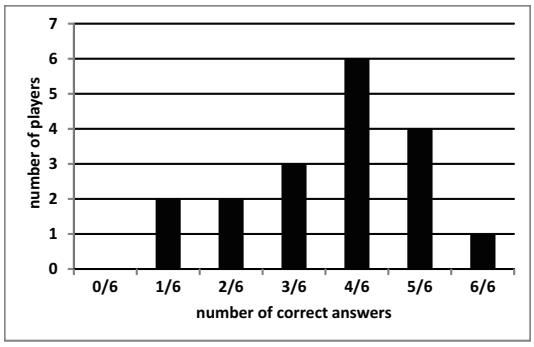

Fig. 14. Histogram of identification scores. One player identified all the pairs of similar or different AIs, but two players identified correctly only one pair in six pairs

first, an agent AI with the same strength as the target player is prepared by using game records. Next, an input-output model using two heuristics is employed for generating individual AIs, and finally the individuals are optimized by using a GA, where both the winning ratio to the agent AI and the similarity to other indivuals are considered as fitness value.

The performance of the proposed method was evaluated through several experiments. The effectiveness of the heuristics for the natural moves was clear. The strength control was successful on the average, but not sufficient for weaker players, because their strength decreases after playing too many games. Some dynamic information such as the advantage of the current situation could be used to control the strength more carefully. The variety of strategies used by the AIs was correctly identified by the human players, but less significant than we expected. One possible explanation is that the selected three features were too advanced for beginner/intermediate players. Features easier to identify by such players could be introduced such as "a tendency to play on the edge of the board".

References

[1] Pieter Spronck, Marc Ponsen, Ida Sprinkhuizen-Kuyper, Eric Postma "Adaptive game AI with dynamic scripting", Springer - MACHINE LEARNING, vol63, no3, 2006
[2] Jean-Marc Alliot and Nicolas Durand "A genetic algorithm to improve an othello program", Artificial Evolution, 307–319, 1995
[3] Siang Y. Chong, Mei K. Tan, and Jonathon D. White "Observing the Evolution of Neural Networks Learning to Play the Game of Othello", Evolutionary Computation, IEEE Transactions on, vol9, issue3, 2005
[4] Shota Maehara, Tsuyoshi Hashimoto, Hiroyuki Kobawayshi. "A new UCT search method using position evaluation function and its evaluation by Othello", IPSJ-GI, 2010-GI-24(5), 1–5, 2010
[5] H. Sato, M. Yamamura, S. Kobayashi."Minimal Generation Gap Model for GAs Considering Both Exploration and Exploitation", Proc. of IIZUKA, 494–497, 1997
[6] ESHELMAN L. J "Real-Coded Genetic Algorithms and Interval-Schemata", Foundations of Genetic Algorithms 2, 187–202, 1993
[7] Levente Kocsis and Csaba Szepesvarim, "Bandit Based Monte-Carlo Planning", Europian Conference on Machine Learning, 282–293, 2006