Emerging Collective Intelligence in Othello Players Evolved by Differential Evolution

Tetsuyuki Takahama
Department of Intelligent Systems
Hiroshima City University
Asaminami-ku, Hiroshima, 731-3194 Japan
Email: takahama@info.hiroshima-cu.ac.jp

Setsuko Sakai
Faculty of Commercial Sciences
Hiroshima Shudo University
Asaminami-ku, Hiroshima, 731-3195 Japan
Email: setuko@shudo-u.ac.jp

***Abstract*—The evaluation function for game playing is very important. However, it is difficult to make a good evaluation function. In this study, we propose to play Othello using collective intelligence of players. The evaluation functions of the players are learned or optimized by Differential Evolution. The objective value is defined based on the total score of the games with a standard Othello player. In order to generate different types of players, the objective value is slightly changed by introducing the stability of each player. Each player can select a next move using the learned evaluation function. The collective intelligence player selects a move based on majority vote where the move voted by many players is selected. It is shown that the collective intelligence is effective to game players through computer simulation.**

***Keywords*—collective intelligence; differential evolution; Othello; two-player game**

I. Introduction

In zero-sum, two-player games with perfect information such as checkers, chess and Othello, successful computer-based approaches often utilize the following three elements:

- Opening book: An opening book is used to choose good moves in opening stage.
- Evaluation function: A next move is decided by using minimax method or $\alpha\beta$-pruning in middle stage. A minimax game tree for legal moves is constructed up to a certain depth. Leaf nodes of the tree are evaluated with the evaluation function. A move which has the best evaluation value is selected as the next move.
- Complete search: The best move in endgame stage is decided by searching all legal moves until the end of the game. Victory or defeat of the game can be determined if players take the best moves always.

When it is difficult to make an evaluation function for the game like Go, Monte Carlo tree search [1] can be adopted. Many playouts, in which both players select legal moves randomly or stochastically until the end of the game, are performed. Since the next move is selected based on the winning percentage of the playouts, the evaluation function is not used.

The evaluation function is very important element for playing game. However, it is difficult to make a good evaluation function, because the special knowledge for the game is necessary to make it and much effort is necessary to improve it through trial and error. In order to solve the difficulty, there are studies to learn the evaluation function or the game strategy automatically or by unsupervised learning. For example, reinforcement learning can be adopted to learn the strategy using the victory or defeat of games as a reward or a penalty. Also, optimization methods such as evolutionary computation can be adopted to learn the evaluation function using the score of a game as an objective value or a fitness value.

In this study, we propose to play Othello using collective intelligence of Othello players in order to investigate whether collective intelligence is effective to two-player games or not. Collective intelligence [2] is group intelligence that emerges from the cooperation or competition of multiple individuals. The intelligence appears in decision making by the individuals such as bacteria, animals, human beings and computer agents. For example, Kasparov versus the World [3] was a game of chess played in 1999 over the Internet. Kasparov faced with the World Team of over 50,000 people where moves are decided by plurality vote. Although Kasparov won, he admitted that he had never expended as much effort on any other game in his life. Also, ensemble learning in machine learning is a method where multiple models (weak learners), which have relatively low performance, are generated and combined to make a better model (strong learner), which have high performance. It is thought that the ensemble learning is an example of collective intelligence.

In this study, collective intelligence is introduced to Othello-learning players. The evaluation functions of the learning players are optimized by Differential Evolution, which is an evolutionary algorithm. The objective value of each player for optimization is defined based on the total score of the games with a standard Othello player. In order to generate different types of players, the objective value is slightly changed by introducing the stability of each player. Each player can select a next move using the learned evaluation function. The next move using collective intelligence of the players is decided by majority vote. That is, the collective intelligence player chooses the move which is voted by many players. It is shown that the collective intelligence is effective to game players through computer simulation.

In Section II, related works are briefly reviewed. Othello and the evaluation function are explained in Section III. An algorithm for evolving Othello players using Differential Evolution is proposed in Section IV. In Section V, collective

intelligence of the players is investigated and experimental results are shown. Finally, conclusions are described in Section VI.

II. Related Works

The representative methods for unsupervised learning of game strategy are reinforcement learning approaches such as temporal difference learning (TDL) [4] and evolutionary learning approaches [5]. A famous reinforcement learning approach was Tesauro's TD-Gammon [6], [7] which played the backgammon. The game strategy of backgammon was expressed by the neural network which was learned by TDL. TD-Gammon was very strong, nearly as strong as the human world champion. Also, the strategies of chess [8] and Othello [9] were learned by TDL. Evolutionary learning approaches have been applied to backgammon [10], checker [11], [12] and so on. Also, comparative studies on TDL and evolutionary approach [13] was performed and hybridization of both approaches [14], [15] was proposed. But players generated by unsupervised learning in chess, Othello, shogi or Go are not comparable to strong human players.

Recently, collective intelligence in games has been paid attention to. Takahama and Sakai [16], [17] proposed to learn multiple players who play tic-tac-toe using neural networks learned by TDL in artificial game society. A consultation system to play Shogi was proposed in [18]. One of the moves that are sent by a set of players is selected as the actual move. It was shown that the system composed of three famous Shogi programs played better than each software individually. Marcolino and Matsubara [19] proposed a multi-agent version of UCT Monte Carlo Go. During the Monte Carlo simulations, one agent is randomly selected in multiple agents (agent database) and the agent will decide the move. The agents are generated by changing the order of the default heuristics of Fuego [20]. It was shown that a group of good agents selected from the database could significantly overcome Fuego.

It is thought that introducing collective intelligence to artificial game players is one of promising approaches in games. There are the following problems:

- Generating different types of players, where each player selects somewhat different moves, is key issue for emerging collective intelligence. Although different types of players can be prepared using some heuristics, the special knowledge for the game is necessary. Also, if a different initial solution or a different initial population is used in TDL or evolutionary approaches, different types of players might be generated. But the types of players tend to become similar when same objective value is used and the optimization process finds nearly optimal solution.
- The number of players is important to emerge collective intelligence. If a complex evaluation function is adopted, execution time for playing games becomes large and the number of players must be reduced.

In this study, stability of players in addition to the fitness of players is considered to generate different types of players using an evolutionary approach. A learning player plays with a standard player in plural games. The fitness value of the learning player in the games is defined by the results of the games. The plural games are repeated and the average and the standard deviation of the fitness values are obtained. Since a player who has small standard deviation is a stable player, it is thought that a player with larger fitness value and smaller standard deviation is a good player. In order to generate different types of players, the weight between the average fitness value and the standard deviation is changed appropriately. Also, the weighted piece counter which is the simplest evaluation function for Othello is adopted and 101 players are evolved to investigate whether collective intelligence can be emerged or not.

III. Othello

Othello is a two-player board game with perfect information and is played on an 8×8 board. There are 64 identical pieces called discs which are white on one side and black on the other side. First move player uses black side and second move player uses white side.

A. Rules

The game starts with two white and two black discs where same-colored discs are on a diagonal with each other in the center on the board. The players take turns putting own colored disc on the board. A legal move consists in placing a disc on an empty position so that it will bound a horizontal, vertical, or diagonal line of opponent player's discs by 2 current player's discs on each end. The sandwiched opponent's discs are reversed and the color of the discs is changed to current player's color after the disc is placed. A player passes if and only if there is no legal move. The game ends when both players passed consecutively. Then, the player who has more discs with their color wins. If both players have the same number of discs with their color, the game is tied.

Fig. 1 shows the initial Othello board and legal moves of black as stars in the left and legal moves of white in the right.

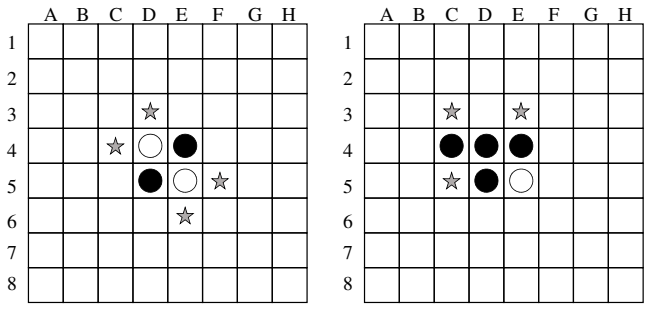

Fig. 1. Initial state of Othello and legal moves.

B. The Othello League

A good reference for different Othello player models and their estimated performance is provided by the Othello Position Evaluation Function League [21]. The goal is not to design state-of-the-art Othello players, but to evaluate position evaluation functions. Therefore, a 1-ply setup is adopted: given the current state of the board, a player generates all legal moves and applies the evaluation function to the next states

which are obtained by the legal moves. The most desirable evaluation value is selected to determine the next move. Ties are resolved at random. Player's rank in the league is based on the score obtained in games played at 1-ply against the standard heuristic player who uses the standard weighted piece counter to be described later.

Since Othello game and the players are deterministic (with an exception when at least two positions have the same evaluation value and a position is randomly selected), there can be only two games with changing the color. In order to provide better performance measure, Othello League introduces some random factors to Othello. Both players are forced to make random moves with the probability of $\varepsilon = 0.1$. In other words, ε greedy strategy is adopted by both players, where 10% of moves are selected randomly.

C. Weighted piece counter

The weighted piece counter (WPC) is the simplest position evaluation function which assigns a weight to each position in Othello board. Let a board position be denoted by $(x,\ y)$ which corresponds the x-th column and the y-th row where $x = 1, 2, \cdots, 8,\ y = 1, 2, \cdots, 8$. Let a weight of the position $(x,\ y)$ be denoted by w_{xy}. The evaluation function $f_{\boldsymbol{w}}(\cdot)$ maps current board state $\boldsymbol{b} = (b_{xy})$ into the evaluation value and is defined as follows:

$$f_{\boldsymbol{w}}(\boldsymbol{b}) = \sum_{y=1}^{8}\sum_{x=1}^{8} w_{xy} b_{xy} \tag{1}$$

$$b_{xy} = \begin{cases} 1 & \text{a black disc is placed} \\ 0 & \text{empty} \\ -1 & \text{a white disc is placed} \end{cases} \tag{2}$$

The first move (black) selects a move that maximizes the evaluation value, but the second move (white) selects a move that minimizes the evaluation value.

Fig. 2 shows the standard heuristic WPC whose weights are hand-coded by Yoshioka et al [22]. This standard WPC is often used in Othello researches as an heuristic expert opponent.

	A	B	C	D	E	F	G	H
1	1	-0.25	0.1	0.05	0.05	0.1	-0.25	1
2	-0.25	-0.25	0.01	0.01	0.01	0.01	-0.25	-0.25
3	0.1	0.01	0.05	0.02	0.02	0.05	0.01	0.1
4	0.05	0.01	0.02	0.01	0.01	0.02	0.01	0.05
5	0.05	0.01	0.02	0.01	0.01	0.02	0.01	0.05
6	0.1	0.01	0.05	0.02	0.02	0.05	0.01	0.1
7	-0.25	-0.25	0.01	0.01	0.01	0.01	-0.25	-0.25
8	1	-0.25	0.1	0.05	0.05	0.1	-0.25	1

Fig. 2. The standard heuristic Weighted Piece Counter.

IV. Learning WPC by Differential Evolution

Multiple players learns their own game strategy represented by WPC using Differential Evolution to emerge collective intelligence.

A. Learning WPC

The number of variables for WPC is 8×8=64. It is possible to reduce the number of variables by taking advantage of symmetries of Othello board. As shown in Fig. 3, WPC can be represented by 10 variables from a to j [23].

	A	B	C	D	E	F	G	H
1	*a*	*b*	*c*	*d*	*d*	*c*	*b*	*a*
2	*b*	*e*	*f*	*g*	*g*	*f*	*e*	*b*
3	*c*	*f*	*h*	*i*	*i*	*h*	*f*	*c*
4	*d*	*g*	*i*	*j*	*j*	*i*	*g*	*d*
5	*d*	*g*	*i*	*j*	*j*	*i*	*g*	*d*
6	*c*	*f*	*h*	*i*	*i*	*h*	*f*	*c*
7	*b*	*e*	*f*	*g*	*g*	*f*	*e*	*b*
8	*a*	*b*	*c*	*d*	*d*	*c*	*b*	*a*

Fig. 3. The 10 variables a to j for weighted piece counter.

The 10 sets of 100 games are played using WPC, which is defined by a decision vector $\boldsymbol{x}$, against the 0.1-greedy standard WPC player with changing first move and second move. Thus, the problem of learning WPC can defined as follows:

$$\begin{aligned} \text{maximize} \quad & f(\boldsymbol{x}) \\ \boldsymbol{x} &= (a, b, c, d, e, f, g, h, i, j) \\ f(\boldsymbol{x}) &= \text{score}(\boldsymbol{x}) + \alpha\, \text{stability}(\boldsymbol{x}) \\ \text{score}_i(\boldsymbol{x}) &= (Win_i + 0.5 Draw_i)/N, i = 1, 2, \cdots, S \\ \text{score}(\boldsymbol{x}) &= \frac{1}{S}\sum_{i=1}^{S} \text{score}_i(\boldsymbol{x}) \\ \text{stability}(\boldsymbol{x}) &= -\sqrt{\frac{1}{S}\sum_{i=1}^{S}(\text{score}_i(\boldsymbol{x}) - \text{score}(\boldsymbol{x}))^2} \end{aligned} \tag{3}$$

where α is the weight between score and stability, Win_i is the number of winning games, $Draw_i$ is the number of draw games in the i-th set. N is the total number of games in one set where $N = 100$. This kind of score is used in real Othello tournament. S is the number of the sets.

In (3), the stability is defined by the negative value of the standard deviation and a player with larger score and larger stability, or smaller standard deviation, can be obtained. In order to generate 101 players, α is changed as $\alpha = 0, 0.02, 0.04, \cdots, 2$.

B. Differential Evolution

Differential evolution (DE) is proposed by Storn and Price [24]. DE is a stochastic direct search method using a population or multiple search points. DE has been successfully

applied to optimization problems including non-linear, non-differentiable, non-convex and multimodal functions [25], [26]. It has been shown that DE is fast and robust to these functions.

In DE, initial individuals are randomly generated within given search space and form an initial population. Each individual contains D genes as decision variables. At each generation or iteration, all individuals are selected as parents. Each parent is processed as follows: The mutation operation begins by choosing several individuals from the population except for the parent in the processing. The first individual is a base vector. All subsequent individuals are paired to create difference vectors. The difference vectors are scaled by a scaling factor F and added to the base vector. The resulting vector, or a mutant vector, is then recombined with the parent. The probability of recombination at an element is controlled by a crossover rate CR. This crossover operation produces a trial vector. Finally, for survivor selection, the trial vector is accepted for the next generation if the trial vector is better than the parent.

There are some variants of DE that have been proposed. The variants are classified using the notation DE/$base$/num/$cross$ such as DE/rand/1/bin and DE/rand/1/exp. "$base$" specifies a way of selecting an individual that will form the base vector. For example, DE/rand selects an individual for the base vector at random from the population. DE/best selects the best individual in the population. "num" specifies the number of difference vectors used to perturb the base vector. In case of DE/rand/1, for example, for each parent $\boldsymbol{x}^i$, three individuals $\boldsymbol{x}^{r1}$, $\boldsymbol{x}^{r2}$ and $\boldsymbol{x}^{r3}$ are chosen randomly from the population without overlapping $\boldsymbol{x}^i$ and each other. A new vector, or a mutant vector $\boldsymbol{m}$ is generated by the base vector $\boldsymbol{x}^{r1}$ and the difference vector $\boldsymbol{x}^{r2} - \boldsymbol{x}^{r3}$, where F is the scaling factor.

$$\boldsymbol{m} = \boldsymbol{x}^{r1} + F(\boldsymbol{x}^{r2} - \boldsymbol{x}^{r3}) \tag{4}$$

"$cross$" specifies the type of crossover that is used to create a child. For example, 'bin' indicates that the crossover is controlled by the binomial crossover using a constant crossover rate, and 'exp' indicates that the crossover is controlled by a kind of two-point crossover using exponentially decreasing the crossover rate. Fig. 4 shows the binomial and exponential crossover. A new child $\boldsymbol{x}^{\text{child}}$ is generated from the parent $\boldsymbol{x}^i$ and the mutant vector $\boldsymbol{m}$, where CR is a crossover rate.

C. The Algorithm of Differential Evolution

The algorithm of DE is as follows:

- Step 0 Population size NP, a scaling factor F and a crossover rate CR are given.
- Step1 Initialization of a population. Initial NP individuals $P = \{\boldsymbol{x}^i | i = 1, 2, \cdots, NP\}$ are generated randomly in search space and form an initial population.
- Step2 Termination condition. If the number of generations reaches the maximum number of generations $T_{\max}$, the algorithm is terminated.
- Step3 DE operations. Each individual $\boldsymbol{x}^i$ is selected as a target vector (parent). If all individuals are selected, go to Step4. A mutant vector $\boldsymbol{m}$ is generated according to Eq. (4). A trial vector (child) is generated from the parent $\boldsymbol{x}^i$ and the mutant vector $\boldsymbol{m}$ using a crossover operation shown in Fig. 4. In this study, the binomial crossover is adopted. If the child is better than the parent, or the DE operation is succeeded, the child survives. Otherwise the parent survives. Go back to Step3 and the next individual is selected as a parent.
- Step4 Survivor selection. The population P is formed by the survivors. Go back to Step2.

```
binomial crossover DE/·/·/bin
  j_rand=randint(1, D);
  for(k=1; k ≤ D; k++) {
    if(k == j_rand || u(0,1) < CR)  x_k^child=m_k;
    else x_k^child=x_k^i;
  }
exponential crossover DE/·/·/exp
  k=1; j=randint(1, D);
  do {
      x_j^child=m_j;
      k=k+1; j=(j+1)%D;
  } while(k ≤ D && u(0,1) < CR);
  while(k ≤ D) {
      x_j^child=x_j^i;
      k=k+1; j=(j+1)%D;
  }
```

Fig. 4. Binomial and exponential crossover operation, where randint(1,D) generates an integer randomly from $[1, D]$ and $u(l, r)$ is a uniform random number generator in $[l, r]$.

Fig. 5 shows a pseudo-code of DE/rand/1.

```
DE/rand/1()
{
// Initialize a population
 P=N individuals generated randomly in S;
 for(t=1; t ≤ T_max; t++) {
  for(i=1; i ≤ N; i++) {
// DE operation
     x^p1=Randomly selected from P(p1 ≠ i);
     x^p2=Randomly selected from P(p2 ∉ {i,p1});
     x^p3=Randomly selected from P(p3 ∉ {i,p1,p2});
     m=x^p1+F(x^p2 - x^p3);
     x^child=trial vector is generated from
     x^i and m by the binomial crossover operation;
// Survivor selection
     if(f(x^child) > f(x^i))  z^i=x^child;
     else                     z^i=x^i;
  }
  P={z^i, i = 1,2,···,N};
 }
}
```

Fig. 5. The pseudo-code of DE, $T_{\max}$ is the maximum number of generations.

V. Experiments for Collective Intelligence

A. Experiment 1 for Generating 101 Players

DE/rand/1/bin is executed 101 times to generate 101 different players with changing the weight α as 0, 0.02, 0.04, $\cdots$, 2. The parameters for DE are shown in Table I.

TABLE I. THE PARAMETERS FOR DE.

Number of individuals	NP	50
Maximum number of generations	$T_{\max}$	200
Initial values for $\boldsymbol{x}$		[-1, 1]
Scaling factor	F	0.5
Crossover		binomial
Crossover rate	CR	0.9

Table II shows the player number, the weight on the stability α, the learned WPC, the average score and the standard deviation for each player from player 1 to player 101. The average values are also shown in the bottom rows as a reference. The highest scored player is player 37 whose score is 0.658, the lowest scored player is player 96 whose score is 0.5385, and the average score over all players is 0.573. The most stable player is player 86 whose standard deviation is 0.0126, the most unstable player is player 4 whose standard deviation is 0.0591, and the average of the standard deviations is 0.0298. These results show that a player with a small weight α on the stability tends to learn WPC with a high score but low stability and a player with a large weight tends to learn WPC with high stability and a low score.

As for the best scored player 37, the score is 0.658 and the standard deviation is second lowest value of 0.0129. The WPC defined by 10 variables is (53.34, -11.30, 20.97, -3.76, -21.61, -2.57, 0.05, 0.50, -0.26, -0.16) which weights the corner position very strongly and is different from the standard WPC. As for the second best scored player 19, the score is 0.6055, the standard deviation is 0.0335, and the WPC is (3.47, 5.70, -0.40, 1.89, 26.89, -1.30, 0.00, -1.85, 2.77, 1.25) which is different from the best WPC and the standard WPC. Thus, it is thought that various WPC can be learned.

B. Experiment 2 for Generalization Ability

Learning 0.1-greedy players against 0.1-greedy standard player is a highly noisy task due to stochastic fluctuation. Also, 1,000 games are only a small fraction of possible Othello games. Thus, the learned players might not have generalization ability to play various games. In order to investigate the generalization ability of the players, the 10 sets of 10,000 games are performed and the results are shown in Table III. The same seed for random numbers in the same set is used for every player.

The best scored player 37 in experiment 1 still attained the best score of 0.6274, although the second best player 19 attained somewhat lower score of 0.5582. It seems that player 37 is very good and stable player. The average score of all players is 0.5302 which is greater than 0.5. Thus, it is thought that players have generalization ability.

C. Experiment 3 for Collective Intelligence

A collective intelligence player (CIP) is realized by 101 players: Each player decides a move using the player's WPC. Each player votes the move. The move which has the maximum number of votes is selected as the move of CIP. If the number of votes is tied, a move is randomly selected from tied votes.

TABLE II. WPC AND RESULTS FOR 101 PLAYERS

No	α	a	b	c	d	e	f	g	h	i	j	fitness	std.
1	0.00	2.09	-2.45	0.40	-0.31	-0.27	-3.00	-3.02	1.03	1.10	2.00	0.5700	0.0497
2	0.02	-1.26	8.56	-0.34	0.71	-3.72	-7.39	-4.32	5.72	1.80	2.73	0.5655	0.0580
3	0.04	2.18	-0.18	0.09	-0.01	-0.05	-3.65	-2.83	1.39	1.44	1.48	0.5740	0.0520
4	0.06	1.73	-1.53	0.34	2.17	-0.69	-5.61	-6.13	1.99	2.38	3.07	0.5965	0.0591
5	0.08	-5.27	-2.82	1.10	1.40	-0.87	-7.40	-7.87	2.20	3.04	3.01	0.5985	0.0529
6	0.10	2.34	2.53	-0.09	0.56	-1.12	-2.37	-0.84	1.70	0.69	0.56	0.5720	0.0322
7	0.12	-2.19	0.37	-0.13	1.67	-1.85	-9.20	-7.79	3.73	3.94	3.98	0.5860	0.0399
8	0.14	1.19	3.93	-2.25	3.34	-3.63	-9.82	-7.91	4.39	4.88	4.65	0.5640	0.0346
9	0.16	-9.41	11.21	-2.00	4.34	38.88	-0.53	-0.12	-3.02	4.70	2.16	0.6000	0.0406
10	0.18	2.81	-0.91	0.28	-0.13	-0.16	-1.88	-1.96	0.59	0.74	1.02	0.5790	0.0378
11	0.20	-2.51	-4.45	-1.90	7.96	-0.60	-8.27	-9.56	2.64	3.74	3.22	0.5785	0.0297
12	0.22	5.99	-1.62	0.54	0.94	-0.32	-6.24	-5.29	2.03	2.79	3.13	0.5715	0.0328
13	0.24	2.99	-0.84	0.31	1.20	-0.37	-5.33	-5.48	2.08	2.39	2.63	0.5915	0.0357
14	0.26	3.72	4.11	-0.84	0.35	-2.94	-4.22	-1.46	3.33	1.02	0.85	0.5635	0.0363
15	0.28	7.05	-2.96	0.74	0.17	-0.53	-3.83	-4.05	1.33	1.63	3.26	0.5785	0.0332
16	0.30	-3.62	-2.50	0.56	1.88	-1.22	-6.04	-5.94	1.94	2.21	2.15	0.5780	0.0409
17	0.32	-0.63	-0.92	1.21	1.22	-0.95	-8.40	-9.24	3.10	3.66	4.09	0.6050	0.0482
18	0.34	-1.14	0.91	0.57	-0.01	-1.29	-9.09	-5.86	3.26	3.69	3.65	0.5745	0.0326
19	0.36	3.47	5.70	-0.40	1.89	26.89	-1.30	0.00	-1.85	2.77	1.25	0.6055	0.0335
20	0.38	-6.17	-1.76	0.46	2.21	-0.23	-8.87	-9.17	3.02	3.74	3.65	0.6010	0.0534
21	0.40	1.53	-1.14	0.70	0.63	-0.09	-4.41	-4.55	1.43	1.64	2.98	0.5940	0.0420
22	0.42	0.40	-1.31	0.56	0.95	-0.36	-2.95	-3.10	0.95	1.37	1.49	0.5915	0.0303
23	0.44	2.52	-1.09	1.12	0.26	-0.34	-8.87	-8.07	3.22	3.58	4.10	0.5785	0.0368
24	0.46	1.87	-2.03	0.13	2.45	-0.80	-9.02	-10.22	2.97	3.84	3.79	0.5930	0.0358
25	0.48	-8.83	-5.59	1.37	0.65	-0.95	-15.21	-12.39	6.58	6.90	7.58	0.5685	0.0407
26	0.50	-3.61	-5.52	1.01	0.75	-0.42	-8.24	-8.34	3.59	3.71	4.55	0.5855	0.0352
27	0.52	1.82	-0.01	0.76	0.63	-0.22	-3.90	-4.10	1.27	2.19	2.63	0.5645	0.0263
28	0.54	2.95	-5.39	2.76	-1.87	-3.29	-26.70	-28.22	9.73	11.06	20.09	0.5830	0.0249
29	0.56	5.56	-1.75	0.16	0.68	-0.24	-3.46	-3.80	1.44	1.82	2.14	0.5535	0.0348
30	0.58	2.09	-0.62	0.54	1.04	-0.07	-5.25	-5.31	1.46	2.04	1.86	0.5770	0.0291
31	0.60	3.85	-0.95	0.10	3.48	-1.11	-7.69	-8.91	2.80	3.12	3.54	0.5860	0.0347
32	0.62	1.80	0.21	0.14	1.25	-0.49	-5.62	-4.34	2.24	2.45	2.47	0.5830	0.0260
33	0.64	6.72	6.81	-0.33	0.96	-3.84	-5.70	-1.84	4.46	1.04	1.02	0.5615	0.0272
34	0.66	-0.79	0.31	-3.29	4.99	-0.48	-6.89	-7.48	1.97	2.66	2.41	0.5805	0.0261
35	0.68	-2.00	-2.83	-0.04	1.47	-0.77	-8.55	-9.46	2.57	3.72	3.68	0.5765	0.0376
36	0.70	-0.67	-1.31	0.28	2.48	-0.05	-5.19	-5.70	1.39	2.37	2.47	0.5935	0.0381
37	0.72	53.34	-11.30	20.97	-3.76	-21.61	-2.57	0.05	0.50	-0.26	-0.16	**0.6580**	0.0129
38	0.74	-1.89	-0.29	2.49	-0.43	-0.79	-9.63	-6.90	4.07	4.22	4.55	0.5655	0.0258
39	0.76	3.03	2.66	-2.74	3.61	-2.23	-6.82	-2.97	4.68	2.04	2.01	0.5670	0.0394
40	0.78	6.52	2.65	-1.67	2.58	-4.01	-6.96	-1.91	5.98	1.44	1.74	0.5560	0.0231
41	0.80	-3.43	-3.91	1.63	0.03	-1.55	-9.97	-10.13	2.83	3.47	6.57	0.5740	0.0207
42	0.82	5.01	-3.29	2.07	0.31	-1.12	-8.36	-8.73	2.68	3.00	5.99	0.5845	0.0190
43	0.84	2.87	0.99	-1.39	2.23	-2.14	-9.77	-3.92	6.15	1.62	1.97	0.5500	0.0220
44	0.86	4.74	3.84	-1.27	1.03	-2.50	-2.88	-0.95	2.56	0.31	0.87	0.5585	0.0190
45	0.88	14.16	1.01	0.88	1.49	-1.37	-5.76	-4.84	2.30	2.43	2.66	0.5775	0.0374
46	0.90	5.20	-5.82	0.10	2.09	-0.16	-8.61	-9.80	2.87	4.05	4.40	0.5715	0.0242
47	0.92	7.59	3.95	0.06	0.50	-4.71	-7.38	-2.01	4.57	0.76	1.66	0.5575	0.0236
48	0.94	8.90	0.27	0.13	1.50	-0.57	-6.10	-4.97	2.49	3.04	3.62	0.5770	0.0362
49	0.96	3.86	-0.39	0.39	0.46	-0.72	-4.06	-3.30	1.76	2.17	2.96	0.5725	0.0307
50	0.98	-1.78	0.56	-1.21	3.40	-3.35	-12.01	-10.78	5.63	6.05	6.42	0.5715	0.0256
51	1.00	2.81	-1.79	1.42	0.32	-1.29	-6.19	-4.78	2.19	3.00	4.28	0.5825	0.0330
52	1.02	1.88	0.89	-3.00	4.72	-0.12	-5.86	-6.16	1.92	2.26	2.06	0.5805	0.0334
53	1.04	2.54	6.15	-1.61	-1.52	-5.31	-6.26	-2.76	5.39	0.16	1.68	0.5750	0.0389
54	1.06	4.26	-2.59	0.64	0.79	0.15	-3.68	-3.78	1.12	1.64	2.55	0.5665	0.0217
55	1.08	0.85	-3.42	-0.26	2.60	-1.49	-9.48	-9.18	3.92	3.93	6.87	0.5665	0.0238
56	1.10	1.97	2.01	-2.27	3.38	-1.17	-6.56	-3.15	4.53	1.86	1.86	0.5545	0.0285
57	1.12	-2.59	-1.92	-0.22	1.41	-0.31	-5.04	-5.79	1.50	2.24	2.52	0.5675	0.0233
58	1.14	3.41	11.07	-3.37	2.92	-6.31	-10.18	-4.09	10.56	1.29	3.93	0.5715	0.0243
59	1.16	10.29	-3.72	0.78	1.17	-0.36	-8.62	-9.51	2.88	3.72	6.41	0.5825	0.0332
60	1.18	5.51	0.00	0.24	1.27	-0.76	-6.24	-4.81	2.48	2.50	2.20	0.5775	0.0371
61	1.20	37.04	4.25	-0.16	2.80	29.76	-0.73	-1.24	-1.68	2.25	2.21	0.5795	0.0281
62	1.22	5.81	-1.93	0.69	0.53	-0.38	-4.09	-4.36	1.28	1.75	1.76	0.5965	0.0430
63	1.24	2.62	-2.77	-1.78	3.57	-1.61	-7.02	-7.33	3.32	3.48	4.07	0.5755	0.0235
64	1.26	1.14	-0.25	0.15	2.97	-0.17	-5.63	-6.94	1.81	2.42	3.32	0.5655	0.0235
65	1.28	1.03	-0.34	-0.12	0.48	-0.33	-3.33	-3.03	1.24	1.43	1.62	0.5635	0.0272
66	1.30	1.62	-0.79	0.37	1.09	-0.43	-4.34	-4.94	1.43	1.84	1.77	0.5765	0.0206
67	1.32	7.23	5.41	-2.56	2.74	-4.72	-4.09	-1.17	4.63	0.45	1.03	0.5680	0.0308
68	1.34	-1.46	-1.79	-0.78	3.71	0.59	-6.04	-7.11	1.75	2.65	2.88	0.5715	0.0258
69	1.36	-2.46	-3.32	-0.14	1.25	-0.65	-6.31	-6.96	1.84	2.87	2.74	0.5725	0.0255
70	1.38	4.16	-0.03	0.08	0.75	-0.07	-3.17	-3.93	0.95	1.49	2.22	0.5710	0.0161
71	1.40	-2.27	0.95	0.91	0.80	-1.54	-7.43	-4.96	2.75	3.24	3.29	0.5705	0.0214
72	1.42	12.59	-1.57	-1.40	2.15	-2.02	-8.09	-8.72	3.74	4.13	4.57	0.5725	0.0162
73	1.44	-0.43	-3.52	0.53	2.29	-0.90	-7.65	-8.80	2.06	2.96	5.30	0.5855	0.0289
74	1.46	0.58	-3.19	0.44	0.87	0.08	-4.23	-4.39	1.35	1.69	3.11	0.5650	0.0302
75	1.48	1.77	-5.28	2.66	0.37	0.82	-10.74	-11.34	4.14	5.58	6.94	0.5610	0.0296
76	1.50	7.96	4.55	-2.08	2.86	-3.11	-7.25	-2.87	6.43	1.85	1.81	0.5515	0.0253
77	1.52	21.73	-2.40	0.42	2.33	-0.30	-9.12	-8.90	2.82	3.03	3.53	0.5685	0.0184
78	1.54	13.56	-1.46	-0.06	1.52	0.46	-4.99	-5.68	2.01	2.09	2.62	0.5435	0.0140
79	1.56	10.08	-1.63	0.22	0.93	-1.62	-7.21	-4.92	2.43	2.55	2.36	0.5810	0.0371
80	1.58	1.97	0.89	-0.49	0.59	-0.85	-2.57	-0.93	1.97	0.48	0.82	0.5525	0.0302
81	1.60	3.53	7.71	-1.98	2.14	-3.19	-5.09	-1.87	4.81	0.51	1.91	0.5545	0.0193
82	1.62	2.40	-1.48	-1.05	2.02	-0.99	-4.63	-4.45	1.78	2.13	2.17	0.5660	0.0324
83	1.64	2.58	-0.69	0.52	0.98	-1.28	-7.26	-6.69	2.57	2.99	3.54	0.5565	0.0203
84	1.66	18.90	9.03	-2.31	3.27	37.96	-2.71	-0.44	-2.06	4.26	1.25	0.5885	0.0206
85	1.68	2.17	0.10	0.62	0.18	-0.49	-2.43	-2.30	0.99	1.04	1.48	0.5720	0.0312
86	1.70	18.57	-1.26	-0.96	2.57	-0.71	-7.35	-9.14	2.64	3.31	2.87	0.5450	**0.0126**
87	1.72	1.54	-2.34	0.58	0.15	-0.68	-5.81	-5.78	1.49	2.33	3.01	0.5655	0.0271
88	1.74	-2.99	1.39	-1.41	3.61	-0.03	-5.66	-6.64	1.63	2.50	3.45	0.5705	0.0213
89	1.76	3.85	-2.60	0.38	1.99	0.50	-7.15	-7.32	2.70	3.28	3.93	0.5710	0.0207
90	1.78	1.37	-0.55	0.24	0.84	-0.29	-3.73	-3.93	1.50	1.95	1.93	0.5620	0.0260
91	1.80	2.16	0.39	1.82	0.82	-0.59	-6.06	-4.13	2.33	2.98	3.61	0.5555	0.0268
92	1.82	1.81	-0.75	1.63	0.46	-1.18	-7.36	-7.39	2.32	3.09	3.32	0.5635	0.0238
93	1.84	-0.64	-4.95	1.74	1.91	0.85	-6.37	-7.26	-0.13	3.15	2.87	0.5465	0.0224
94	1.86	1.27	0.30	0.58	1.98	0.15	-3.84	-3.54	1.37	1.49	1.45	0.5515	0.0210
95	1.88	-2.38	-0.64	0.60	-0.42	-0.19	-3.72	-3.83	1.32	1.73	1.81	0.5940	0.0251
96	1.90	8.11	3.84	-1.88	1.41	-2.21	-3.07	-1.23	3.13	0.64	1.10	0.5385	0.0234
97	1.92	-0.36	-0.45	0.75	0.82	-0.06	-4.17	-4.75	1.35	1.96	3.37	0.5600	0.0227
98	1.94	-0.05	-1.07	1.30	3.02	-1.95	-14.79	-12.50	4.90	6.11	6.16	0.5745	0.0193
99	1.96	11.05	2.20	0.82	2.82	-1.86	-14.10	-15.42	4.16	5.94	5.05	0.5580	0.0237
100	1.98	4.90	4.01	-2.82	3.59	-4.08	-8.85	-2.93	5.98	1.71	0.50	0.5540	0.0229
101	2.00	2.54	4.10	-0.37	1.14	-1.21	-4.42	-1.33	2.97	0.98	0.87	0.5410	0.0198
average	1.00	3.59	-0.08	0.13	1.53	-0.01	-6.45	-5.66	2.66	2.62	3.09	0.5730	0.0298

TABLE III. THE RESULTS IN 10 SETS OF 10,000 GAMES FOR 101 PLAYERS

No	set 1	set 2	set 3	set 4	set 5	set 6	set 7	set 8	set 9	set 10	average	std.
1	0.5294	0.5321	0.5243	0.5282	0.5275	0.5359	0.5374	0.5348	0.5330	0.5431	0.5326	0.0052
2	0.5109	0.5075	0.5078	0.5169	0.5061	0.5099	0.5181	0.5169	0.5159	0.5091	0.5119	0.0043
3	0.5269	0.5263	0.5272	0.5240	0.5322	0.5200	0.5287	0.5265	0.5293	0.5245	0.5266	0.0031
4	0.5435	0.5321	0.5372	0.5346	0.5366	0.5383	0.5424	0.5435	0.5412	0.5394	0.5389	0.0037
5	0.5382	0.5419	0.5507	0.5464	0.5446	0.5418	0.5424	0.5494	0.5360	0.5452	0.5437	0.0044
6	0.5230	0.5284	0.5296	0.5288	0.5312	0.5194	0.5112	0.5272	0.5282	0.5232	0.5250	0.0057
7	0.5209	0.5172	0.5237	0.5325	0.5294	0.5230	0.5319	0.5299	0.5295	0.5213	0.5259	0.0051
8	0.5133	0.5161	0.5161	0.5016	0.5097	0.5045	0.5058	0.5083	0.4992	0.5143	0.5089	0.0057
9	0.5555	0.5590	0.5507	0.5546	0.5558	0.5511	0.5511	0.5448	0.5531	0.5502	0.5526	0.0037
10	0.5234	0.5242	0.5285	0.5305	0.5320	0.5282	0.5252	0.5310	0.5318	0.5228	0.5278	0.0034
11	0.5179	0.5232	0.5290	0.5282	0.5275	0.5246	0.5188	0.5316	0.5292	0.5256	0.5256	0.0043
12	0.5286	0.5327	0.5367	0.5334	0.5232	0.5332	0.5341	0.5398	0.5344	0.5289	0.5325	0.0044
13	0.5535	0.5415	0.5505	0.5435	0.5499	0.5500	0.5466	0.5387	0.5460	0.5552	0.5475	0.0050
14	0.5082	0.5071	0.5135	0.5095	0.5084	0.5094	0.5112	0.4992	0.5034	0.5117	0.5082	0.0040
15	0.5462	0.5405	0.5410	0.5377	0.5412	0.5422	0.5415	0.5261	0.5454	0.5395	0.5401	0.0053
16	0.5236	0.5201	0.5248	0.5241	0.5179	0.5293	0.5169	0.5253	0.5238	0.5254	0.5231	0.0036
17	0.5428	0.5474	0.5407	0.5329	0.5367	0.5465	0.5452	0.5396	0.5422	0.5393	0.5413	0.0043
18	0.5344	0.5337	0.5249	0.5264	0.5413	0.5419	0.5381	0.5329	0.5367	0.5383	0.5349	0.0054
19	0.5517	0.5534	0.5461	0.5707	0.5487	0.5597	0.5612	0.5615	0.5624	0.5661	0.5582	0.0075
20	0.5396	0.5446	0.5532	0.5387	0.5474	0.5493	0.5485	0.5451	0.5409	0.5406	0.5448	0.0046
21	0.5374	0.5393	0.5442	0.5423	0.5437	0.5395	0.5303	0.5456	0.5429	0.5380	0.5403	0.0043
22	0.5333	0.5316	0.5284	0.5286	0.5336	0.5369	0.5443	0.5387	0.5423	0.5287	0.5346	0.0054
23	0.5344	0.5341	0.5291	0.5271	0.5291	0.5305	0.5280	0.5250	0.5275	0.5354	0.5300	0.0033
24	0.5508	0.5514	0.5536	0.5539	0.5586	0.5542	0.5563	0.5477	0.5590	0.5487	0.5534	0.0036
25	0.5392	0.5304	0.5235	0.5320	0.5329	0.5295	0.5330	0.5312	0.5396	0.5322	0.5324	0.0044
26	0.5343	0.5251	0.5300	0.5325	0.5324	0.5396	0.5339	0.5308	0.5466	0.5367	0.5342	0.0055
27	0.5360	0.5323	0.5350	0.5292	0.5218	0.5268	0.5291	0.5193	0.5376	0.5348	0.5302	0.0058
28	0.5415	0.5279	0.5357	0.5336	0.5397	0.5354	0.5281	0.5393	0.5427	0.5369	0.5361	0.0048
29	0.5311	0.5270	0.5240	0.5389	0.5283	0.5179	0.5348	0.5299	0.5336	0.5292	0.5295	0.0056
30	0.5347	0.5315	0.5323	0.5308	0.5353	0.5315	0.5361	0.5261	0.5367	0.5346	0.5330	0.0030
31	0.5548	0.5468	0.5602	0.5505	0.5520	0.5393	0.5590	0.5514	0.5457	0.5496	0.5509	0.0059
32	0.5393	0.5383	0.5449	0.5339	0.5442	0.5478	0.5416	0.5346	0.5375	0.5465	0.5409	0.0046
33	0.5241	0.5262	0.5297	0.5301	0.5293	0.5285	0.5222	0.5290	0.5285	0.5256	0.5273	0.0025
34	0.5474	0.5471	0.5415	0.5488	0.5441	0.5505	0.5417	0.5402	0.5457	0.5514	0.5458	0.0037
35	0.5373	0.5464	0.5413	0.5353	0.5314	0.5424	0.5400	0.5433	0.5385	0.5405	0.5396	0.0040
36	0.5302	0.5315	0.5291	0.5274	0.5353	0.5289	0.5265	0.5352	0.5320	0.5309	0.5307	0.0028
37	0.6294	0.6257	0.6259	0.6305	0.6257	0.6211	0.6323	0.6234	0.6298	0.6300	0.6274	0.0034
38	0.5205	0.5215	0.5264	0.5260	0.5296	0.5209	0.5237	0.5288	0.5254	0.5262	0.5249	0.0030
39	0.5140	0.5078	0.5106	0.5123	0.5086	0.5081	0.5097	0.5161	0.5133	0.5071	0.5108	0.0029
40	0.5020	0.5103	0.5109	0.5095	0.5101	0.5096	0.5082	0.5020	0.5027	0.4990	0.5064	0.0042
41	0.5438	0.5366	0.5488	0.5411	0.5422	0.5438	0.5446	0.5430	0.5431	0.5409	0.5428	0.0029
42	0.5430	0.5421	0.5360	0.5302	0.5461	0.5389	0.5324	0.5480	0.5354	0.5442	0.5396	0.0057
43	0.5042	0.5132	0.5088	0.5085	0.4997	0.5134	0.5120	0.5069	0.5153	0.5179	0.5100	0.0052
44	0.5183	0.5318	0.5170	0.5262	0.5286	0.5201	0.5310	0.5156	0.5221	0.5203	0.5231	0.0056
45	0.5349	0.5403	0.5343	0.5443	0.5283	0.5349	0.5380	0.5416	0.5368	0.5386	0.5372	0.0042
46	0.5468	0.5343	0.5411	0.5404	0.5382	0.5453	0.5454	0.5387	0.5375	0.5365	0.5404	0.0040
47	0.5186	0.5131	0.5081	0.5051	0.5051	0.5183	0.5130	0.5106	0.5098	0.5129	0.5115	0.0045
48	0.5289	0.5170	0.5299	0.5222	0.5318	0.5319	0.5221	0.5246	0.5272	0.5282	0.5264	0.0046
49	0.5242	0.5279	0.5265	0.5315	0.5273	0.5303	0.5241	0.5226	0.5235	0.5256	0.5264	0.0028
50	0.5290	0.5220	0.5281	0.5406	0.5315	0.5322	0.5367	0.5261	0.5298	0.5204	0.5296	0.0058
51	0.5112	0.5192	0.5092	0.5171	0.5161	0.5231	0.5110	0.5286	0.5152	0.5200	0.5171	0.0057
52	0.5463	0.5468	0.5309	0.5392	0.5400	0.5413	0.5401	0.5421	0.5462	0.5505	0.5423	0.0052
53	0.5139	0.5092	0.5121	0.5157	0.5117	0.5151	0.5179	0.5219	0.5141	0.5070	0.5139	0.0040
54	0.5342	0.5405	0.5336	0.5280	0.5406	0.5311	0.5401	0.5246	0.5298	0.5351	0.5338	0.0052
55	0.5274	0.5277	0.5263	0.5302	0.5210	0.5264	0.5281	0.5340	0.5288	0.5354	0.5285	0.0039
56	0.5094	0.5133	0.5155	0.5099	0.5020	0.5080	0.5085	0.5000	0.5094	0.5114	0.5087	0.0045
57	0.5442	0.5349	0.5251	0.5287	0.5405	0.5318	0.5317	0.5367	0.5377	0.5362	0.5348	0.0053
58	0.5170	0.5159	0.5168	0.5178	0.5184	0.5212	0.5145	0.5192	0.5071	0.5222	0.5170	0.0040
59	0.5444	0.5412	0.5530	0.5357	0.5425	0.5390	0.5422	0.5366	0.5425	0.5353	0.5412	0.0049
60	0.5369	0.5361	0.5341	0.5303	0.5321	0.5288	0.5186	0.5257	0.5347	0.5367	0.5314	0.0055
61	0.5392	0.5372	0.5414	0.5304	0.5403	0.5373	0.5415	0.5367	0.5449	0.5419	0.5391	0.0038
62	0.5444	0.5413	0.5419	0.5327	0.5392	0.5375	0.5373	0.5382	0.5374	0.5301	0.5380	0.0040
63	0.5391	0.5310	0.5308	0.5292	0.5333	0.5503	0.5390	0.5374	0.5358	0.5337	0.5360	0.0058
64	0.5144	0.5153	0.5265	0.5173	0.5122	0.5236	0.5265	0.5075	0.5151	0.5178	0.5176	0.0059
65	0.5402	0.5370	0.5278	0.5286	0.5279	0.5327	0.5371	0.5292	0.5324	0.5309	0.5324	0.0042
66	0.5497	0.5452	0.5423	0.5521	0.5520	0.5526	0.5609	0.5554	0.5471	0.5502	0.5508	0.0050
67	0.5231	0.5212	0.5207	0.5189	0.5164	0.5174	0.5223	0.5158	0.5195	0.5250	0.5200	0.0028
68	0.5325	0.5231	0.5244	0.5252	0.5262	0.5229	0.5317	0.5351	0.5272	0.5347	0.5283	0.0045
69	0.5353	0.5266	0.5215	0.5224	0.5299	0.5236	0.5280	0.5262	0.5082	0.5317	0.5253	0.0070
70	0.5281	0.5243	0.5225	0.5242	0.5264	0.5257	0.5325	0.5294	0.5181	0.5301	0.5261	0.0039
71	0.5214	0.5302	0.5372	0.5334	0.5295	0.5300	0.5269	0.5242	0.5188	0.5280	0.5280	0.0052
72	0.5291	0.5247	0.5272	0.5198	0.5312	0.5278	0.5284	0.5219	0.5243	0.5262	0.5261	0.0033
73	0.5400	0.5320	0.5375	0.5306	0.5407	0.5320	0.5407	0.5335	0.5361	0.5370	0.5360	0.0036
74	0.5284	0.5334	0.5180	0.5261	0.5271	0.5254	0.5317	0.5291	0.5242	0.5264	0.5270	0.0040
75	0.5222	0.5279	0.5153	0.5092	0.5331	0.5286	0.5301	0.5179	0.5202	0.5182	0.5223	0.0071
76	0.5101	0.5135	0.5153	0.5095	0.5160	0.5216	0.5187	0.5170	0.5109	0.5150	0.5148	0.0037
77	0.5277	0.5181	0.5301	0.5256	0.5238	0.5238	0.5232	0.5330	0.5227	0.5221	0.5250	0.0041
78	0.5173	0.5242	0.5196	0.5161	0.5276	0.5329	0.5148	0.5246	0.5195	0.5317	0.5228	0.0061
79	0.5314	0.5180	0.5294	0.5282	0.5228	0.5315	0.5338	0.5397	0.5269	0.5314	0.5293	0.0056
80	0.5196	0.5199	0.5220	0.5196	0.5225	0.5288	0.5175	0.5289	0.5197	0.5310	0.5230	0.0046
81	0.5283	0.5257	0.5302	0.5240	0.5313	0.5342	0.5259	0.5243	0.5232	0.5207	0.5268	0.0039
82	0.5289	0.5337	0.5341	0.5289	0.5326	0.5301	0.5332	0.5326	0.5334	0.5315	0.5319	0.0019
83	0.5203	0.5332	0.5270	0.5335	0.5310	0.5214	0.5316	0.5266	0.5290	0.5291	0.5283	0.0043
84	0.5420	0.5431	0.5474	0.5466	0.5495	0.5471	0.5465	0.5423	0.5407	0.5392	0.5444	0.0032
85	0.5130	0.5216	0.5143	0.5162	0.5131	0.5125	0.5162	0.5332	0.5131	0.5178	0.5171	0.0060
86	0.5169	0.5085	0.5172	0.5112	0.5180	0.5141	0.5146	0.5129	0.5107	0.5162	0.5140	0.0030
87	0.5116	0.5150	0.5135	0.5149	0.5092	0.5089	0.5181	0.5091	0.5134	0.5101	0.5124	0.0029
88	0.5175	0.5165	0.5187	0.5226	0.5162	0.5163	0.5162	0.5172	0.5075	0.5171	0.5166	0.0035
89	0.5292	0.5259	0.5314	0.5313	0.5304	0.5310	0.5301	0.5347	0.5305	0.5208	0.5295	0.0036
90	0.5252	0.5410	0.5310	0.5413	0.5363	0.5308	0.5333	0.5299	0.5364	0.5293	0.5335	0.0050
91	0.5169	0.5241	0.5196	0.5181	0.5272	0.5205	0.5200	0.5153	0.5148	0.5178	0.5194	0.0036
92	0.5370	0.5346	0.5369	0.5329	0.5403	0.5287	0.5326	0.5305	0.5306	0.5340	0.5338	0.0033
93	0.5183	0.5152	0.5149	0.5161	0.5101	0.5179	0.5102	0.5167	0.5170	0.5167	0.5153	0.0028
94	0.5177	0.5211	0.5226	0.5204	0.5141	0.5252	0.5129	0.5219	0.5243	0.5205	0.5201	0.0038
95	0.5375	0.5293	0.5369	0.5368	0.5313	0.5313	0.5272	0.5253	0.5375	0.5281	0.5321	0.0045
96	0.5251	0.5200	0.5204	0.5314	0.5120	0.5110	0.5181	0.5309	0.5215	0.5200	0.5210	0.0064
97	0.5288	0.5270	0.5274	0.5408	0.5387	0.5336	0.5464	0.5438	0.5325	0.5375	0.5357	0.0065
98	0.5263	0.5353	0.5316	0.5344	0.5382	0.5341	0.5357	0.5278	0.5353	0.5302	0.5329	0.0036
99	0.5334	0.5218	0.5337	0.5308	0.5239	0.5127	0.5352	0.5214	0.5252	0.5314	0.5270	0.0068
100	0.5154	0.5257	0.5199	0.5163	0.5053	0.5167	0.5195	0.5137	0.5095	0.5110	0.5153	0.0055
101	0.5102	0.5138	0.5106	0.5196	0.5197	0.5063	0.5076	0.5152	0.5145	0.5154	0.5133	0.0043
average	0.5304	0.5297	0.5301	0.5298	0.5303	0.5303	0.5309	0.5301	0.5299	0.5306	0.5302	0.0045

Table IV shows the results of CIP (101 players) and the results of CIP without the best player 37 (100 players) under the same condition of experiment 2. The average results of 101 players in Table III is also included for comparison. In every set of games, CIP and CIP without player 37 attained better results compared with the average results. CIP without player 37 outperformed CIP on average although the difference is very small. It is thought that the effect of the best player can be negligible.

Fig. 6 shows the scores of CIP, CIP without player 37 and all players in the 10 sets. A thick line shows the results of CIP and CIP without player 37, a nearly straight line shows the average scores of all players, and the line over 0.6 shows the results of player 37. It is clear that CIP with and without player 37 outperformed many players in 101 players and outperformed the average scores in all sets. Therefore, CIP is very good and stable player although player 37 outperformed all players including CIP with and without player 37.

It is difficult to know the best player among many players in advance because the learning task is highly noisy and the best player in the learning games does not always attain the best result in unlearned games. On the contrary, in this experiment, CIP player is always better than the average player. Thus, it is thought that utilizing collective intelligence of many players is effective compared with selecting a player randomly.

D. Discussion

Player 37 is a very strong player against standard player. However, it is not sure whether player 37 is a strong player in general. Another experiment, where CIP plays with player 37, is performed in order to compare the strength of CIP and player 37. Table V shows the results of CIP and the results of CIP without player 37 against player 37 under the same condition of experiment 3. CIP attained the average winning percentage 0.5073 and CIP without player 37 attained the average winning percentage 0.5098 against player 37. It is thought that CIP and CIP without player 37 is stronger than player 37.

It is possible to learn a stronger player than a particular player by playing with the player. However, it is difficult to know the learned player is strong in general. In this study, a very strong player 37 against standard player can be learned and the winning percentage against standard player of player 37 is much larger than that of CIP. Nevertheless CIP player can defeat player 37 on average. Thus, it is thought that CIP is more stable player than player 37 and can defeat not only standard player but also player 37.

VI. CONCLUSIONS

Collective intelligence is one of very important issues in computational intelligence. In this study, the effect of collective intelligence is investigated in Othello players who learn the game strategy based on an unsupervised evolutionary approach. Experiments show that the collective intelligence by majority vote of 101 players outperformed the average players in highly noisy task of learning Othello players.

In the future, the following studies will be performed:

- Learning players played with one standard player in this study. In real life, players will learn game

TABLE IV. THE RESULTS IN 10 SETS OF 10,000 GAMES FOR COLLECTIVE INTELLIGENCE PLAYER

No	set 1	set 2	set 3	set 4	set 5	set 6	set 7	set 8	set 9	set 10	average	std.
CIP	0.5512	0.5423	0.5513	0.5477	0.5476	0.5410	0.5570	0.5520	0.5410	0.5401	0.5471	0.0055
CIP w/o 37	0.5527	0.5466	0.5476	0.5483	0.547	0.5473	0.5524	0.5585	0.5403	0.5556	0.5496	0.0050
average	0.5304	0.5297	0.5301	0.5298	0.5303	0.5303	0.5309	0.5301	0.5299	0.5306	0.5302	0.0045

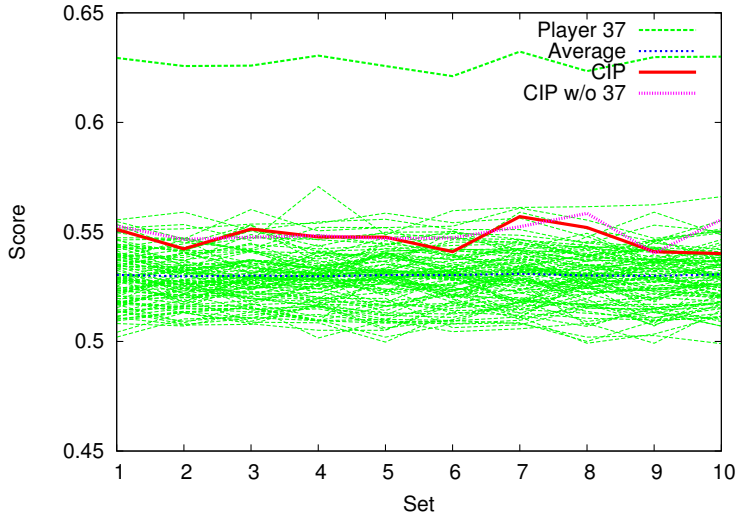

Fig. 6. Score of CIP and all players in the 10 sets.

TABLE V. THE RESULTS IN 10 SETS OF 10,000 GAMES FOR COLLECTIVE INTELLIGENCE PLAYER AGAINST PLAYER 37

No	set 1	set 2	set 3	set 4	set 5	set 6	set 7	set 8	set 9	set 10	average	std.
CIP	0.5152	0.5101	0.5148	0.5139	0.5054	0.4998	0.5067	0.5009	0.5083	0.4978	0.5073	0.0057
CIP w/o 37	0.5029	0.5102	0.5130	0.5145	0.5079	0.5092	0.4988	0.5198	0.5068	0.5148	0.5098	0.0056
average	0.5091	0.5102	0.5139	0.5142	0.5067	0.5045	0.5028	0.5104	0.5076	0.5063	0.5085	0.0056

strategies by playing with various types of players in a game society. It is interesting to consider the relation between collective intelligence and an artificial game society.

- All players had the same weight to vote the next move in this study. If proper weights for each player can be learned, the performance of collective intelligence will be improved.

ACKNOWLEDGMENT

This research is supported in part by Grant-in-Aid for Scientific Research (C) (No. 24500177, 26350443) of Japan society for the promotion of science and the 2014 Research Fund (ChosaKenkyu-Hi) of the Center for the Co-creation of Hiroshima's Future, Hiroshima Shudo University.

REFERENCES

[1] C. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, pp. 1–43, 2012.

[2] A. Pentland, "Collective intelligence," *IEEE Computational Intelligence Magazine*, vol. 1, no. 3, pp. 9–12, 2006.

[3] P. Marko and G. M. Haworth, "The Kasparov-World match," *ICGA Journal*, vol. 22, no. 4, pp. 236–238, 1999.

[4] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.

[5] D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette, "Evolutionary algorithms for reinforcement learning," *Journal of Artificial Intelligence Research (JAIR)*, vol. 11, pp. 241–276, 1999.

[6] G. Tesauro, "Practical issues in temporal difference learning," *Machine Learning*, vol. 8, pp. 257–277, 1992.

[7] G. Tesauro, "Temporal difference learning and TD-gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[8] S. Thrun, "Learning to play the game of chess," in *Advances in neural information processing systems*, D. G.Tesauro and T.Leen, Eds. Morgan Kaufmann Publishers, 1995, vol. 7.

[9] M. van der Ree and M. Wiering, "Reinforcement learning in the game of Othello: Learning against a fixed opponent and learning from self-play," in *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning 2013*, Apr. 2013, pp. 108–115.

[10] J. B. Pollack and A. D. Blair, "Co-evolution in the successful learning of backgammon strategy," *Machine Learning*, vol. 32, no. 3, pp. 225–240, 1998.

[11] K. Chellapilla and D. Fogel, "Evolving neural networks to play checkers without relying on expert knowledge," *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1382–1391, Nov. 1999.

[12] K. Chellapilla and D. Fogel, "Evolving an expert checkers playing program without using human expertise," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 422–428, Aug. 2001.

[13] S. Lucas, "Investigating learning rates for evolution and temporal difference learning," in *IEEE Symposium On Computational Intelligence and Games 2008*, Dec. 2008, pp. 1–7.

[14] M. Szubert, W. Jaskowski, and K. Krawiec, "Coevolutionary temporal difference learning for Othello," in *IEEE Symposium on Computational Intelligence and Games 2009*, Sep. 2009, pp. 104–111.

[15] L. Pena, S. Ossowski, J. Pena, and S. Lucas, "Learning and evolving combat game controllers," in *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, Sep. 2012, pp. 195–202.

[16] T.Takahama and S.Sakai, "Learning game strategy by multi-agent td players," in *Proc. of 2nd International Conference on Software Engineering, Artificial Intelligence, Networking & Parallel/Distributed Computing*, Aug. 2001, pp. 731–738.

[17] T.Takahama and S.Sakai, "Learning game strategy in an artificial game society," in *Proc. of Pan-Yellow-Sea International Workshop on Information Technologies for Network Era 2002*, Mar. 2002, pp. 172–179.

[18] T. Obata, T. Sugiyama, K. Hoki, and T. Ito, "Consultation algorithm in shogi: Can a set of players create a single strong player?" in *The 14th Game Programming Workshop*, Nov. 2009, (in Japanese).

[19] L. S. Marcolino and H. Matsubara, "Multi-agent Monte Carlo Go," in *The 10th International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '11, vol. 1. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 21–28.

[20] M. Enzenberger, M. Muller, B. Arneson, and R. Segal, "Fuego — an open-source framework for board games and Go engine based on Monte Carlo tree search," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 4, pp. 259–270, 2010.

[21] K. Krawiec and M. G. Szubert, "Learning N-tuple networks for Othello by coevolutionary gradient search," in *Proc. of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 355–362.

[22] T. Yoshioka, S. Ishii, and M. Ito, "Strategy acquisition for the game Othello based on reinforcement learning," *IEICE Transactions on Information and Systems*, vol. 82, no. 12, pp. 1618–1626, 1999.

[23] P. Hingston and M. Masek, "Experiments with Monte Carlo Othello," in *IEEE Congress on Evolutionary Computation 2007*, Sep. 2007, pp. 4059–4064.

[24] R. Storn and K. Price, "Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[25] K. Price, R. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.

[26] U. K. Chakraborty, Ed., *Advances in Differential Evolution*. Springer, 2008.