Ensemble Approaches in Evolutionary Game Strategies: A Case Study in Othello

Kyung-Joong Kim and Sung-Bae Cho

Abstract — **In pattern recognition area, an ensemble approach is one of promising methods to increase the accuracy of classification systems. It is interesting to use the ensemble approach in evolving game strategies because they maintain a population of solutions simultaneously. Simply, an ensemble is formed from a set of strategies evolved in the last generation. There are many decision factors in the ensemble of game strategies: evolutionary algorithms, fusion methods, and the selection of members in the ensemble. In this paper, several evolutionary algorithms (evolutionary strategy, simple genetic algorithm, fitness sharing, and deterministic crowding algorithm) are compared with three representative fusion methods (majority voting, average, and weighted average) with selective ensembles (compared with the ensemble of all members). Additionally, the computational cost of an exhaustive search for the selective ensemble is reduced by introducing multi-stage evaluations. The ensemble approach is tested on the Othello game with a weight piece counter representation. The proposed ensemble approach outperforms the single best individual from the evolution and ensemble searching time is reasonable.**

I. Introduction

Ensemble is a method to combine multiple decision models expecting synergism to increase the performance of systems [1][2]. It is composed of a number of models and each member contributes to the final decision of the ensemble. If each model's decision boundary is different, they can generate a new one by combining them. If they can cooperate well, the final decision boundary could be better than one of each member. In this way, the ensemble can improve the accuracy and the generalization capability on unseen dataset.

There are many factors in the success of the ensemble system. Each member should be good although it is not necessary to be the best. Also, they're not identical because there is no performance gain from the combination of the same models [3]. The number of members is important and there is evidence that the combination of the many models could be worse than the one of subsets of them [4]. Finally, there are many different ways to combine outputs from each member to generate the final decision [5].

Evolutionary game is a promising research area that combines a lot of games with evolutionary algorithms [6]. Fogel *et al*. evolved master-level players for a game of checkers and chess [7][8]. There is a good source of references for this area [9]. There are many different types of games raging from traditional board games (chess, checkers, go, Othello, and backgammon) to video games. It opens door to designing game strategies with limited expert knowledge.

It is natural to use the ensemble approach in the evolutionary algorithms because it maintains a number of solutions simultaneously [10][11]. Unlike other search algorithms, it is a population-based search and results in multiple models. Simply, an ensemble is formed with the individuals in the last generation. It is possible to form ensembles without multiple runs of learning.

There is a little attention on the ensemble research for evolutionary games. The main focus of the evolutionary games research is to exploit the best individual in the last generation. There are few papers on this topic. Kim *et al*. combined several neural network strategies evolved for a game of checkers [12]. Yang *et al*. introduced the coalition of multiple strategies in iterated prisoner's dilemma (IPD) game [13].

In this paper, an ensemble approach is systematically tested in the platform of Othello. The control parameters of the ensemble are the way to evolve each member, a fusion method, and the choice of members from candidates. There are several choices for each factor and their effect is investigated on the game of Othello. Additionally, a new method is proposed to minimize the computational cost in finding an ensemble exhaustively.

The rest of this paper is organized as follows. Section II describes backgrounds of the proposed methods: Rules of Othello, computational intelligence approaches for the game, and the ensemble approaches for evolutionary games. Section III applies ensemble approaches to the game of Othello. Control parameters of the ensemble are the learning methods for individual members, selection of members from a pool of strategies, and fusion methods. Section IV describes the experimental results and analysis.

II. Backgrounds

A. A Game of Othello

Othello has a very simple rule but it takes lifetime to master the game for human. The only rule for the game is to sandwich other player's discs with one's own discs and the captured discs flip to one's disc. The goal of the game is to maximize the number of discs after the end of game. It is

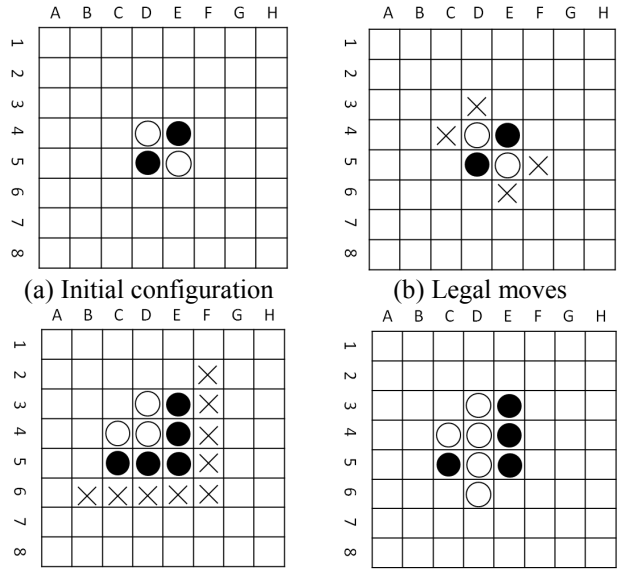
Kyung-Joong Kim is a postdoctoral researcher at the department of mechanical and aerospace engineering, Cornell University, Ithaca, NY, 14850, USA (kk499@cornell.edu)

Sung-Bae Cho is a professor at the department of computer science, Yonsei University, Seoul 120-749, South Korea (sbcho@cs.yonsei.ac.kr)

978-1-4244-2974-5/08/$25.00 ©2008 IEEE

played on an 8×8 board and a game ends when there is no valid move or the board is full. **Figure 1** explains the rule of the Othello. It is difficult because there are many tactics/strategies and drastic change of scores at the end of the game.

It is not surprising that there are annual world Othello championships because it is a challenging game to human. Unlike computers, human relies on pattern recognition, logical thinking and selective attention. Learning the expert-level skills is not an easy task and it requires a lot of time and several thousands of games played with others.



(a) Initial configuration (b) Legal moves

(c) Legal moves (d) After D6 is played

Figure 1. Examples of Othello rules

B. Computational Intelligence for Othello

Othello is a popular game in computational intelligence society because of its complexity. Buro *et al*. developed world-class level Othello programs called Logistello [14]. Lucas *et al*. compared temporal difference learning (TDL) with co-evolutionary learning (CEL) for the game [15]. Runarsson *et al*. investigate the effect of look-ahead depth of a game tree in learning position evaluation functions for Othello. Chong *et al*. use evolutionary algorithms to learn spatial neural networks as an evaluation function for board configuration of Othello [17].

There are many different types of representation for Othello strategy. Lucas proposed *N*-Tuple systems to represent Othello strategy [18]. It outperformed the best strategy in Othello competition at Congress on Evolutionary Computation 2006. In many cases, a simple weight piece counter representation is used [16][15]. It was an 8×8 matrix and each entry has a weight for each square of the board. In [17], a spatial neural network representation was used. 91 sub-boards are extracted from each board configuration and they are inputted to a neural network.

At Congress on Evolutionary Computation 2006, Othello competition was organized and opened to public to submit their evolved strategies. At that time, the only representation was weight piece counter and multi-layer neural networks. Kim *et al*. won the competition with a method of an incremental hybrid learning seeded with strategies learned by TDL [19].

Othello is a deterministic game and small randomness is introduced to increase the number of games between two players. "Deterministic" means that the game score is always the same if it is played in the same sequence. This limits the number of games between two distinct strategies to 2. This is because there is no randomness throughout the game. Lucas *et al*. introduced small randomness in the game of Othello to increase the number of unique games between two players [15]. This was also used in the Othello competition.

Monte-Carlo (MC) algorithms are used in Othello playing. This method is promising in the game of Go. For each move, it simply continues the game by playing random moves until it reaches to the end of the game. Among candidates, the one with the highest winning ratio is chosen as the next move. This is simple but has potential to compete against the traditional min-max search with a game tree. However, this is computationally expensive to get correct statistics. Archer evaluates the MC algorithm with Othello game [20]. Hingston *et al*. evolved a weight piece counter to guide selectively the MC algorithm [21]. Nijssen boosted MC algorithm with domain knowledge and it was competitive against a powerful program WZEBRA (look-ahead depth=1, no opening knowledge) [22].

C. An Ensemble Approach for Evolutionary Games

There are few works on ensemble approaches for evolutionary games. Kim *et al.* used a deterministic crowding genetic algorithm to evolve diverse spatial neural networks for checkers [12]. Distinct strategies are identified with clustering algorithms and they are combined with majority voting. It outperforms the single best player evolved with a standard genetic algorithm. Yang *et al*. proposed a collective decision making of IPD strategies evolved [13]. The final output of the ensemble is calculated with weighted averaging.

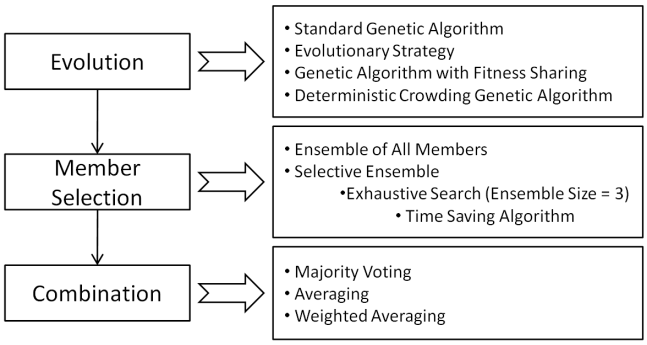


Figure 2. Overview of the ensemble framework

III. AN ENSEMBLE OF EVOLUTIONARY PLAYERS

In this section, each component of the ensemble is introduced step by step (summarized in **Figure 2**). There are many different types of evolutionary algorithms to evolve game strategies. In this paper, we deal with four different algorithms (a standard genetic algorithm (GA), evolutionary strategy, a GA with fitness sharing, and deterministic crowding GA). We compare two different approaches in the member selection of ensembles (the combination of all candidates and selective ensemble). In case of the selective approach, there are a large number of possible ensembles and an efficient heuristic is required to minimize computational cost to find good one. Finally, three representative fusion methods are defined in the game domain (voting, averaging, and weighted averaging).

A. Evolutionary Algorithms

- Standard Genetic Algorithm (SGA): Each strategy is represented with a real-value vector. Selection is done with fitness-proportionate selection (roulette wheel selection). Mutation and crossover are used to generate offspring. Fitness can be calculated statically based on the results of games against well-known heuristics or dynamically against each other.

- Evolutionary Strategy (ES): This method is successful in checkers, Othello, and chess domain [7][8][17]. Each strategy is represented with a real-value vector and additional self-adaptive parameters associated to the vector. It uses only mutation and each parent generates one offspring. Among the pool of parents and offspring, the best half is chosen as the next generation's new parents. The updating rule for the weight and self-adaptive parameter is shown in [7]. Because it is based on tournament selection, it can maintain the diversity of population [17].

- Genetic Algorithm with Fitness Sharing (FSGA): This approach is the same with the SGA except that it readjusts the fitness based on the similarity of individuals [23]. The original fitness of the individual is f and it is readjusted to f_s. N is the population size and σ is a sharing radius. $d(i)$ is the distance with the ith strategy.

$$f_s = \frac{f}{\sum_{i=0}^{N} sh(i)} \qquad sh(i) = \begin{cases} 1 - \dfrac{d(i)}{\sigma} & d(i) \le \sigma \\ 0 & d(i) > \sigma \end{cases}$$

- Deterministic Crowding Genetic Algorithm (DCGA) (**Figure 3**): It is similar to evolutionary strategies but parents compete against their children [24]. Two parents generate two offspring with genetic operators (crossover and mutation). From a pair of similar parents and child, the one with higher fitness is chosen for the next generation's new parents.

B. Member Selection

The members of an ensemble are chosen from the population of the last generation and it is not trivial to choose the members for an ensemble because of its huge ensemble search space. If the population size is N, there are N candidates for the member of the ensemble. A straightforward approach is to combine all candidates available. It generates one ensemble of N members. A subset of N members would lead to 2^N possible ensembles. If the size of the ensemble is fixed as M, the number of possible ensembles is ${}_N\mathrm{C}_M$. Clustering algorithms are used in [10][12][25]. From each cluster, the best player is chosen as a representative one and the final ensemble is formed with the representatives.

In this paper, two approaches are adopted and compared. The first one is combining all individuals in the population of the last generation. The next one is enumerative search of the ensemble candidates with fixed size. In case of ensemble size = 3, the total number of ensembles is ${}_N\mathrm{C}_3$. The best one among all candidates is chosen as a final ensemble. Although there is no additional computational cost to form an ensemble for the first approach, it is known that the combination of all may not be better than one of the subsets of them. Meanwhile, the second approach requires a lot of computational cost to enumerate all ensembles and evaluate them.

```
// P : The population of game strategies
// p_i : ith individual of P
// fitness(p_i) : Return the fitness of p_i
// d(p_i,p_j): Return the distance between p_i and p_j
// shuffling() : Randomly rearrange the order of individuals
// survive(p_i): p_i survives to the next generation

FOR (gen = 0; gen < MAX_GEN; gen++) {
  Shuffling();
  FOR (I = 0; i < POP_SIZE; I += 2) {
    c1,c2=crossover(p_i, p_i+1);
    c1=mutation(c1); c2=mutation(c2);
    IF ((d(p_i,c1)+ d(p_i+1,c2))< (d(p_i,c2)+ d(p_i+1,c1)) {
      IF (fitness(p_i) > fitness(c1)) survive (p_i);
      ELSE survive (c1);
      IF (fitness(p_i+1) > fitness(c2)) survive (p_i+1);
      ELSE survive (c2);
    }
    Else {
      IF (fitness(p_i) > fitness(c2)) survive (p_i);
      ELSE survive (c2);
      IF (fitness(p_i+1) > fitness(c1)) survive (p_i+1);
      ELSE survive (c1);
} }}
```

Figure 3. A pseudo code for DCGA

C. Exhaustive Ensemble Searching with Multi-Stage Evaluations for Time Saving

In this paper, a new multi-stage evaluation method is proposed to reduce computational cost in enumerative

search of an ensemble (**Figure 4**). The bottleneck of the exhaustive search is the evaluation of an ensemble by playing a number of games. The accuracy of the evaluation is related to the number of games played by the ensemble. If G is the games played, the total computational cost for the enumerative search is ${}_N C_M \times G$. In the new evaluation method, each ensemble is evaluated from low accuracy to high accuracy step by step. If it is identified that the ensemble is not better than the best one in low accuracy evaluation, it passes the high accuracy evaluation and goes to the next ensemble.

```
// T : The total number of possible ensembles
//   : Individuals are sorted based on their performance
//   : Enumeration is based on the sorted order
// evaluate (i,G) : Return a scoring point from G games
// B : The best ensemble found
// B_s : The scoring point of B
FOR (i = 0; i < T; i++) {
  FOR (g = MIN_G; g ≤ MAX_G; g×=STEP_SIZE) {
    S=evaluate(i,g);
    IF( S/g < B_S/MAX_G ) break; // normalized comparison
    ELSE IF(g==MAX_G) { B=i;B_s=S}
}}
```

Figure 4. A pseudo code of enumerative search based on the multi-stage evaluation (Scoring point = the number of win + the number of draw×0.5)

D. Fusion Methods

In this paper, three representative fusion methods are defined in the domain of game. The ensemble is defined as $E=\{m_1,m_2,\ldots,m_M\}$, where it is composed of M members. Let's assume that there are L legal moves by the ensemble player at current board configuration. In a majority voting method, each member of the ensemble votes for one of L legal moves. The one with the highest vote is decided as the next move of an ensemble. In an averaging method, the evaluation value for each legal move is averaged over all members of the ensemble. In a weighted averaging method, their contribution to the averaging is weighted based on a prior knowledge on each member's performance.

IV. Experimental Results on Othello Game

TABLE 1 summarizes the parameters used in the experiments. The population sizes of the SGA and FSGA are double the ones of ES and DCGA. In SGA and FSGA, only parents are evaluated but in ES and DCGA, the parents and offspring are evaluated together. For a fair comparison, the population size is adjusted. In FSGA and DCGA, Euclidean distance of WPC is used as a distance measure. In FSGA, the sharing radius is decided as the half of the average distances among all individuals. The final results are average of 10 runs. The size of the ensemble is fixed to 3.

Othello is used as a game to test the ensemble approaches. Because the game is deterministic, 10% randomness is applied in the move selection [15]. In evolution stage, the scoring point (# of wins + # of draw×0.5) against standard heuristic [26] is used as a fitness function. In G games, the choice of color is even (half of the game is played with black and the remaining is played with white).

TABLE 1. PARAMETERS OF THE EXPERIMENTS

(a) Parameters dependent on evolutionary algorithm

	SGA	ES	FSGA	DCGA
Population Size	20	10	20	10
Crossover Rate	0.8	-	0.8	0.8
Mutation Rate	0.1	1.0	0.1	0.1

(b) Common parameters

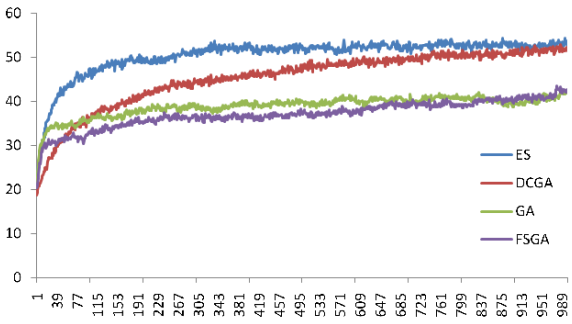
Evaluations per one Generation	20
G in Evolution	100
Randomness (ε) in Move Selection	0.1
Maximum Generation	1000
# of Runs	10
MIN_G, MAX_G, STEP_SIZE in the Ensemble Search	100,10000,10
Size of an Ensemble	3

*G: The number of games played in the evaluation

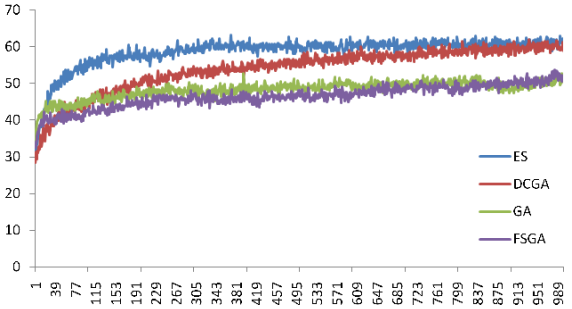
Computational cost is one of the important issues in this research. Weight piece counter (WPC) is chosen as a representation because it is very fast to calculate evaluation. It is possible to use multi-layer neural networks or spatial neural networks as a representation but it is much slower than one of WPC. For one run, the number of games played is $20\times100\times1000=2\times10^6$. To minimize the computational cost for the exhaustive ensemble searching, the best 10 individuals from the last generation is chosen. The possible number of ensembles is ${}_{10}C_3=\frac{10!}{3!}=120$. For accurate evaluation, in ensemble searching, 10000 games are played for each candidate. If there is no time saving heuristic, the games played for the exhaustive search is $120\times10000=1.2\times10^6$.

Figure 5 shows the average and maximum fitness of the four evolutionary algorithms. ES and DCGA are better than GA and DCGA. GA converges in the early stage of evolution. FSGA is a bit worse than the GA until 800 generations but it is nearly the same at 1000 generations. ES converges at 400 generations but DCGA steadily increases its fitness. Explicit mechanism to increase diversity causes slow convergence of evolution.

TABLE 2 summarizes the max, average and min of the individuals in the last generation. It shows that ES is the best one and the second is DCGA. SGA and FSGA are worse than the two methods. There is difference between SGA and FSGA. In ES, every individual obtains high score and there is small difference between max and min. However, in DCGA, there is a big difference between max and min.



(a) Average fitness



(b) Maximum fitness

Figure 5. Progress of the evolution (Y-axis is a fitness)

TABLE 2. THE SCORING POINT OF THE INDIVIDUALS IN THE LAST GENERATION
(10000 games against heuristic, ε =0.1)

	MAX	AVG	MIN
SGA	4225±481	4125±387	4249±427
ES	5551±327	5343±366	5055±393
FSGA	4248±429	4107±570	4077±592
DCGA	5434±347	5181±376	4590±689

The best individual scores 6315. It comes from evolutionary strategies (**Figure 6**). In a positional evaluation like WPC, the corners are very important and the next square to the corners are dangerous area. This WPC reflects this idea very well. Because ES allows continuously growing its weight value, there is big weight (90.87344).

90.87344	-0.26256	-0.60127	-2.54618	-0.18856	0.517852	-0.35238	-0.02166
-182.335	0.007959	0.284569	-0.11609	34.9188	4.939041	0.599046	0.334568
0.086362	-308.841	0.186365	-0.44385	4.671535	0.550587	-0.01158	0.292283
-0.91666	0.026851	-0.07039	0.052792	-0.31023	0.099765	-1.43215	-0.05711
2.184365	-521.709	-0.49365	-0.56033	3.860042	0.25989	0.071583	-0.91255
-53.9194	-2023.94	1.452537	0.12882	-10.9312	16.20616	-3.07895	-1.65374
-100.435	19.0072	-6.99013	-7.40195	-17.6748	-7.31492	-3.88584	0.346161
8.413686	-1.34717	-0.90809	0.988112	-6.86644	1.882877	-3.55869	-9.36959

Figure 6. WPC that scores the best (6315)

TABLE 3 summarizes the uniqueness and average distance of the population of the last generation. For SGA and FSGA, the best 10 individuals are chosen from 20. Uniqueness is defined as the number of unique individuals. The average distance is calculated from the sum of distances from 10×9 pairs. The ES and DCGA show the highest uniqueness (10). SGA and FSGA maintain 4~5 unique individuals and there are identical individuals in the population. FSGA is a bit better than SGA in terms of uniqueness. In ES, the real-value in WPC can increase continuously and the average distance is too high. In SGA, FSGA, and DCGA, the real-value ranges from 0 to 1.0. DCGA maintains higher average distance than SGA and FSGA.

TABLE 3. UNIQUENESS AND AVERAGE DISTANCE ANALYSIS

	Uniqueness	Average Distance
SGA	4.7±1.8	0.9±0.5
ES	10.0±0.0	1192169±2558915
FSGA	5.2±1.9	0.9±0.7
DCGA	10.0±0.0	3.7±1.9

TABLE 4. THE PERFORMANCE OF THE BEST ENSEMBLE FOUND AGAINST STANDARD HEURISTICS
(10000 games, ε =0.1)

(a) Performance of the ensemble of all

	Best Single Individual	Majority Voting	Averaging	Weighted Averaging
SGA	4225±481	4266±422	**4286±385**	4279±428
ES	**5551±327**	5462±349	5401±410	5406±405
FSGA	4248±429	4327±444	4313±451	**4338±474**
DCGA	**5434±347**	5366±351	5286±471	5277±504

(b) Performance of the ensemble of three members

	Best Single Individual	Majority Voting	Averaging	Weighted Averaging
SGA	4225±481	4401±417	**4421±415**	4415±420
ES	5551±327	**5627±338**	5621±351	5606±356
FSGA	4248±429	4466±474	**4476±458**	4460±452
DCGA	5434±347	5538±322	5544±297	**5550±317**

(c) Computational cost for exhaustive search (10 runs, 3 fusion methods)

	# of Games (without time saving) (A)	# of games (with time saving) (B)	Gain (A/B)
SGA	3.6×10^7	4.73×10^6	7.6
ES	3.6×10^7	3.43×10^6	10.5
FSGA	3.6×10^7	4.76×10^6	7.5
DCGA	3.6×10^7	3.30×10^6	10.9

A. Ensemble against the Heuristics

The exhaustive searching for the ensemble is done with the time saving algorithm. It is compared with the combination of all individuals. The criterion to select the best ensemble is the performance against the standard heuristics. In

the weighted averaging, the weight of the individuals is decided based on the performance against the standard heuristics (10000 games, ε =0.1).

In **TABLE 4**, the performance of an ensemble is summarized with computational cost comparison. The combination of all individuals is not always better than the single best individual from the last generation. In SGA and FSGA, the ensemble performs better than the single best one but it is not true in ES and DCGA. The ensemble of 3 members outperforms the best single individual and the combination of all individuals. It is not clear which fusion method is the best. In the exhaustive ensemble search, total number of games is 10 runs × 3 fusion methods × 1.2×10^6 $=3.6\times10^7$. The number of games is 7~10 times smaller than the original one when the multi-stage evaluation approach is used. In ES and DCGA, the cost gain is bigger than one of GA and FSGA.

The best ensemble scores 6425. It is from evolutionary strategies with the weighted averaging method. **Figure 7** shows the weight piece counter matrix for three members in the best ensemble. The rank of the three individuals is 1st, 2nd and 7th among 10 individuals in the last generation. The average score of the members is 6220 and the performance gain from the ensemble is 205 games.

Figure 8 shows a new WPC derived from the three WPC's. If the entry in 1st WPC is x_1 and the weight for the WPC is w_1, the new WPC is defined as follows.

$$x = x_1 \times w_1 + x_2 \times w_2 + x_3 \times w_3$$

If the weighted averaging requires three evaluations per move, the new WPC results in the same output with one evaluation per move. This is also available to the averaging fusion method.

90.87344	-0.26256	-0.60127	-2.54618	-0.18856	0.517852	-0.35238	-0.02166
-182.335	0.007959	0.284569	-0.11609	34.9188	4.939041	0.599046	0.334568
0.086362	-308.841	0.186365	-0.44385	4.671535	0.550587	-0.01158	0.292283
-0.91666	0.026851	-0.07039	0.052792	-0.31023	0.099765	-1.43215	-0.05711
2.184365	-521.709	-0.49365	-0.56033	3.860042	0.25989	0.071583	-0.91255
-53.9194	-2023.94	1.452537	0.12882	-10.9312	16.20616	-3.07895	-1.65374
-100.435	19.0072	-6.99013	-7.40195	-17.6748	-7.31492	-3.88584	0.346161
8.413686	-1.34717	-0.90809	0.988112	-6.86644	1.882877	-3.55869	-9.36959

(a) Individual score = 6315

114.7926	-0.25153	-0.55609	-0.21261	-0.1886	0.517915	-0.5832	-0.02162
-165.244	-0.22415	0.072315	-0.11475	31.16748	1.048603	0.59952	0.590148
0.086061	-134.127	0.18663	-0.18699	4.664459	0.550608	0.020289	0.290606
-1.42674	0.02416	0.00516	0.026828	-0.31377	0.099659	-1.46347	-0.05524
2.183821	-629.879	-0.50517	-0.43393	3.483493	0.266898	0.071782	-0.91376
-52.1856	-1476.7	1.444749	0.223989	-10.5789	16.26362	-3.08034	3.264987
-120.47	14.18692	-6.46894	-13.0218	-12.9431	-13.6865	-6.26634	0.331853
3.036566	-1.36676	-1.36427	0.969968	-6.02968	1.50647	-2.73418	-9.18163

(b) Individual score = 6131

82.56643	-0.26157	-0.57237	-1.63131	-0.1886	0.518737	0.21246	-0.02167
-272.135	-0.60232	-0.01443	-0.10992	35.65144	2.586812	0.59853	0.619492
0.085876	-163.869	0.188045	-0.41777	4.653471	0.550572	0.008109	0.288663
-0.81888	0.026788	-0.06804	0.045281	-0.30821	0.098399	-1.4419	-0.04668
2.184586	-508.09	-0.47791	-0.56385	3.43172	0.269811	0.07245	-0.91287
-54.4103	-1303.54	1.219795	-0.07662	-11.5493	16.15825	-3.0785	-2.64743
-98.8229	15.93903	-6.32784	-6.813	-15.0184	-9.85218	-10.2493	0.350946
4.692565	-1.31076	-0.79334	0.985993	-7.04528	1.946833	-3.69742	-9.23639

(c) Individual score = 6214

Figure 7. The ensemble that scores the best (6425)

B. Ensemble against the Best Single Individual

TABLE 5 summarizes the performance of ensembles against the best single individual. The best ensemble is found by an exhaustive search with the time saving heuristics. The ensemble size is fixed as three in the search. If the scoring point is larger than 5000, it means that the ensemble outperform against the single best player. In the weighted averaging, the weight is decided based on the performance against standard heuristics (10000 games, ε =0.1). The combination of all individuals is worse than the single best one. In the ensemble of three members, the ensemble outperforms against the single best player with any fusion methods. Like the previous results, it is not clear which fusion method is superior. In ES and DCGA, the ensemble gains more score than the one from GA and FSGA. The computational cost gain from the time saving algorithm is approximately 8~18.

95.96609	-0.25861	-0.5768	-1.47479	-0.18859	0.518167	-0.24012	-0.02165
-206.624	-0.27153	0.11526	-0.1136	33.93023	2.877463	0.59903	0.513425
0.086101	-203.159	0.187012	-0.35077	4.663195	0.550589	0.005448	0.290526
-1.05169	0.025946	-0.04479	0.04176	-0.31072	0.099275	-1.44569	-0.05302
2.18426	-552.714	-0.49219	-0.51997	3.593685	0.265496	0.071937	-0.91306
-53.5132	-1604.23	1.372472	0.091674	-11.0213	16.20908	-3.07926	-0.36854
-106.481	16.40169	-6.59834	-9.0523	-15.2355	-10.2533	-6.78709	0.343053
5.40778	-1.34148	-1.01976	0.981445	-6.65107	1.780501	-3.33398	-9.26348

Figure 8. New WPC derived from the three members in the best ensemble

TABLE 5. THE PERFORMANCE OF THE BEST ENSEMBLE FOUND AGAINST THE BEST SINGLE INDIVIDUAL

(10000 games, ε =0.1)

(a) Performance of the ensemble of all

	Majority Voting	Averaging	Weighted Averaging
SGA	**4988±330**	4923±352	4917±359
ES	**5087±612**	4914±432	4918±442
FSGA	**4984± 85**	4966±122	4958±131
DCGA	4921±173	4925±380	**4941±350**

(b) Performance of the ensemble of three members

	Majority Voting	Averaging	Weighted Averaging
SGA	5152±269	**5211±306**	5211±307
ES	**5483±420**	5462±457	5437±393
FSGA	**5202±214**	5195±190	5194±167
DCGA	5430±282	**5606±417**	5599±407

(c) Computational cost for exhaustive search (10 runs, 3 fusion methods)

	# of Games (without time saving) (A)	# of games (with time saving) (B)	Gain (A/B)
SGA	3.6×10^7	3.80×10^6	9.5
ES	3.6×10^7	2.20×10^6	16.3
FSGA	3.6×10^7	4.28×10^6	8.4
DCGA	3.6×10^7	1.98×10^6	18.1

C. Discussion

Diversity is related to the success of the evolution. In the uniqueness and average distance analysis show the reason of success by the ES and DCGA. They maintain higher uniqueness and diversity than SGA and FSGA. In the analysis of the population of the last generation, ES and DCGA outperform the SGA and FSGA.

Diversity and good base member is a key in the success of the ensemble. ES and DCGA maintain high diversity and good individuals. It leads to the good ensembles and they outperform SGA and FSGA. Because SGA and FSGA have individuals with less fitness and the diversity is low, there is limitation to get comparable results with the ES and DCGA.

Computational cost issue is important in game evolution. It is possible to get better performance with the introduction of complex representation (Multi-layer neural networks, and spatial neural networks) but they takes a lot of time. Also, the ply-depth can be increased but it increases computational cost significantly. In the ensemble searching, the time saving algorithm is essential to get results in a reasonable time. The factor is approximately 7~18. Although this time saving sacrifices the accuracy of the enumerative search, it is important to get good ensemble in a reasonable time.

V. Conclusions and Future Works

There is performance gain from forming an ensemble of game strategies evolved. From the experiment on Othello game, the ensemble from the population of the last generation can outperform the best individual player. Selective ensembles are better than the one of all individuals. The use of evolutionary algorithm and the choice of the member is directly related to the success of the ensemble.

The enumerative search of ensemble can be improved with other techniques. Genetic algorithms are used to search for ensemble of several classifiers for bioinformatics problem [27]. In this method, there is no restriction on the size of the ensemble. Greedy approach is one of the techniques to form an ensemble and they can be applied to the game domain [28]. It starts from an empty ensemble and adds the member that maximizes the performance increase. On the other hand, it is possible to start from full ensemble and delete one member at a time in a greedy manner.

The speed of the evolution is dependent on the time required to do a game between two strategies. A bit-board representation could be used to reduce the time for one game. In the representation, each entry of the board is represented as two bits and bitwise operators are used to update the board. Another way to increase the speed of evaluation is to use distributed computing. GPU (Graphical Processing Unit) in a graphic card is ready to do highly parallel computing with less expensive hardware. Multi-core machines can be used to accelerate the speed of evolution.

The strategies from co-evolution can be benefit from the ensemble approach. In this work, we only consider the evolution against static heuristic player and the final solution has less generalization ability. Co-evolution is promising to increase winning ratio against unseen strategies. The same enumerative ensemble searching can be used to the population of the last generation of the co-evolution.

Acknowledgments

This research was supported by MKE, Korea under ITRC IITA-2008-(C1090-0801-0046).

References

[1] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21-45, 2006.

[2] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, Wiley, 2004.

[3] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: A survey and categorization," *Information Fusion*, vol. 6, no. 1, pp. 5-20, 2005.

[4] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artificial Intelligence*, vol. 137, no. 1, pp. 239-263, 2002.

[5] A. Verikas, A. Lipnickas, K. Malmqvist, M. Bacauskiene, and A. Gelzinis, "Soft combination of neural classifiers: A comparative study," *Pattern Recognition Letters*, vol. 20, no. 4, pp. 429-444, 1999.

[6] S. M. Lucas, "Computational intelligence and games: Challenges and opportunities," *International Journal of Automation and Computing*, vol. 5, pp. 45-57, 2008.

[7] K. Chellapilla, and D. Fogel, "Evolution, neural networks, games, and intelligence," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1471-1496, 1999.

[8] D. B. Fogel, T. J. Hays, S. L. Hahn, and J. Quon, "A self-learning evolutionary chess program," *Proceedings of the IEEE*, vol. 92, no. 12, pp. 1947-1954, 2004.

[9] S. Lucas, and G. Kendall, "Evolutionary computation and games," *IEEE Computational Intelligence Magazine*, pp. 10-18, Feb 2006.

[10] K.-J. Kim, and S.-B. Cho, "Evolutionary ensemble of diverse artificial neural networks using speciation," *Neurocomputing*, vol. 71, no. 7-9, pp. 1604-1618, 2008.

[11] X. Yao, and Md. M. Islam, "Evolving artificial neural network ensembles," *IEEE Computational Intelligence Magazine*, vol. 3, no. 1, pp. 31-42, 2008.

[12] K.-J. Kim, and S.-B. Cho, "Systematically incorporating domain-specific knowledge into evolutionary speciated checkers players," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 615-627, 2005.

[13] S.-R. Yang, and S.-B. Cho, "Co-evolutionary learning with strategic coalition for multiagents," *Applied Soft Computing*, vol. 5, pp. 193-203, 2005.

[14] M. Buro, "How machines have learned to play Othello," *IEEE Intelligent Systems*, vol. 14, no. 6, pp. 12-14, 1999.

[15] S. M. Lucas, and T. P. Runarsson, "Temporal difference learning versus co-evolution for acquiring Othello

position evaluation," *IEEE Symposium on Computational Intelligence and Games*, pp. 52-59, 2006.

[16] T. Runarsson, and E. O. Jonsson, "Effect of look-ahead search depth in learning position evaluation functions for Othello using ε -greedy exploration," *IEEE Symposium on Computational Intelligence and Games*, pp. 210-215, 2007.

[17] S. Y. Chong, M. K. Tan, and J. D. White, "Observing the evolution of neural networks learning to play the game of Othello," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 240-251, 2005.

[18] S. M. Lucas, "Learning to play Othello with N-tuple systems," *Australian Journal of Intelligent Information Processing*, vol. 4, pp. 1-20, 2008.

[19] K.-J. Kim, H-J. Choi, and S.-B. Cho, "Hybrid of evolution and reinforcement learning for Othello players," *IEEE Symposium on Computational Intelligence and Games*, pp. 203-209, 2007.

[20] R. Archer, *Analysis of Monte Carlo Techniques in Othello*, B.S. Thesis, The University of Western Australia, 2007.

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

[22] P. Nijssen, *Playing Othello using Monte Carlo*, B.S. Thesis, Universiteit Maastricht, Netherland, 2007.

[23] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[24] T. Baeck, D. B. Fogel, and Z. Michalewicz, *Evolutionary Computation 2: Advanced Algorithms and Operators*, Taylor & Francis, 2000.

[25] Y. Liu, X. Yao and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 380-387, 2000.

[26] T. Yoshioka, S. Ishii, and M. Ito, "Strategy acquisition for the game "Othello" based on reinforcement learning," *IEICE Transactions on Information and Systems*, E82-D 12, pp. 1618-1626, 1999.

[27] K.-J. Kim and S.-B. Cho, "An evolutionary algorithm approach to optimal ensemble classifiers for DNA microarray data analysis," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, pp. 377-388, 2008.

[28] I. Partalas, G. Tsoumakas, E. Hatzikos, and I. Vlahavas, "Ensemble selection for water quality prediction," *Proceedings of the 10th International Conference on Engineering Applications of Neural Networks*, pp. 428-435, 2007.