Proceedings of the 2007 IEEE Symposium on
Computational Intelligence and Games (CIG 2007)

Hybrid of Evolution and Reinforcement Learning for Othello Players

Kyung-Joong Kim, Heejin Choi and Sung-Bae Cho
Dept. of Computer Science, Yonsei University
134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, South Korea
kjkim@cs.yonsei.ac.kr, heinch@naver.com, sbcho@cs.yonsei.ac.kr

***Abstract*—Although the reinforcement learning and evolutionary algorithm show good results in board evaluation optimization, the hybrid of both approaches is rarely addressed in the literature. In this paper, the evolutionary algorithm is boosted using resources from the reinforcement learning. 1) The initialization of initial population using solution optimized by temporal difference learning 2) Exploitation of domain knowledge extracted from reinforcement learning. Experiments on Othello game strategies show that the proposed methods can effectively search the solution space and improve the performance.**

Keywords: Othello, Reinforcement Learning, Temporal Difference Learning, Domain Knowledge

I. Introduction

Reinforcement learning [1] and evolutionary algorithm [2] are separately used to learn the strategies of board games. Although evolutionary algorithm is known for good performance in games, they require much computational resource compared to the reinforcement learning. Meanwhile, reinforcement learning can learn strategies quickly with relatively less computational resources. The hybridization of the both methods can improve the pure evolutionary algorithm for optimizing game evaluation function [3].

Because reinforcement learning can find a good solution quickly with less resource, it is promising to exploit reinforcement learning first and pass the results to the evolutionary algorithm for further optimization. In CEC 2006 Othello competition, the hybrid of evolutionary algorithm and temporal difference learning method won the final league [4]. It showed relatively high generalization ability compared to other models using either temporal difference learning or evolutionary algorithm. The method used in the best player is to exploit the best individual from the temporal difference learning as a seed for the evolutionary algorithm.

It is also promising to use domain knowledge extracted from reinforcement learning like self-playing in the evolutionary process. It is known that the incorporation of domain knowledge is useful for the pure evolution to improve the performance [5][6][7]. Its idea is to exploit previously easily accessible domain knowledge to leverage the pure evolutionary approach. Opening list, opening DB, endgame DB, and transcripts of previous games can be used as domain knowledge. Addition of such knowledge might minimize the evolution time and quality of final output. Because the domain knowledge could restrict the search space to be explored, it is expected that the evolutionary algorithm can find good solution easily and fast.

Othello is a very short game that requires only 60 moves by both players. Because of this, the importance of opening is very important. Slight advantage in the early stage of game often becomes huge difference in the end of the game. Although advantage in the early stage doesn't mean win of the game, it is true that the player with the advantage have more probability of winning. Also, the game is very difficult to estimate the results of the final score because there is huge fluctuation in the score at the end game stage. Expert players investigate all possible lines from the current board configuration and decide the best line at the endgame stage.

Strong Othello programs like LOGISTELLO [8], NTEST [9] and WZEBRA [10] have their own opening book. It contains pre-calculated evaluation value for each move in the early stage of the game. The value is calculated from the self-playing of thousands of games. For each game, the final score is used to evaluate the opening used in the game. If the game is finalized with 34-30 as Black win, the opening used is evaluated with +4 for Black. Although this is a bit different from temporal difference learning, it is similar that the knowledge is from self-playing of one player.

We propose two methods for hybrid the reinforcement learning and evolutionary algorithm. 1) Initialization of the population in evolutionary algorithm using solution optimized by temporal difference learning 2) Evolution of Othello strategies using opening book from self-playing and endgame solver that quickly calculates the goodness of the position in the endgame stage.

II. Related Works

There are many publications about the learning of game strategies using evolutionary computation. They can be categorized into pure evolution, the hybrid of evolutionary algorithm with domain knowledge, and hybrid of reinforcement learning with evolution.

The most successful example of the pure evolutionary approach for the game is the Fogel's checkers program [2].

They have applied evolutionary neural network for the evaluation of checkers. Without help of domain knowledge, they can evolve master-level player and evaluate the performance in the game site.

In Othello, Miikkulainen et al. applied neural network optimized by genetic algorithm without game tree [11]. They showed that their evolved neural network learnt the mobility strategy and the world-class level player checked the transcripts of the play with comments.

Chong *et al*. evolved neural networks as an evaluation function in the game tree for Othello [12]. They observed the evolution of the neural networks based on the winning rates against static strategies. They reported that the evolutionary neural networks can improve their performance through the evolution. Also, they evaluated the effect of the spatial preprocessing layer, self-adaptive mutation, and tournament selection.

Kim *et al*. used opening knowledge (well-known opening list) and endgame DB (from Chinook) for evolving checkers strategies [5]. In the middle stage of the game, speciation algorithm is used to generate diverse evolutionary neural networks for the evaluation of the leaf node in the game tree. They reported that the incorporation of expert knowledge can speed up the evolution and improve the performance.

Kim *et al*. used opening knowledge (well-known opening list) to boost the performance of evolutionary Othello players [6]. Opening list summarized by human experts is used in the early stage of each game played in the evolution. The experimental results show that the evolution with the opening knowledge show improved performance. Because they used position-based evaluation of board configuration, it is not possible to achieve comparable performance to the other programs. Also, they used only 1-ply game tree for the middle stage of the game.

Fogel *et al*. used opening database and endgame database to evolve the evolutionary chess players. They used three object neural networks that cover different areas of chess boards. Also, they used material values of pieces and positional value tables. The evolved strategies showed good performance compared to previous knowledge-based players.

The relationships between the reinforcement learning and the evolutionary algorithm are one of the interesting research issues. The both methods are compared separately or combined for synergism.

Lucas *et al*. compared two learning methods for acquiring position evaluation for small Go boards [13]. The methods studied are temporal difference learning using the self-play gradient-descent method and co-evolutionary learning using an evolutionary strategy. They concluded that the temporal difference learning usually performs better than the co-evolutionary algorithm in the standard setup. However, in the right configuration, the co-evolutionary algorithm performs better than the counterpart.

Lucas *et al*. compares the use of temporal difference learning (TDL) versus co-evolutionary learning (CEL) for acquiring position evaluation functions for the game of Othello [14]. For Othello, they reported that TDL learns much faster than CEL, but that properly tuned CEL can learn better playing strategies.

Singer proposed the hybridization of evolutionary algorithm and reinforcement learning for Othello game strategy acquisition [3]. In each generation, reinforcement learning is used to train the individual of the population. They reported that the strategy evolved for 3 months played at roughly intermediate level.

III. METHODS

A. The Game of Othello

Othello is a deterministic game which is played by two players. It is usually played on 8×8 boards and there are 64 squares. It is a kind of perfect information game and both players have no hidden information. Each disc is similar to coin but each side has different colors. One is white, the other is black. At the initial stage of the game, both players choose his color. If one player chooses white, the other player is black. The initial board configuration is shown figure 1. Initially, four discs are placed in the center of the board.

The game always starts with black player. The rule of the game is very simple. The only rule is sandwiching other player's discs by using his discs and flipping the discs sandwiched to his disc color. The capturing is possible in any direction and multiple directional capturing is also available. The game is continued until there is no available move for both players. At the last stage of the game, the one with more discs wins the game. If there is equal number of discs, the game called as draw.

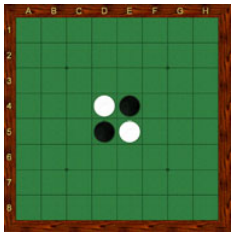

Figure 1. The initial board configuration (Image from WZEBRA)

B. Overview of the Proposed Methods

The proposed method is composed of two stages. At first, temporal difference learning is used to find premature solution. It discovers the area that has many high performance solutions. After finding the useful solution, it is used to find better strategies using evolutionary algorithm. The knowledge from the first strategy can be stored in a different form for the evolutionary search. For example, they (the knowledge from reinforcement learning) are weights of neural networks or opening DB. The evolutionary algorithm can be many different forms. Both of co-evolutionary algorithm and evolution with fixed evaluation function can be used.

The knowledge from the self-playing can be used to initialize the population of the evolutionary algorithm and it also can be directly used in the evaluation function of the evolution. By exploiting previously discovered knowledge from reinforcement learning, the evolutionary algorithm can discover better strategy.

C. Initialization of Population Using TDL Results

This section is related to the CEC 2006 competition and introduces the method used for winning the award [4]. The purpose of the competition is to promote the research on a new method for evolving Othello strategy. Othello is very complex game and enough to be used as a platform for many variants of evolutionary algorithms. They provided two different forms of strategy representation: weight matrix for positional strategy and multi-layer perceptrons. The preliminary round, the submitted strategies are evaluated using static opponent with standard heuristics. Because the game is a kind of deterministic one, there are only two different games between two strategies. To increase the number of games between them, 10% randomness is added to the selection of moves for both players.

After playing 1000 games with the static strategy, the number of win, draw and loss are used to calculate score. Based on the score, the players are ranked. Because they are ranked based on the results against the static strategy, it is expected that player biased to the static strategy would get high rank. The final winner is determined from the competition among the best players from each person (final round). In this stage, the player that has more generalization capability against other best players of each person will get high probability of win the competition. In the competition, it is not allowed to see other player's strategy and thus it was not possible to create a strategy specifically tailored to be superior to the other submitted strategies.

```
 1: /* TDL_B: The best strategy learnt from TDL */
 2: /* POP: Population of evolutionary search */
 3: /* POP[i] : ith individual of the population */
 4: /* POP_SIZE : Population size */
 5: /*MAX_GEN : The maximum number of generation*/
 6:
 7: FOR (i=1;i<POP_SIZE;i++) { POP[i]=TDL_B;}
 8:
 9: FOR (i=1;i<MAX_GEN;i++) {
10:   fitness_evaluation(POP);
11:   roulette_wheel_selection(POP);
12:   /* mutation */
13:     FOR(j=1;j<POP_SIZE;j++){
14:     FOR (all segments of POP[j]) {
15:       IF(rand() < mutation_probability)
16:         IF(rand()%2==0){
17:           Segment of POP[j]+=0.01;}
18:         ELSE{
19:           Segment of POP[j]–= 0.01;}}}
20:   elitist(POP); }
```

Figure 2. The pseudo code of the hybrid algorithm.

In the competition, we have used a hybrid of temporal difference learning and evolution for learning strategy. Temporal difference learning is a kind of reinforcement learning [15]. The strategy that is discovered from the temporal difference learning is used as a seed of evolutionary search. The pseudo code of the proposed method is described in figure 2. Temporal difference learning is useful to learn strategy fast but there is still room for adjusting the parameters of TDL results using evolutionary algorithm. Lucas *et al*. mentioned that evolutionary algorithm could produce better results compared to TDL but requires more computational resources and tuning [13][14]. The proposed algorithm can save the time for evolutionary algorithm by exploiting TDL that learns a good strategy quickly.

D. Exploiting Knowledge from Self-Playing

Previously, we have used well-summarized opening list in the process of evolutionary Othello player [6]. The list has 76 openings that are frequently used by human players. It has only the name and the sequence of the openings. There is no evaluation value for each opening. Also, it is limited to the most popular openings and it cannot deal with variations of popular openings. Strong Othello programs have their own opening books that cover huge number of opening lines. They learn the opening book automatically from their self-playing games and transcripts of top-players [16]. They adjust and expand the opening book based on the results of the game.

There are two ways to construct opening books for strong Othello programs [17][18]. The first method is manually constructing books with the help of experts and transcripts. It selects popular and important opening lines manually. The second method is based on the results of the huge number of games. If the game of result is loss, the opening used get negative reward. The assumption of this approach is that the result of the game is largely related to the selection of opening and errors on the other stages have relatively low effect on the results. However, this assumption is not true for real-world situation. Although the player selects bad opening, it can make a win by the mistake of the other players at the end of the game. The way to overcome this shortcoming is to use self-play of strong programs with high depth because it makes relatively low error and reveals the effect of openings clearly.

In this paper, the opening knowledge from self-play and games between top players is used in the process of evolution. The knowledge can be regarded as results of reinforcement learning. The results of games are used to give reward of openings. By playing more games, the relevance of openings are continuously updated based on reward value. If the knowledge can be exploited, the scope that evolutionary algorithm covers is minimized. Furthermore, endgame solver can calculate the results of the game perfectly and quickly. Both of the knowledge can significantly reduce the complexity of learning Othello players. Figure 3 summarizes the pseudo code of the knowledge-incorporated evolutionary algorithm.

```
 1: /* OPENING: Opening knowledge from self-play */
 2: /* ENDGAME: Endgame solver */
 3: /* POP: Population of evolutionary search */
 4: /* POP[i] : ith individual of the population */
 5: /* POP_SIZE : Population size */
 6: /*MAX_GEN : The maximum number of generation*/
 7:
 8: FOR (i=1;i<MAX_GEN;i++) {
 9:    Offspring = mutate (POP);
10:    FOR (j=1;j<POP_SIZE*2;j++){
11:       index_list=select_opponents(POP,Offspring);
12:        /* do game between j and index_list */
13:       FOR(k=1;k<60;k++){
14:             IF(current sequence is not out-of-opening)
15:             OPENING;
16:             ELSE IF(empty squares < threshold)
17:             ENDGAME;
18:             ELSE
19:             execute_game_tree();
20:       }
21:    }
22:    POP=select(POP+Offspring);
23: }
```

Figure 3. Knowledge incorporated evolutionary algorithm.

IV. Experimental Results

A. Hybrid of TDL and Evolution

Kim *et al*. won the CEC 2006 Othello competition [4]. They initialized the population of evolutionary algorithm with known well-playing individual learnt from temporal difference learning. Their evolutionary algorithm used only simple mutation and the evolved strategy is slightly different from the original seed. But the competition results show that the evolved strategies have better generalization capability than other players including the original seed player.

The CEC competition has 904 entries (submissions) from more than 10 persons. Each person can submit more than one strategy. The strategy learnt from TDL is acquired from the competition website (organizer opened it) and it is ranked in the top 10. It is downloadable from http://algoval.essex.ac.uk:8080/othello/html/SampleMLP.txt. It is represented with multi-layer perceptron (MLP) with 64 input neurons, 32 hidden neurons and 1 output neuron. The depth of the game tree is set to 1. It is because the purpose of the competition is to find a way to evolve strategies rather than optimizing the game tree search. The parameters of hybrid algorithm are as follows. The population size is 50, the maximum number of generation is 100, and mutation rate is 0.01. There are two mutations: $w`=w+0.01$ and $w`=w-0.01$. The fitness of the individual is calculated from the following equation.

Fitness = (Number of wins) × 1.0 + (Number of draws) × 0.5

Each individual plays 1000 games against standard heuristics represented weights matrix [6].

Figure 4 shows the change of fitness and although it starts from the near 630, it converges to the 640. The analysis showed that only 80 parameters are different from the initial networks learnt from TDL among total 2113 parameters.

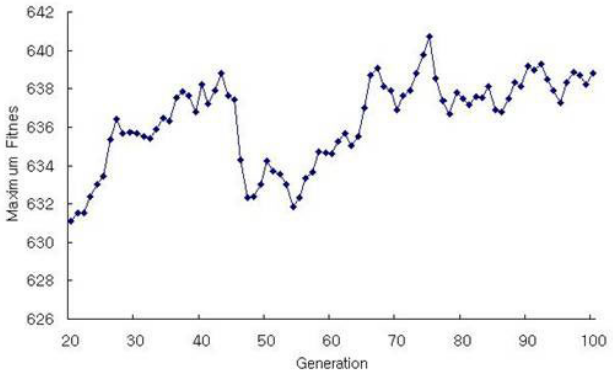

Figure 4. The fitness change of hybrid evolution (each point is averaged the previous 20 generations to smooth the graph).

At the preliminary league, the best solution of the hybrid algorithm (kjkim-mlp-3) is ranked as 3rd among 904 submissions. It is not the best player in the trial league. The final competition league shows that the proposed method performs better than other strategies. The result of the competition is from [4] and summarized in table I, II, and III. There are 12 finalists from 12 persons. In the table, mlp-again2 (2nd rank in the preliminary league, it is the same one with the player used for the initialization of population) shows low rank compared to the proposed one. Although the preliminary league showed that the proposed method gets low rank compared to the mlp-again2, the proposed method outperforms the mlp-again2 in the final round.

TABLE I
SUMMARIZATION OF COMPETITION RESULTS (RANDOMNESS IN MOVE SELECTION = 0%) EACH PLAYER HAS 2 GAMES WITH OTHER PLAYERS.

rank	Win	Draw	Loss	Name
1	20	0	2	kjkim-mlp-3
2	17	1	4	Alez V
3	17	0	5	NButtBradford1b
4	14	2	6	mlp-again2
5	13	2	7	delete-me-cel-1-10
6	13	1	8	brookdale4
7	8	2	12	tomy0
8	7	1	14	fedevadeculo
9	5	0	17	last weeb1
10	5	1	16	jesz3
11	5	2	15	Jorge
12	1	2	19	tpr-tdl-01-500000

In the web page's report, the proposed algorithm (kjkim-mlp-3) outperforms the player learnt from TDL which is used for initialization of the population of the hybrid

algorithm. For 1000 games (1% randomness), kjkim-mlp-3 gets 723 wins and 10 draws.

TABLE II
SUMMARIZATION OF COMPETITION RESULTS (RANDOMNESS IN MOVE SELECTION = 1%) EACH PLAYER HAS 20 GAMES WITH OTHER PLAYERS.

rank	Win	Draw	Loss	Name
1	181	4	35	kjkim-mlp-3
2	170	7	43	Alez V
3	161	12	47	mlp-again2
4	157	5	58	NButtBradford1b
5	138	10	72	brookdale4
6	137	17	66	delete-me-cel-1-10
7	73	7	140	fedevadeculo
8	70	14	136	tomy0
9	58	17	145	Jorge
10	55	7	158	jesz3
11	46	2	172	last weeb1
12	17	12	191	tpr-tdl-01-500000

TABLE III
SUMMARIZATION OF COMPETITION RESULTS (RANDOMNESS IN MOVE SELECTION = 10%) EACH PLAYER HAS 20 GAMES WITH OTHER PLAYERS.

rank	Win	Draw	Loss	Name
1	163	1	56	kjkim-mlp-3
2	161	4	55	mlp-again2
3	158	3	59	Alez V
4	153	9	58	brookdale4
5	150	6	64	delete-me-cel-1-10
6	147	5	68	NButtBradford1b
7	73	7	140	Fedevadeculo
8	71	5	144	Jorge
9	68	4	148	jesz3
10	67	3	150	tomy0
11	58	6	156	last weeb1
12	21	7	192	tpr-tdl-01-500000

B. Knowledge-Incorporated Evolution

In the Othello community, the widely used Othello programs are WZEBRA and NTEST. They are one of the strongest programs in the world. The source code of the ZEBRA is available on the internet under GPL (http://radagast.se/othello/zebra.tar.gz). WZEBRA is a windows version of ZEBRA. It contains opening books with more than 500,000 positions. In this paper, we used the opening book in the evolution stage and endgame solver is used when the number of empty squares is below 4. To increase diversity of openings chosen from the opening DB, the opening is randomly selected among the best 3 moves. It will help the evolutionary player can deal with many variations of good openings.

The neural network used for evaluating the configuration of board is the same with the Fogel's method used for checkers [2]. Previously, Chung et al. used the architecture for Othello [12]. The depth of game tree used for evolution and competition among the final evolved strategies is 2. The population size is 20. Spatial preprocessing layer is used as the same with [12]. The number of games played for the fitness evaluation is 5. There are four different versions of evolution.

EV, EV_O, EV_E, and EV_O_E. EV represents evolution, O represents opening and E represents endgame. EV_O_E represents evolution with opening and endgame knowledge. EV means pure evolution without domain knowledge. EV_O and EV_E mean the evolution with only one domain knowledge (either opening or endgame).

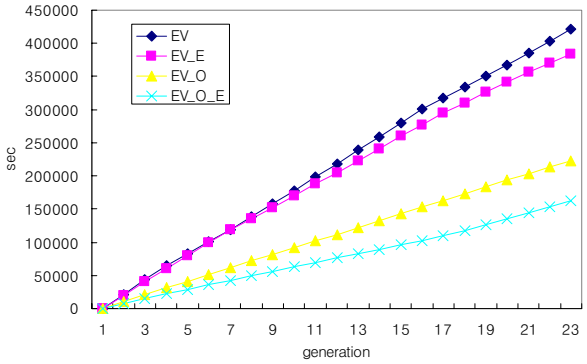

Figure 5. The evolution time of the four different versions (EV > EV_E > EV_O > EV_O_E)

Figure 5 shows the evolution time of the four different versions. Because opening knowledge save the time for game tree searching, it reduces much time for evolution. Also, endgame solver reduces the evolution time. It means that domain knowledge can significantly reduce required time for the evolution. Table IV compares the number of generations evolved for the same time span.

TABLE IV
THE NUMBER OF GENERATIONS EVOLVED FOR 3 DAYS

	EV	EV_E	EV_O	EV_O_E
# of generations	300	327	597	770

We have compared individuals evolved using the same time (computational resource). EV (300 generations), EV_E (327 generations), EV_O (597 generations), and EV_O_E (770 generations) are compared. The results are summarized in table V. It shows that if they used the same computational resource, the one with opening and endgame performs the best. However, the effect of endgame is relatively high. The number of games is 800 (20 individuals × 20 individuals × 2 games (change of colors)).

We have compared individuals evolved with the same number of generations (450 generations). Table VI summarizes the results. In this case, the EV_O_E outperforms other strategies clearly. Meanwhile, the EV_O performs worse than EV. This results show that the

incorporation of knowledge can save the time and discover better strategies combined with knowledge.

TABLE V
THE COMPARISON OF FOUR VERSIONS WITH THE SAME COMPUTATIONAL CONSUMPTION

	Win	Draw	Loss
EV_O_E vs. EV	498	24	278
EV_O_E vs. EV_O	509	26	265
EV_O_E vs. EV_E	392	19	389
EV_O vs. EV	386	29	385
EV_E vs. EV	545	13	242
EV_E vs. EV_O	496	28	276

TABLE VI
THE COMPARISON OF FOUR VERSIONS WITH THE SAME NUMBER OF GENERATIONS

	Win	Draw	Loss
EV_O_E vs. EV	494	21	285
EV_O_E vs. EV_O	532	17	251
EV_O_E vs. EV_E	399	22	379
EV_O vs. EV	355	28	417
EV_E vs. EV	463	18	319
EV_E vs. EV_O	517	24	259

Figure 6 shows the analysis the game between EV_O_E and EV (with the same number of generations). EV_O_E leads the game in the early stage of the game using opening knowledge. After 12 moves of the white, the game is out-of-opening. From the point, the EV_O_E used evolutionary neural networks to evaluate the board configuration with game tree (depth = 2). Because of the big mistake of EV_O_E at 15th move, the game is led by EV. However, after 22nd move of the EV, the game is again leaded by the EV_O_E and it controls the game to the end of the game.

The reason that the EV_O performs worse than EV_E is the early out-of-opening. Although EV_E has low performance at the early stage of the game, it can reverse the results at the end stage of the game. It is better to go out-of-opening as earlier as possible because it minimizes the effect of opening knowledge. Figure 7 shows the situation that describes such phenomenon. At the early stage of the game, the game is leaded by EV_O. Until 10th move, the game is not out-of-opening and EV_O has gained advantage. After the out-of-opening, the EV_E has gained control of the game slightly but it is returned to the EV_O after 30th moves. However, the EV_E performs better at the end stage of the game. Because the endgame solver has invoked when the number of empty squares is 4, the slight win at the end stage of the game means the win of the EV_E.

Figure 8 shows the game between EV_O_E and EV_O. Because both players use the opening knowledge, the game is continued with tie score until 26th moves. After out-of-the opening, there is some fluctuation but the EV_O_E controls the game and finally the endgame solver leads the win of the EV_O_E.

Figure 9 shows the results between EV_E and EV_O_E. It shows that the EV_O_E has gained lead of the game at the early stage of the game. Although it loses the control immediately, the control of the game is returned to EV_O_E after 18th moves. Although the EV_E performs well in the middle of the game, it is not enough to reverse the results of the game. Because both players have the endgame solver, the good job of the EV_E at the middle stage of the game cannot regain the lead of the game.

The analysis showed that the evolutionary neural networks has adapted to the domain knowledge and its synergy provide time save and performance improvement.

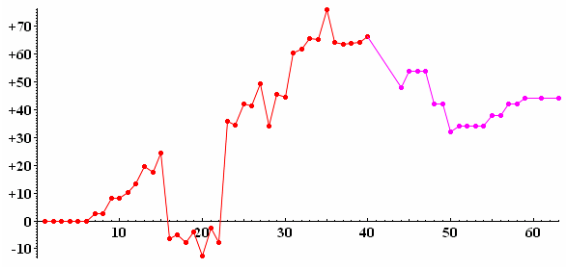

Figure 6. The analysis of the game between EV_E_O (black) and EV (white). + means black leads the game.

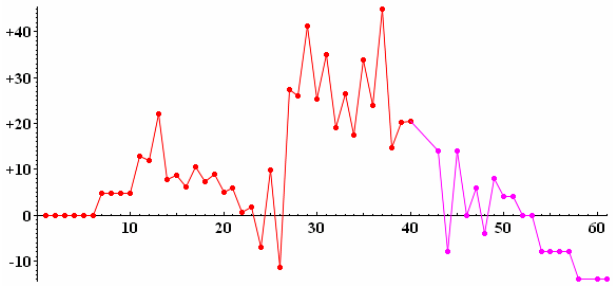

Figure 7. The analysis of the game between EV_O (black) and EV_E (white).

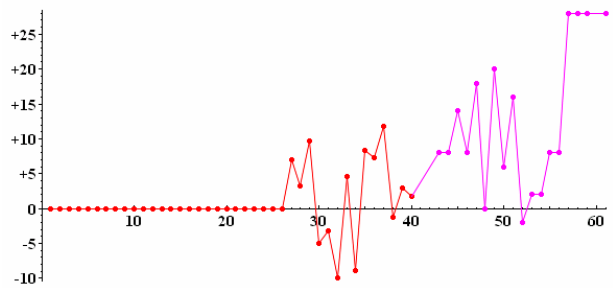

Figure 8. Analysis of the game between EV_O_E (black) and EV_O (white).

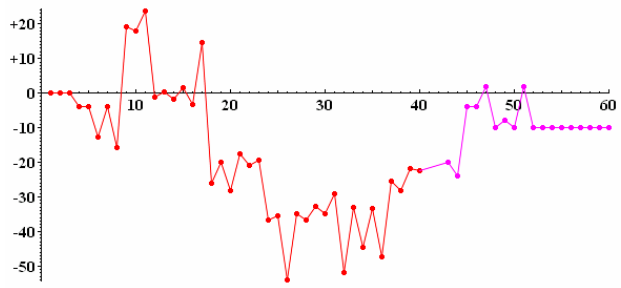

Figure 9. The analysis of the game between EV_E (black) and EV_O_E (white).

V. CONCLUSIONS

This work attempted to incorporate the results of reinforcement learning (TDL and self-playing) to the evolutionary neural networks. Strategy learned from TDL is used to initialize the evolutionary search and the evolved strategy performs better than the initial TDL strategy clearly. In our work, the effect of domain knowledge incorporation in the evolutionary Othello players is systematically evaluated. It shows that the effect of endgame is large than the one of opening DB. The use of the both knowledge performs better than one with single knowledge. Because the effect of knowledge is different, it is useful to control the level of performance and knowledge insertion effort effectively. As a future work, we have to expand the depth of game tree and adopt a deeper endgame solver.

ACKNOWLEDGEMENTS

This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment), IITA-2006-(C1090-0603-0046)

REFERENCES

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

[2] D. B. Fogel, *Blondie24: Playing at the Edge of AI*, Morgan Kaufmann, 2002.

[3] J. A. Singer, "Co-evolving a neural-net evaluation function for Othello by combining genetic algorithms and reinforcement learning," *Lecture Notes in Computer Science*, vol. 2074, pp. 377-389, 2001.

[4] CEC 2006 Othello Competition Results, http://algoval.essex.ac.uk:8080/othello/html/CEC2006Results.html.

[5] 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.

[6] K.-J. Kim, and S.-B. Cho, "Evolutionary Othello players boosted by opening knowledge," *IEEE Congress on Evolutionary Computation*, pp. 984-991, 2006.

[7] 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.

[8] M. Buro, "Improving heuristic mini-max search by supervised learning," *Artificial Intelligence*, vol. 134, no. 1-2, pp. 85-99, 2002.

[9] NTEST, http://othellogateway.com/ntest/Ntest/index.htm.

[10] WZEBRA, http://www.radagast.se/othello/.

[11] D. E. Moriarty and R. Miikkulainen, "Discovering complex Othello strategies through evolutionary neural networks," *Connection Science*, vol. 7, pp. 195-209, 1995.

[12] 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.

[13] T. P. Runarsson, and S. M. Lucas, "Co-evolution versus self-play temporal difference learning for acquiring position evaluation in small-board Go," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 628-640, 2005.

[14] 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.

[15] R. Sutton and A. G. Barto, *Reinforcement Learning*, MIT Press, 1998.

[16] M. Buro, "Toward opening book learning," *ICCA Journal*, vol. 22, no. 2, pp. 98-102, 1999.

[17] T. R. Lincke, "Strategies for the automatic construction of opening books," *Lecture Notes in Computer Science*, vol. 2063, pp. 74-86, 2001.

[18] R. M. Hyatt, "Book learning – A methodology to tune an opening book automatically," *ICCA Journal*, vol. 22, no. 1, pp. 3-12, 1999.