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

Effect of look-ahead search depth in learning position evaluation functions for Othello using ϵ-greedy exploration

Thomas Philip Runarsson
Science Institute
University of Iceland, Iceland
tpr@hi.is

Egill O. Jonsson
Science Institute
University of Iceland, Iceland
egilljon@hi.is

***Abstract*— This paper studies the effect of varying the depth of look-ahead for heuristic search in temporal difference (TD) learning and game playing. The acquisition position evaluation functions for the game of Othello is studied. The paper provides important insights into the strengths and weaknesses of using different search depths during learning when ϵ-greedy exploration is applied. The main findings are that contrary to popular belief, for Othello, better playing strategies are found when TD learning is applied with lower look-ahead search depths.**

Keywords: Temporal difference learning, heuristic search, look-ahead depth, ϵ-greedy exploration, Othello.

I. Introduction

A common means of realizing game playing strategies for board games is by heuristic search, where a heuristic *value function* is used to evaluate board positions. In order to select the best move a player will try all possible moves, resulting in a number of so called afterstates [7]. Each afterstate is evaluated using the heuristic value function and the move corresponding to the highest afterstate value chosen. This is also known as 1-ply heuristic search.

Samuel's checker players [6] were one of the first to use heuristic search and apply a learning method that would improve the heuristic value function over time. Consider a typical game tree as the one depicted in Fig. 1. The open and black circles represent the afterstates for two opponents playing a game. Each afterstate, s, has the value, $V(s)$. The afterstate values can be improved during game play as follows,

$$V(s) \leftarrow V(s) + \alpha\big(V(s_B) - V(s)\big)$$

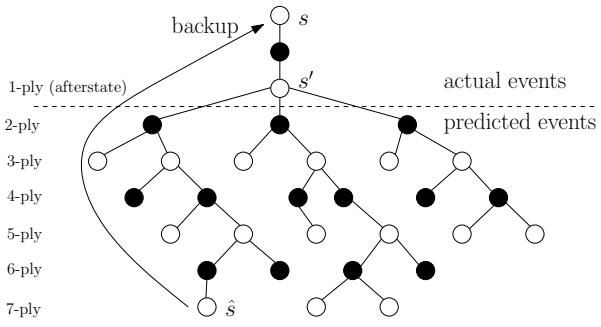

Fig. 1. *Game search tree and backup diagram for TD learning. Black circles represent afterstate positions for opponent to move.*

where s_B is the backed up state, α a step size parameter, and $\big(V(s_B) - V(s)\big)$ is the so called *temporal-difference* (TD). A backup based on a single step, i.e. $V(s')$ corresponds to actual events. However, if one could foresee the moves of the opponent, for example by knowing its value function, future events may be predicted. For example, a 7-ply backup may be applied, as depicted in Fig. 1, using the value of afterstate $\hat{s}$.

When learning through self-play both players use and update the same value function. One player chooses moves so as to maximize the value of the resulting game positions while the other attempts to find positions of minimum value. Since the players use the same value function both have a model of their opponent and are capable of predicting the moves ahead. The benefits of searching deeper than one step is to obtain better move selections. If one has a perfect model of the opponent and an imperfect value function, then it is believed that deeper search will usually produce better moves. This was found to be the case for Tesauro's TD-gammon [8]. Furthermore, if the search results in an end game state, then the effect of the imperfect value function is removed, and the move determined must be optimal.

The computational cost of a deep look-ahead search can be high and is therefore often limited to a number of steps. Although clearly look-ahead is useful during play, as noted above, it remains unknown to what extent look-ahead is helpful during learning. For example, does a value-function learned at 2-ply play a better game at 4-ply than a value function learned at 4-ply playing at the same ply? Clearly there is a tradeoff as to the number of games that can be played and the depth of look-ahead applied during learning within a pre-specified computing time. Look-ahead does, however, propose a way deciding a distribution of backups that may result in faster and a better approximation of the optimal or greedy value function. Optimal in self-play implies playing optimally against a copy of itself. However, a self-play player may also learn to play better against many opponents.

Based on the arguments above one may assume that playing at a deeper ply during learning should result in better value functions. However, the experimental result presented here show the opposite to be the case for Othello when the value function is approximated using a weighted piece counter and ϵ-greedy exploration is employed.

The paper is organized as follows. In section II a brief

description of the game Othello is given. In section III the implementation of TD learning is described in full detail. This is followed by an extensive set of experimental results and evaluation of value functions learned in section IV. The paper is then concluded with a discussion and summary of main findings.

II. Othello

The game of Othello is played on an 8×8 board, with a starting configuration as shown in Fig. 2 with the middle 4 squares occupied. Black plays first, and the game continues until the board is full (after 60 turns), or until neither player is able to move. Note that a player $must$ move if able to, passing only happens when a player has no legal moves available.

A legal move is one which causes one or more opponent counters to be flipped. Counters are flipped when they lie on a continuous line (horizontal, vertical, or diagonal) between the newly placed counter, and another counter of the placing player. Counters placed in one of the four corners can never satisfy this condition, and can therefore never be flipped. Hence, the corners play a pivotal role in the game, and valuing them highly tends to be the first thing learned. Indeed the weighted piece counter (WPC) [9] used as a benchmark in that study also reflects this. There the highest value of 1 is given to all four corners. To hinder the possibility of an opponent getting a corner, the squares next to them should be avoided. For this reason they are given the lowest value -0.25. As a consequence the WPC encourages the players to place its counter at advantageous squares. The total set of weights for this heuristic player is given in Fig. 3. Note that the weights of this heuristic player are symmetric under reflection and rotation, and have just 10 distinct values out of a possible 64. It would be possible to simplify the learning task by enforcing this kind of symmetry, and adjusting just 10 parameters instead of 64. This would mean building in more expert knowledge however, and could also place undesirable constraints on the value function.

The first strong learning Othello program developed was Bill [3], [4]. Later, the first program to beat a human champion was Logistello [2], the best Othello program from 1993–1997. Logistello also uses a linear weighted evaluation function but with more complex features than just the plain board. Nevertheless, the weights are tuned automatically using self-play. Logistello also uses an opening book based on over $23,000$ tournament games and fast game tree search [1].

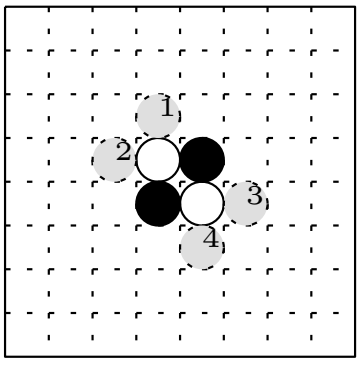

Fig. 2. *The initial Othello board, showing the four possible first moves, which are all equivalent under reflection and rotation (black moves first).*

```
 1.00 -0.25 0.10 0.05 0.05 0.10 -0.25  1.00
-0.25 -0.25 0.01 0.01 0.01 0.01 -0.25 -0.25
 0.10  0.01 0.05 0.02 0.02 0.05  0.01  0.10
 0.05  0.01 0.02 0.01 0.01 0.02  0.01  0.05
 0.05  0.01 0.02 0.01 0.01 0.02  0.01  0.05
 0.10  0.01 0.05 0.02 0.02 0.05  0.01  0.10
-0.25 -0.25 0.01 0.01 0.01 0.01 -0.25 -0.25
 1.00 -0.25 0.10 0.05 0.05 0.10 -0.25  1.00
```

Fig. 3. The weights ($\boldsymbol{w}$) for the heuristic player [9].

III. Implementation

The heuristic value function, a WPC, to be learned is described by:

$$f(\boldsymbol{x}) = \sum_{i=1}^{8\times 8} w_i x_i + w_0 \tag{1}$$

where x_i is the value at square i on the board, which is 0 when Empty, 1 if Black, and -1 for White. The single scalar output of function $f(\boldsymbol{x})$ is interpreted as follows. The value indicates which position is most favorable for a particular player, with larger values favoring Black, and smaller values White.

In TD learning the weights of the evaluation function are updated during game play using a gradient-descent method. Let $\boldsymbol{x}$ be the board observed by a player about to move, and similarly $\boldsymbol{x}'$ the board after a number of predicted moves. Then the evaluation function may be updated during play as follows [7, p.199]:

$$\begin{aligned} w_i &\leftarrow w_i + \alpha\big[v(\boldsymbol{x}') - v(\boldsymbol{x})\big]\frac{\partial v(\boldsymbol{x})}{\partial w_i} \\ &= w_i + \alpha\big[v(\boldsymbol{x}') - v(\boldsymbol{x})\big]\big(1 - v(\boldsymbol{x})^2\big)x_i \end{aligned} \tag{2}$$

where

$$v(\boldsymbol{x}) = tanh(f(\boldsymbol{x})) = \frac{2}{1+\exp(-2f(\boldsymbol{x}))} - 1 \tag{3}$$

is used to force the value function v to be in the range -1 to 1. This method is known as gradient-descent TD(0) [7]. If $\boldsymbol{x}'$ is a terminal state then the game has ended and the following update is used:

$$w_i \leftarrow w_i + \alpha\big[r - v(\boldsymbol{x})\big]\big(1 - v(\boldsymbol{x})^2\big)x_i$$

where r corresponds to the final utilities: $+1$ if the winner is Black, -1 when White, and 0 for a draw. Similarly, when a predicted state $\boldsymbol{x}'$ is terminal their values will correspond to these final utilities.

The update rule is perhaps the simplest version of temporal difference learning and works quite well on this task. If the step size parameter α, in (2), is reduced properly over time

```
1  if u() < ε do
2     make purely random legal move
3  else
4     make best legal move based on the state evaluation function
5  od
```

Fig. 4. *The ϵ-greedy technique for forcing random moves, where $u()$ returns a random number drawn from a uniform distribution $\in [0\ 1]$.*

this method will also converge [7, p. 13]. During game play, with probability $\epsilon = 0.1$, a random or exploratory move is forced. This is known as an ϵ-greedy policy and is described in Fig 4.

Note that, TD(0) is attempting to learn the probability of winning from a given state (when following the ϵ-greedy policy). However, at higher ply than 1-ply the moves are predicted on the assumption that a greedy policy is followed, i.e. $\epsilon = 0$. This assumption, and the fact that a WPC is used to approximate the value function, may lead to a curious result.

IV. Experimental Results

The C-program developed and described in [5] was used to perform the experiments described in this section. Each of the experiments is repeated 10 times. Initially $\alpha = 0.01$ which is then reduced by a factor of 0.95 every 500 games with a total of 50.000 games played per experiment. The exploration rate was kept constant at the rate of $\epsilon = 0.1$ and the initial weights, $\boldsymbol{w}$, are set to zero in each case. These settings were found to give the best results in a previous study [5], when learning at 1-ply. A second set of experiments are then conducted where ϵ is reduced every 500 games by a factor of 0.9. The aim here is to reduce the effects caused by randomness inherent in the ϵ-greedy approach, which may disrupt learning as the number of ply depth increases.

A. Evaluation

Value functions approximated by the WPC described in the previous section are learned through self-play. Each experiment is performed ten times, using 1 to 5–ply look-ahead search, resulting in a total of 50 independently derived approximate value functions. Each player trained at a given ply was then matched with all of the players trained at a different ply. Leagues are held by forcing players to play at pre-specified ply. In other words, each of the ten players trained at a given ply will play each of the ten players trained at a another ply, once in each color, resulting in a total of 200 games. Note that forced random moves are *not* used in these leagues, i.e. $\epsilon = 0$. In the case of a draw both players receive half a win.

When the players play against each other, at the very ply they were trained at, the result is as shown in table I. Here, for example, one can see that the ten players trained at 5–ply, and playing at 5–ply, will win 187.5 games out of 200 against the ten players playing and trained at 1–ply. The number of games won then decreases as the number of ply of the opponents increase. This result is as one may have expected. The total number of wins are summed up in the last column. In all cases players trained and playing at higher ply exhibit the best performance. Is this due to the quality of the value function learned, greater look-ahead used during play or perhaps both? In order to answer this question two kinds of player evaluation are performed.

TABLE I

Number of wins out of 200 games when players play at the same ply as they were trained at.

v's	**1–ply**	**2–ply**	**3–ply**	**4–ply**	**5–ply**	$\sum$wins
1–ply	–	33.5	25	9.5	12.5	80.5
2–ply	166.5	–	81	53	26.5	327
3–ply	175	119	–	92.5	66	452.5
4–ply	190.5	147	107.5	–	78.5	523.5
5–ply	187.5	173.5	134	121.5	–	616.5

Firstly, players trained at a given ply are allowed to compete amongst themselves at different ply. These results are given in the following section IV-B. The results illustrate the effect of look-ahead on the quality of play. Secondly, in order to evaluate the quality of the value function learned, when training at different ply, the players compete with a fixed look-ahead depth. If, for example, we expect the quality of the value functions learned using greater lookahead to be the best, then the corresponding players should win most games, or at least do so at the ply they trained at. These results are presented in section IV-C.

In order to investigate the impact ϵ has on the quality of the learned value function additional experiments are needed. Section IV-D presents results for when the above experiments are repeated, however, with the ϵ exploratory noise reduced slowly during learning. In section IV-E, the quality of the values function found for a fixed $\epsilon = 0.1$ are compared with the ones found using a slowly reduced ϵ.

Finally, as an additional test of the quality of the value functions found, the players compete with the heuristic value function represented in Fig. 3. In this case only 20 games are played as this represents only a single player. This result is presented in section IV-F.

B. Players trained at n-ply competing amongst themselves at different ply.

Here the ten players trained at a given ply compete among each other but using different look-ahead search depths during play. In all the five different cases it is clear, from

TABLE II

The players playing at their trained ply against themselves at different ply. The rows are the players training ply, and the columns the ply they played against themselves.

	1	**2**	**3**	**4**	**5**
1-ply	–	45	32	16.5	4.5
2-ply	173	–	53	60.5	27
3-ply	189	138.5	–	68.5	27
4-ply	195	154	149.5	–	76.5
5-ply	197	176.5	177	139	–

table II, that playing with a deeper look-ahead search will result in the most number of wins. For example, the players trained at 5-ply and playing at 5-ply against themselves at only 1-ply will result in 197 games won out of 200. This number gradually reduces to 139 as the ply played increases from 1 to 4, see bottom row in table II. The empty bottom right cell correspond to 5-ply versus 5-ply which necessarily results in 100 games won for both sides.

The results in table II confirm that playing at higher ply will in all cases result in a better game playing strategy, regardless of the depth of search applied during the learning of the value function.

C. Players competing at the same ply

In this section value functions found by applying different search depths during learning are compared directly. The ten players, using the different value functions learned, compete with one another using a fixed depth of search from 1 to 5–ply.

When all players are forced to play at 1-ply the players trained at a higher ply perform significantly worse. This result may be seen in table III. The table shows how the 1–ply players win 95 games against the players trained at 2–ply, 103 against the 3–ply players, 131.5 against the 4–ply, and finally 148.5 against the 5–ply players. Clearly, players trained at a higher ply should not play at a lower ply. In this league the 2–ply and 5–ply players perform worst with a total of 349 and 333.5 wins out of 800 respectively. The best performance is observed for the players actually trained at 1–ply with 478 wins in total.

TABLE III

Number of wins out of 200 games when all players are forced to play at 1–ply.

	1–ply	**2–ply**	**3–ply**	**4–ply**	**5–ply**	$\sum$wins
1–ply	–	95	103	131.5	148.5	478
2–ply	105	–	64.5	96	83.5	349
3–ply	97	135.5	–	105.5	117.5	455
4–ply	68.5	104	94.5	–	117	459
5–ply	51.5	116.5	82.5	83	–	333.5

TABLE IV

Number of wins out of 200 games when all players are forced to play at 2–ply.

	1–ply	**2–ply**	**3–ply**	**4–ply**	**5–ply**	$\sum$wins
1–ply	–	68	91	94.5	98.5	352
2–ply	132	–	131	137.5	112.5	513
3–ply	109	69	–	123.5	136.5	438
4–ply	105.5	62.5	76.5	–	99	343.5
5–ply	101.5	87.5	63.5	101	–	353.5

TABLE V

Number of wins out of 200 games when all players are forced to play at 3–ply.

	1–ply	**2–ply**	**3–ply**	**4–ply**	**5–ply**	$\sum$wins
1–ply	–	65.5	78.5	113	106.5	363.5
2–ply	134.5	–	98	135.5	119	487
3–ply	121.5	102	–	120.5	117.5	461.5
4–ply	87	64.5	79.5	–	102	333
5–ply	93.5	81	82.5	98	–	355

TABLE VI

Number of wins out of 200 games when all players are forced to play at 4–ply.

	1–ply	**2–ply**	**3–ply**	**4–ply**	**5–ply**	$\sum$wins
1–ply	–	53	42	71.5	90	256.5
2–ply	147	–	119.5	119.5	115	501
3–ply	158	80.5	–	96	133.5	468
4–ply	128.5	80.5	104	–	100	413
5–ply	110	85	66.5	100	–	361

TABLE VII

Number of wins out of 200 games when all players are forced to play at 5–ply.

	1–ply	**2–ply**	**3–ply**	**4–ply**	**5–ply**	$\sum$wins
One	–	57.5	58.5	93	84.5	293
Two	142.5	–	116.5	121.5	121.5	502
Three	141.5	83.5	–	104	129.5	458
Four	107	78.5	96	–	111	392.5
Five	116	78.5	70.5	89	–	354

The experiment is repeated, this time playing at a fixed ply of 2, 3, 4, and 5. These results are shown in tables IV, V, VI and VII respectively. This is where a curious observation is made. With the exception of the players trained at 1-ply, the players trained at lower ply than they are currently playing at consistently perform better than those trained at a higher ply. This result is perhaps clearest in table VII. In general, the overall best result is clearly for the players trained at only 2–ply, followed by 3–ply, 4–ply and 5–ply, in that order. The worst performance is exhibited at 1–ply, unless the intention is to actually play at 1–ply.

D. Players using reduced ϵ trained at n-ply competing amongst themselves at different ply.

The result in the previous section may be due to the increased uncertainty of the backed up state due to the ϵ-greedy policy followed by the players. To minimize this effect it was decided to reduce ϵ every 500 games by a factor of 0.9.

The experiments conducted in the previous section IV-C were repeated, using this modification, and these results are presented in tables VIII to XII. When the players are forced to compete at 1–ply the players trained at 3–ply are the best followed by 1–ply (as before). In general there seems to be a greater variation in the number of games won, compared to the results in section IV-C. However, when playing at higher ply, once again the 2–ply players are the overall winners. When all play at 5–ply (see fig XII) the ordering remains basically the same with the exception that now the 4–ply players perform better than the 3–ply players (note that the 3–ply players played overall best at 1–ply).

It would be interesting to know whether the players learned in this section have greater or lesser performance to those of section IV-C. This is investigated in the following section.

TABLE VIII

Number of wins out of 200 games when all players (with reduced ϵ) are forced to play at 1–ply.

	1–ply	**2–ply**	**3–ply**	**4–ply**	**5–ply**	$\sum$wins
1-ply	0	122	106	147.5	109.5	485
2-ply	78	0	56.5	152.5	96	383
3-ply	94	143.5	0	151	122.5	511
4-ply	52.5	47.5	49	0	69	218
5-ply	90.5	104	77.5	131	0	403

TABLE IX

Number of wins out of 200 games when all players (with reduced ϵ) are forced to play at 2–ply.

	1–ply	**2–ply**	**3–ply**	**4–ply**	**5–ply**	$\sum$wins
1-ply	0	62	65	104.5	85.5	317
2-ply	138	0	98.5	154.5	122.5	513.5
3-ply	135	101.5	0	125	112.5	474
4-ply	95.5	45.5	75	0	90	306
5-ply	114.5	77.5	87.5	110	0	389.5

TABLE X

Number of wins out of 200 games when all players (with reduced ϵ) are forced to play at 3–ply.

	1–ply	**2–ply**	**3–ply**	**4–ply**	**5–ply**	$\sum$wins
1-ply	0	36.5	75.5	93	87	292
2-ply	163.5	0	114.5	143.5	123	544.5
3-ply	124.5	85.5	0	115.5	121	446.5
4-ply	107	56.5	84.5	0	100	348
5-ply	113	77	79	100	0	369

TABLE XI

Number of wins out of 200 games when all players (with reduced ϵ) are forced to play at 4–ply.

	1–ply	**2–ply**	**3–ply**	**4–ply**	**5–ply**	$\sum$wins
1-ply	0	20.5	30	43.5	59.5	153.5
2-ply	179.5	0	110	138	128	555.5
3-ply	170	90	0	97	111	468
4-ply	156.5	62	103	0	126.5	448
5-ply	140.5	72	89	73.5	0	375

TABLE XII

Number of wins out of 200 games when all players (with reduced ϵ) are forced to play at 5–ply.

	1–ply	**2–ply**	**3–ply**	**4–ply**	**5–ply**	$\sum$wins
1-ply	0	32.5	57	56.5	59.5	205.5
2-ply	167.5	0	141	106	135.5	550
3-ply	143	59	0	100.5	125	427.5
4-ply	143.5	94	99.5	0	121	458
5-ply	140.5	64.5	75	79	0	359

E. Players using reduced ϵ competing with fixed ϵ players at different ply.

When the ten players trained with ϵ reduced slowly during learning are matched against the players using a fixed ϵ the results are as shown in table XV. Here one notices that reducing ϵ resulted in overall better players when applying 2,3 and 5–ply search, but worse players at 1 and 4–ply search.

TABLE XIII

Players trained at various ply (reduced ϵ) versus the players at fixed ϵ at different ply.

	1-ply	**2-ply**	**3-ply**	**4-ply**	**5-ply**	$\sum$wins
1-ply	70	62	72.5	66.5	89.5	360.5
2-ply	83	123.5	103.5	121.5	142	573.5
3-ply	100	110	90	105	115	520
4-ply	39	88	80	124	117	448
5-ply	82.5	115.5	79	126.5	119.5	523

F. Players trained at different ply versus heuristic player

As a final evaluation of the value functions learned, the different players are compared with a player using the heuristic value function represented in Fig. 3. Both opponents will play at the same but varying search depths. This result is depicted in table XIV for players trained with a fixed $\epsilon = 0.1$. Only 20 games are played in this instance as the heuristic player represents only a single player and not ten different players as in the previous evaluations. By this means of evaluation it is found that the players trained at 3–ply perform overall the best, followed by 2–ply and 1–ply. The higher ply players again perform the worst. Table XV shows the same results for the players using a slowly reduced ϵ. In this case all players perform significantly worse when playing at 1–ply, but significantly better at higher ply, with the exception of players trained at 1–ply. Here, overall, the 4–ply players are best.

TABLE XIV

Players trained at various ply versus the heuristic player at different ply.

	1-ply	**2-ply**	**3-ply**	**4-ply**	**5-ply**	$\sum$wins
1-ply	10	13.5	15	16	16	70.5
2-ply	9	14.5	14	19	13.5	70
3-ply	12	15	15	19	16	77
4-ply	10	6	13	12	18	59
5-ply	7	9	14.5	10.5	10	51

TABLE XV

Players trained at various ply (reduced ϵ) versus the heuristic player at different ply.

	1-ply	**2-ply**	**3-ply**	**4-ply**	**5-ply**	$\sum$wins
1-ply	5	9	13	16.5	6	49.5
2-ply	4	15	15.5	14	19	67.5
3-ply	4.5	16	9	18	18	65.5
4-ply	6	17	13	17.5	18	71.5
5-ply	0	13	16.5	13	18	60.5

V. SUMMARY AND CONCLUSION

A number of WPC, using different depths of ply during learning, were found using TD learning. Our expectations were that better value functions would be learned when training with deeper look-ahead search. However, this was not found to be the case. This is an important result since the cost of look-ahead is usually high. If deeper search is not as useful, as one may have expected, then a significant amount of computing time may be saved.

The main results are that, during game playing, better decisions are made when deeper look-ahead is used. However, when learning, a depth of 2–ply look-ahead search results in the best value functions for Othello. That is, train at 2–ply and play at the highest ply possible.

A plausible explanation for this may be that the predicted backup values used for TD learning are not accurate, since a ϵ-greedy policy is followed (see Fig. 4) rather than a purely greedy one. However, reducing ϵ during learning did not help confirm this suspicion. It is also possible that the WPC is far too simple and unable to capture the value function to be learned. This may also have a disruptive effect on the gradient based TD algorithm.

The main conclusion of these experimental findings is that one should not assume that deeper ply search will necessarily result in better policies learned. Further studies are necessary to explain this result.

REFERENCES

[1] M. Buro, "ProbCut: An effective selective extension of the Aalpha-Beta algorithm," *ICGA Journal*, vol. 18, pp. 71 – 76, 1995.

[2] ——, "LOGISTELLO – a strong learning othello program," 1997, http://www.cs.ualberta.ca/ mburo/ps/log-overview.ps.gz.

[3] K.-F. Lee and S. Mahajan, "A pattern classification approach to evaluation function learning," *Artificial Intelligence*, vol. 36, pp. 1 – 25, 1988.

[4] ——, "The development of a world class othello program," *Artificial Intelligence*, vol. 43, pp. 21 – 36, 1990.

[5] S. Lucas and T. P. Runarsson, "Temporal difference learning versus co-evolution for acquiring othello position evaluation," in *IEEE Computational Intelligence and Games*, 2006, pp. 52–58.

[6] A. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, pp. 211 – 229, 1959.

[7] R. Sutton and A. Barto, *Introduction to Reinforcement Learning*. MIT Press, 1998.

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

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