Experimental Analysis of the Probability Method

KOBAYASHI Yasuyuki, NANBA Shinya, HAGIO Retush, OMURA Masahiko

Journal: Interdisciplinary Information Sciences
Volume: 13
Number: 2
Page Range: 189-201
Year: 2007-09-18
URL: http://hdl.handle.net/10097/45540
Experimental Analysis of the Probability Method

Yasuyuki KOBAYASHI\(^1\), Shinya NANBA\(^2\), Retushi HAGIO\(^3\) and Masahiko OMURA\(^3\)

\(^1\)Department of Mathematics and Computer Science, Interdisciplinary Faculty of Science and Engineering, Shimane University
\(^2\)Hyougo Prefectural Yamasaki High School
\(^3\)Department of Mathematics and Computer Science, Interdisciplinary Graduate School of Science and Engineering, Shimane University

Received September 4, 2006; final version accepted June 1, 2007

The Backward Induction Method, which is the most basic algorithm used for game tree searches, has two weak points. First, the move selected by this method is assured to be the best move as far as the search depth of the game tree is concerned, but is not necessarily the best move towards the end of the game. Secondly, the values evaluated for the leaf nodes do not necessarily give the best advantage at the end of the game.

In a previous paper, we proposed a new algorithm, the Probability Method, which is useful for games finishing at the constant moves such as Othello.

In this paper we compare the Probability Method with the Backward Induction and Bayesian Methods using Othello.

Moreover we propose a pruning procedure for the Probability Method and compare it with the alpha-beta pruning procedure used in the Backward Induction Method.

We show that the Probability Method is more effective than both of the Backward Induction and Bayesian Methods and that the pruning procedure for the Probability Method is more advantageous than alpha-beta pruning in the Backward Induction Method for some phases.

KEYWORDS: game tree search, winning probability, Probability Method, pruning procedure

1. Introduction

The Backward Induction Method is the most basic game tree search for two person complete information games such as Chess, Othello, Shogi, and Go. However, it has the following shortcomings when the game does not finish at the search depth; (1) the move selected by the Backward Induction Method only promises to be the best move within the search depth, but it is not necessarily the best move towards the end of the game and (2) the values evaluated for the leaf nodes do not necessarily provide an advantage towards the end of the game.

In a previous paper [3], we proposed a new game tree search method, the Probability Method, which is available for games finishing at the constant moves such as Othello.

On the other hand, the Bayesian Method gives an empirical distribution of the expected values at each leaf node [1]. This method has more information at the leaf nodes than the Probability Method because we can derive a winning probability for the Probability Method from an empirical distribution for the Bayesian Method.

In Section 2, we use Othello to compare the Probability Method with the Backward Induction and Bayesian Methods and show that our method is the most effective.

In Section 3, we propose a pruning procedure for the Probability Method and demonstrate its correctness. In addition, we use Othello to compare our pruning procedure with the alpha-beta pruning procedure used in the Backward Induction Method.

Finally, conclusions are given in Section 4.

2. Comparisons with the Backward Induction and Bayesian Methods

2.1 Previous results

We distinguish two players; the max player and the min player. Let a node be termed a max node if the max player is to move, and a min node otherwise, as shown in Fig. 1. The numbers beside the circles/boxes represent the back up values \(G(p)\) derived from the Backward Induction Method from the max player’s point of view. Let \(p_1, \ldots, p_d\) be the son nodes of the node \(p\) with depth \(D’(< D)\); \(G(p)\) by the Backward Induction Method with depth \(D\) is defined as follows.

\[ G(p) = \max_{p_1, \ldots, p_d} \min_{p} G(p_1), \min_{p} G(p_2), \ldots, \min_{p} G(p_d) \]

\[ G(p) = \max_{p_1, \ldots, p_d} \min_{p} G(p_1), \min_{p} G(p_2), \ldots, \min_{p} G(p_d) \]

\[ G(p) = \max_{p_1, \ldots, p_d} \min_{p} G(p_1), \min_{p} G(p_2), \ldots, \min_{p} G(p_d) \]

\[ G(p) = \max_{p_1, \ldots, p_d} \min_{p} G(p_1), \min_{p} G(p_2), \ldots, \min_{p} G(p_d) \]

\[ G(p) = \max_{p_1, \ldots, p_d} \min_{p} G(p_1), \min_{p} G(p_2), \ldots, \min_{p} G(p_d) \]

\[ G(p) = \max_{p_1, \ldots, p_d} \min_{p} G(p_1), \min_{p} G(p_2), \ldots, \min_{p} G(p_d) \]

\[ G(p) = \max_{p_1, \ldots, p_d} \min_{p} G(p_1), \min_{p} G(p_2), \ldots, \min_{p} G(p_d) \]

\[ G(p) = \max_{p_1, \ldots, p_d} \min_{p} G(p_1), \min_{p} G(p_2), \ldots, \min_{p} G(p_d) \]

\[ G(p) = \max_{p_1, \ldots, p_d} \min_{p} G(p_1), \min_{p} G(p_2), \ldots, \min_{p} G(p_d) \]

\[ G(p) = \max_{p_1, \ldots, p_d} \min_{p} G(p_1), \min_{p} G(p_2), \ldots, \min_{p} G(p_d) \]
where the function $g$ signifies the max player’s advantage.

The max player selects the node with the highest value from among all of the son nodes. On the other hand, the min player selects the node with the lowest value. We assume that this game does not finish at depth 3 but that the max player can search only the game tree given in Fig. 1 to select his move. The Backward Induction Method yields a value of 0.8 on node A, which means that this game will proceed to position I with value 0.8 after three turns if the two players select son nodes using the above mentioned strategy.

The Backward Induction Method is the most effective algorithm if the max player can search till the end of the game, because the max player can give the result of the game to each leaf node. But if the max player cannot search till the end of the game such as in Fig. 1, the Backward Induction Method only assures that the max player can carry the game forward to a position that has a back-up value for the root node. This does not necessarily assure an advantage at the end of the game.

The Probability Method is available for positions just before the point at which each player can search till the end of the game. In Fig. 2 we assume that after the max player has carried the game forward to either position B or C, each player can search till the end of the game using the Backward Induction Method, and the values 1 (which means the max player wins), 0 (ties) and $-1$ (loses) can be given to each leaf node. Moreover we assume that the values given to each leaf node represent the probability that the max player will win if the game moves forward from that leaf node.

At position D the max player can select the best move by searching till the end of the game. So if the max player knows that he cannot win the game from position G, then he selects position H. If he cannot win the game from either position, then he loses the game from the position D. In Fig. 2, the probability of the max player winning from position D is $(1 - 0.7) \times (1 - 0.6) = 0.88$. At position B the min player selects position D, so that the probability of the max player winning from position B is the same as from position D, 0.88.

On the other hand if the game goes forward from position C, the max player can win the game if and only if the min player cannot win the game from positions E and F. Therefore the probability of the max player winning from position C is $0.8 \times 0.9 = 0.72$. The probability of the max player winning from position B, 0.88, is greater than from position C, 0.72. This means that the best selection for the max player at position A is position B.

In the following we define the winning probability $F(p)$ at node $p$ for the max player. We assume the following:

1) each leaf node in the game tree $G$ has a value 1 (which means the max player wins), 0 (ties) or $-1$ (loses),

2) each player can search till a depth $D(> 0)$ from the root node of the game tree,
3) \( f \) is a static evaluation function which gives the probability that the max player will win at each node with depth \( D \) in the game tree.
4) from each node with depth 1 we can search till the end of the game tree.

Let \( p_1, \ldots, p_d \) be the son nodes of node \( p \) with depth \( D(< D) \).

(1) \( p \) is not a root node

\[
F(p) = \begin{cases} 
  f(p) & (p \text{ is a node with depth } D) \\
  & \\
  1 - \prod_i (1 - F(p_i)) & (p \text{ is a max node with depth } D(< D)) \\
  & \\
  \prod_i F(p_i) & (p \text{ is a min node with depth } D(< D)) 
\end{cases}
\]

(2) \( p \) is a root node

\[
F(p) = \begin{cases} 
  \max_i F(p_i) & (p \text{ is a max node}) \\
  \min_i F(p_i) & (p \text{ is a min node}) 
\end{cases}
\]

Each player can search till the end of the game from every node except the root node using the Backward Induction Method, and the values 1, 0 and \(-1\) given to each leaf node. Therefore if the max player can find at least one son node with a back up value of 1, then he can win the game.

The probability that the max player can win the game is \( 1 - \prod_i (1 - F(p_i)) \) because \( \prod_i (1 - F(p_i)) \) is the probability that the max player cannot find a winning son node. On the other hand if the min player cannot find a winning son node, then the max player can win the game. That probability is \( \prod_i F(p_i) \). At the root node the max player selects the son node with the highest winning probability.

Baum and Smith [1] define a Bayesian model of the uncertainty of the expected payoff from each possible move.

Figure 3 shows a simple tree with distributions of the expected values if the two players select optimal moves from the root position. We consider this in the same way as the negamax approach. The negamax value \( v \) of a given position \( p \) represents the value of the position to the player whose turn it is to play; the value to the other player is assumed to be \(-v\). Position \( V \) has a probability of 1/2 of being good for the max player, with an expected value of \(-8\), and probability of 1/2 of being good for the min player, with an expected value of 8. We write this distribution at position \( V \) as \((-8, 8/2)\). The distribution at position \( W \) is \((-6, 8/2)\). There are now 4 possible configurations of the leaf values, shown in Fig. 4. Referring to the left side of Fig. 4, in this case, the max player selects position \( V \) from position \( U \) because he can expect 8 from position \( V \), while he can expect 6 from position \( W \). Therefore the expected value of position \( U \) is 8=(\( \max(8, 6) \)) and its probability is 1/8=(\( 1/2 \times 1/4 \)). In each configuration, the probabilistic weight and the expected values of the internal nodes are shown. The distribution at node \( U \) will be \((8, 6, -2)\) and at node \( T \) \((-8, -6, 2)\), and the expected values at \( S \) and \( T \) are 0 and \(-4\), respectively. Therefore \( T \) is better than \( S \) for the max player.

Generally, the probability density function \( \rho^{(0)}(x) \) associated with node \( \rho \) is the sum of the probability masses \( p_i \) located at \( x_i \):
The Bayesian Method calculates the distributions of expected values at internal nodes based on the distributions of expected values at the leaf nodes. Therefore it contains more information than the Backward Induction and Probability Methods. But the calculation time for the Bayesian Method will be longer than both other methods because the number of $x_i$ in Eq. (1) will be large in most real games.

$$\rho^{(n)}(x) = \sum_i \rho_i^{(n)} \delta(x - x_i^{(n)}) \quad (1)$$

$$\delta(x) = \begin{cases} 0 : & x \neq 0 \\ 1 : & x = 0 \end{cases} \quad (2)$$

The Bayesian Method calculates the distributions of expected values at internal nodes based on the distributions of expected values at the leaf nodes. Therefore it contains more information than the Backward Induction and Probability Methods. But the calculation time for the Bayesian Method will be longer than both other methods because the number of $x_i$ in Eq. (1) will be large in most real games.

$$p = \int_{x>0} \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{(x-h)^2}{2\sigma^2} \right) \, dx, \quad (3)$$

because $x > 0$ means that the max player can win the game.

In this paper we give the winning probability based on the following.

First, we prepared 300 records of Othello games, and made game trees with depth $D(D = 2, 3$ and $5)$ at turn number $S(S = 40, 46, 50, 53, 54)$.

### 2.2 Experimental comparison

In this section we carry out experiments with Othello to demonstrate the advantages of the Probability Method over the Backward Induction and Bayesian Methods.

We need to give the winning probabilities at the leaf nodes for the Probability Method. First, we consider how to give these.

Buro [2] estimates a back-up value $\nu$ with a search depth of 8 from a back-up value $\nu'$ with a search depth of 4 by using linear regression analysis of games of Othello. He obtains the result that $\nu$ is distributed as a normal distribution $N(\nu, \sigma^2)$, where $\sigma = 0.542$ and $\nu = 1.036\nu' - 0.009$ at 28 discs (the number of chips on the board) and $\sigma = 0.884$ and $\nu = 0.956\nu' - 0.067$ at 44 discs. Therefore $\nu$ is approximately distributed as a normal distribution $N(\nu', \sigma^2)$.

Referring to these results we consider the following to be the case:

1) For a leaf node with value $h$, the back-up value $y$ found by searching the game tree from the leaf node to the end of the game is approximately distributed as a normal distribution $N(h, \sigma^2)$.

2) If we assume 1), the winning probability for the max player at a leaf node with value $h$ is given by:

$$p = \int_{x>0} \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{(x-h)^2}{2\sigma^2} \right) \, dx, \quad (3)$$

because $x > 0$ means that the max player can win the game.

In this paper we give the winning probability based on the following.

First, we prepared 300 records of Othello games, and made game trees with depth $D(D = 2, 3$ and $5)$ at turn number $S(S = 40, 46, 50, 53, 54)$. 

![Fig. 3. Game tree search by the Bayesian Method.](image)

![Fig. 4. Back-up values and their probabilities at position U.](image)
Let \( x \) be the difference between the number of black and white chips at a leaf node and \( y \) be that at the end of the game when both the max and min players chose their best moves from the leaf node. We can attain \( ax + b \) as the least squares estimator of \( y \) and the residual sum of squares \( \sigma^2 \) by a regression analysis. Table 1 shows the results of the regression analysis. Figure 5 shows a histogram of \( h_i - g_i \) at \( S = 46 \) and \( D = 3 \), where \( h_i \) is estimated from the regression equation and \( g_i \) is the final value of the game if both players play optimally. \( f(x) \) is the density function of the normal distribution \( N(0, \sigma^2) \). Table 2 shows the results of a Kolmogorov-Smirnov test of whether the data are distributed as a normal distribution or not. * denotes that the value is less than the critical point of 5 percent, 0.078410. These results mean that when a leaf node has an evaluation value \( h \), the back-up value \( g \) by searching from the leaf node

### Table 1. Regression coefficient and residual sum of squares.

<table>
<thead>
<tr>
<th>S</th>
<th>40</th>
<th>46</th>
<th>50</th>
<th>53</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>D=2</td>
<td>-0.5208942</td>
<td>-0.6916849</td>
<td>0.0280726</td>
<td>0.1793199</td>
</tr>
<tr>
<td></td>
<td>D=3</td>
<td>-0.9659944</td>
<td>-0.7218563</td>
<td>0.1606241</td>
<td>0.2787119</td>
</tr>
<tr>
<td></td>
<td>D=5</td>
<td>-0.1799118</td>
<td>-0.3235347</td>
<td>0.4893653</td>
<td>0.5710697</td>
</tr>
<tr>
<td>b</td>
<td>D=2</td>
<td>10.16445</td>
<td>12.59766</td>
<td>10.83266</td>
<td>-8.504321</td>
</tr>
<tr>
<td></td>
<td>D=3</td>
<td>-9.499308</td>
<td>0.07493650</td>
<td>-1.673551</td>
<td>0.8806797</td>
</tr>
<tr>
<td></td>
<td>D=5</td>
<td>-17.76279</td>
<td>-2.896933</td>
<td>-3.076337</td>
<td>1.224023</td>
</tr>
<tr>
<td>( \sigma^2 )</td>
<td>D=2</td>
<td>297.6070</td>
<td>192.5029</td>
<td>294.7741</td>
<td>202.2471</td>
</tr>
<tr>
<td></td>
<td>D=3</td>
<td>250.6166</td>
<td>192.5029</td>
<td>320.0529</td>
<td>215.8179</td>
</tr>
<tr>
<td></td>
<td>D=5</td>
<td>150.5699</td>
<td>343.5041</td>
<td>288.4721</td>
<td>153.1623</td>
</tr>
</tbody>
</table>

### Table 2. Kolmogorov-Smirnov test.

<table>
<thead>
<tr>
<th>D</th>
<th>S = 40</th>
<th>S = 46</th>
<th>S = 50</th>
<th>S = 53</th>
<th>S = 54</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.031378*</td>
<td>0.052140*</td>
<td>0.055262*</td>
<td>0.062915*</td>
<td>0.040333*</td>
</tr>
<tr>
<td>3</td>
<td>0.041966*</td>
<td>0.027502*</td>
<td>0.063307*</td>
<td>0.054245*</td>
<td>0.056167*</td>
</tr>
<tr>
<td>5</td>
<td>0.059386*</td>
<td>0.092301</td>
<td>0.109068</td>
<td>0.052004*</td>
<td>0.063826*</td>
</tr>
</tbody>
</table>

Fig. 5. Histogram of \( h_i - g_i \) at \( S = 46 \) and \( D = 3 \).
to the end of the game is approximately distributed as a normal distribution $N(h, \sigma^2)$. Therefore, let the evaluation function be $h = ax + b$, where $x$ is the difference between the number of black and white chips at a leaf node and $a$ and $b$ are the values represented in Table 1. The winning probabilities for the Probability Method are given by Eq. (3).

Next we obtain the empirical distribution for the Bayesian Method from the histogram of $h_0 - g_0$ in Fig. 5. When the evaluation value is $h_0$, the probability that the final value of the game will be $g_0$ is obtained from $y_0/n$, where $y_0$ is the value on the $y$ axis at $x_0 = h_0 - g_0$ on the $x$ axis for the histogram and $n$ is the total number of nodes.

We use the evaluation function $ax + b$ for the Backward Induction Method, where $a$ and $b$ are given in Table 1 and $x$ is the difference between the number of black and white chips at a leaf node.

Now we compare the Probability Method with the Bayesian and Backward Induction Methods by evaluating how often these methods can select a winning move. We also compare the search times of these methods.

In the following we describe the experimental process.

1. We prepare 300 records of Othello games which are different to those used the regression analysis.
2. The max player chooses his next move by searching using the Probability, Bayesian and Backward Induction Methods with depth $D (D = 2, 3, 5)$. We use the winning probabilities for the Probability Method, the evaluation values for the Backward Induction Method, and the empirical distribution for the Bayesian Method which are all shown in Table 1.
3. We compare the moves chosen by three methods in the following way. From the moves chosen by three methods, we search the game to the end in order to find out whether the moves result in a win, lose or draw if the both players play optimally. We compare how often each method chooses a winning move. We also compare the total search time for 300 records for each method. We use an AMD athlon(tm) 64 Processor 3000+ (2.00 GHz) with 512 MB RAM.

Tables 3 and 4 show the results of the experiments. Table 3 represents the following percentage, $PNT$, of the winning moves selected by each method:

$$PNT = \frac{W + 1/2 \times D}{T} \times 100,$$

where $W$ is the number of moves resulting in a win, $D$ is the number of moves that result in a draw, and $T$ is the total number of records.

<table>
<thead>
<tr>
<th>Table 3. Percentages of winning moves.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
</tr>
<tr>
<td>----------------------------------------</td>
</tr>
<tr>
<td><strong>Probability Method</strong></td>
</tr>
<tr>
<td>$D = 2$</td>
</tr>
<tr>
<td>$D = 3$</td>
</tr>
<tr>
<td>$D = 5$</td>
</tr>
<tr>
<td><strong>Bayesian Method</strong></td>
</tr>
<tr>
<td>$D = 2$</td>
</tr>
<tr>
<td>$D = 3$</td>
</tr>
<tr>
<td>$D = 5$</td>
</tr>
<tr>
<td><strong>Backward Induction Method</strong></td>
</tr>
<tr>
<td>$D = 2$</td>
</tr>
<tr>
<td>$D = 3$</td>
</tr>
<tr>
<td>$D = 5$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4. Search times (seconds).</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
</tr>
<tr>
<td>----------------------------------</td>
</tr>
<tr>
<td><strong>Probability Method</strong></td>
</tr>
<tr>
<td>$D = 2$</td>
</tr>
<tr>
<td>$D = 3$</td>
</tr>
<tr>
<td>$D = 5$</td>
</tr>
<tr>
<td><strong>Bayesian Method</strong></td>
</tr>
<tr>
<td>$D = 2$</td>
</tr>
<tr>
<td>$D = 3$</td>
</tr>
<tr>
<td>$D = 5$</td>
</tr>
<tr>
<td><strong>Backward Induction Method</strong></td>
</tr>
<tr>
<td>$D = 2$</td>
</tr>
<tr>
<td>$D = 3$</td>
</tr>
<tr>
<td>$D = 5$</td>
</tr>
</tbody>
</table>
From the results of the experiments, we can confirm the following:

1. The percentages of winning moves selected by the Probability and Bayesian Methods are greater than that given by the Backward Induction Method in all cases except $S = 54$ and $D = 5$. However, the percentage of winning moves for the Probability and Bayesian Methods are not very different.

2. It is noticed that the greater the depth $D$ and the larger the number of turns $S$ become, the greater the percentage of winning moves becomes (Figs. 6 and 7). It is considered that the values given at the leaf nodes more correctly reflect the result of the game due to the fact that the leaf nodes are nearer to the end of the game.
3. The search times for the Probability and Backward Induction Methods are less than the Bayesian Method. However, the search time for the Probability and Backward Induction Methods are not much different.

4. Therefore, the Probability Method is better than the Backward Induction Method in terms of the percentage of winning moves and is better than the Bayesian Method with respect to the search time.

3. Comparison with Alpha-Beta Pruning

In the previous section, we showed that the Probability Method is better than the Backward Induction Method in terms of the percentage of winning moves.

However, the Backward Induction Method uses the alpha-beta pruning procedure which is given by the following algorithm (expressed in an ad-hoc ALGOL-like language) [4].

```algol
integer procedure B(node p, integer alpha, integer beta);
begin integer i, d, m, t;
determine the successor nodes p1, ..., pd;
if d = 0 then B := f(p) else
begin m := alpha;
for i := 1 step 1 until d do
begin t := B(p, -beta, -m);
if m > t then m := t;
if m ≥ beta then go to done;
end;
done: B := m;
end;
end
```

The alpha-beta pruning procedure is generally used to speed up such search processes without loss of information.

In this section, we propose a pruning procedure for the Probability Method and compare it with the alpha-beta pruning procedure.

3.1 Pruning procedure for the probability method

The following algorithm computes \( R(p) \) and \( F(p) \) which are the winning probabilities for the player who is moving from node \( p \).

```algol
real procedure R(node p);
begin integer i, d, real m, t;
determine the successor nodes p1, ..., pd;
if d = 0 then R := f(p) else
begin m := ∞;
for i := 1 step 1 until d do
begin t := F(p);
if m > t then m := t;
end;
R := 1 - m;
end;
end

real procedure F(node p);
begin integer i, d, real m;
determine the successor nodes p1, ..., pd;
if d = 0 then F := f(p) else
begin m := 1;
for i := 1 step 1 until d do
m := m * F(p);
F := 1 - m;
end;
end
```
real procedure $R_1$ (node $p$);
begin integer $i, d$; real $m, t$;
determine the successor nodes $p_1, \ldots, p_d$;
if $d = 0$ then $R_1 := f(p)$ else
begin $m := \infty$;
for $i := 1$ step 1 until $d$ do
begin $t := F_1(p_i, m)$;
if $m > t$ then $m := t$;
end;
end;
end;

real procedure $F_1$ (node $p$, real $\alpha$);
begin integer $i, d$; real $m$;
determine the successor nodes $p_1, \ldots, p_d$;
if $d = 0$ then $F_1 := f(p)$ else
begin $m := 1$;
for $i := 1$ step 1 until $d$ do
begin $m := m * F_1(p_i, \infty)$;
if $\alpha \leq 1 - m$ then go to done;
end;
done: $F_1 := 1 - m$;
end;
end;

Here $\infty$ denotes a value greater than 1.

Figure 8 shows the value of $R$ for the root node, and the values of $F$'s for the other nodes which are computed from the values of $f$. Figure 9 shows the same situation evaluated by procedure $R_1$.

Exploring node B, we have $F(B) = 0.12$. Moreover, by exploring node E, we can deduce that

$$F(C) = 1 - F(E) * F(F) \geq 1 - F(E) = 0.2.$$ 

Then we have $\text{min}(F(B), F(C)) = 0.12$ without the need to explore node F. Therefore we have

$$R_1(A) = 1 - \text{min}(F(B), F(C)) = 0.88.$$ 

We can prove $R_1(p) = R(p)$ for any root node $p$ by the following lemma and theorems.
Lemma  \( F_1(p, \infty) \) is executed for any node \( p \) with depth \( n \geq 2 \), and we have \( F_1(p, \infty) = F(p) \).

Proof. If \( q \) is the parent node of node \( p \), the program \( F_1(q, \alpha) \) calls \( F_1(p, \infty) \). In the program \( F_1(p, \infty) \), \( \alpha \) is greater than \( 1 - m \) because \( \alpha = \infty \) and \( 0 \leq m \leq 1 \). Therefore we have \( F_1(p, \infty) = F(p) \) by induction on the depth of \( p \). □

Theorem 1. For any node \( p \) with depth 1, if \( F(p) < \alpha \), then \( F_1(p, \alpha) = F(p) \) and if \( F(p) \geq \alpha \), then \( F_1(p, \alpha) \geq \alpha \).

Proof. For any \( j(1 \leq j \leq d) \), we have \( m = \prod_{i=1}^{j} F(p_i) \) at \( 'm := m \ast F_1(p_i, \infty)' \) of the the \( j \)-th iteration of the for loop in the program \( F_1(p, \alpha) \), because we have \( F_1(p_i, \infty) = F(p_i) \) by Lemma.

i) In the case of \( F(p) < \alpha \), we have

\[
F(p) = 1 - \prod_{i=1}^{d} F(p_i) = 1 - \prod_{i=1}^{d} F(p_i, \infty) < \alpha.
\]

Since \( 1 - \prod_{i=1}^{d} F_1(p_i, \infty) \) is a monotonously increasing function of \( j \), the program does not go to ‘done’ from ‘if \( \alpha \leq 1 - m \) then go to done;’. Therefore we have

\[
F_1(p, \alpha) = 1 - \prod_{i=1}^{d} F_1(p_i, \infty) = 1 - \prod_{i=1}^{d} F(p_i) = F(p).
\]

ii) In the case of \( F(p) \geq \alpha \), we have

\[
F(p) = 1 - \prod_{i=1}^{d} F(p_i) = 1 - \prod_{i=1}^{d} F_1(p_i, \infty) \geq \alpha.
\]

Since \( 1 - \prod_{i=1}^{d} F_1(p_i, \infty) \) is monotonously increasing function of \( j \), there exists a value of \( i \) for which the program goes to ‘done’ from ‘if \( \alpha \leq 1 - m \) then go to done;’ of the \( i \)-th iteration of the for loop. □

Theorem 2. For any root node \( p \), we have \( R_1(p) = R(p) \).

Proof. Let \( p_1, \ldots, p_d \) be the successor nodes of a root node \( p \). We show that the invariant condition

\[
m = \min(F(p_1), \ldots, F(p_{i-1}))
\]

is satisfied at the beginning of the \( i \)-th iteration of the for loop of the program \( R_1(p) \), where the minimum operation over an empty set is conventionally defined to be \( \infty \).

When Eq. (4) holds for \( i = j \), we can show that it also holds for \( i = j + 1 \) as follows:

i) In the case of \( F(p_j) \geq m \), the value \( m \) is not changed at ‘if \( m > t \) then \( m := t;’ because \( F_1(p_j, m) \geq m \) by Theorem 1. Therefore we have \( m = \min(F(p_1), \ldots, F(p_{i-1})) = \min(F(p_1), \ldots, F(p_j)) \) at the beginning of the \( j + 1 \)-th iteration of the for loop.

ii) In the case of \( F(p_j) < m \), we have \( F_1(p_j, m) = F(p_j) < m \) by Theorem 1. The value \( m \) is changed to \( F_1(p_j, m) \) at ‘if \( m > t \) then \( m := t;’ Therefore we have \( m = (F(p_j, m) = F(p_j)) = \min(F(p_1), \ldots, F(p_j)) \) at the beginning of the \( j + 1 \)-th iteration of the for loop.
So we have $R_1(p) = R(p) = 1 - m$. 

In the following, we can gain the minimum number of terminal nodes which are examined by $R_1$.

**Theorem 3.** Consider a complete $d$-tree with depth $D$. For every node $p$ of the tree, let $L(p)$ be the minimum number of terminal nodes examined from the node $p$ by $R_1$. Then we have $L(p) = 2d^{D-1} - d^{D-2}$ for the root node $p$ of the given tree.

**Proof.** Let $p_1, \ldots, p_d$ be the successor nodes of $p$. All descendant nodes of the node $p_1$ are examined by $R_1$. So we have $L(p_1) = d^{D-1}$. Let $p_{i1}, \ldots, p_{id}$ be the successor nodes of $p_i (2 \leq i \leq d)$. If $F_1(p_i, \infty) \leq 1 - F_1(p_{i1}, \infty)$ for every $i (2 \leq i \leq d)$, then the node $p_i (2 \leq i, j \leq d)$ is not examined by $R_1$. So we have $L(p_i) = L(p_{i1}) = d^{D-2}(2 \leq i \leq d)$. Therefore we have $L(p) = L(p_1) + \sum_{i=2}^{d} L(p_i) = d^{D-1} + (d-1)d^{D-2} = 2d^{D-1} - d^{D-2}$.

Since the minimum number of terminal nodes examined from the node $p$ by the alpha-beta pruning [4] is $d^{\lfloor \frac{D}{2} \rfloor} + d^{\lfloor \frac{D}{2} \rfloor} - 1$, where $\lfloor x \rfloor$ and $\lceil x \rceil$ signify counting and omitting fractions below decimals of $x$, respectively, then the alpha-beta pruning is more powerful than $R_1$.

**3.2 Comparison of the pruning procedures**

In this section we compare the searching times of the pruning procedure $R_1$ for the Probability Method and the

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>40</th>
<th>46</th>
<th>50</th>
<th>53</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Probability Method</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D = 2$</td>
<td>0.312</td>
<td>0.218</td>
<td>0.172</td>
<td>0.157</td>
<td>0.156</td>
<td></td>
</tr>
<tr>
<td>$D = 3$</td>
<td>1.719</td>
<td>0.843</td>
<td>0.453</td>
<td>0.281</td>
<td>0.250</td>
<td></td>
</tr>
<tr>
<td>$D = 5$</td>
<td>115.953</td>
<td>27.156</td>
<td>6.484</td>
<td>1.516</td>
<td>0.875</td>
<td></td>
</tr>
<tr>
<td><strong>R1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D = 2$</td>
<td>0.281</td>
<td>0.203</td>
<td>0.172</td>
<td>0.141</td>
<td>0.125</td>
<td></td>
</tr>
<tr>
<td>$D = 3$</td>
<td>1.469</td>
<td>0.688</td>
<td>0.422</td>
<td>0.250</td>
<td>0.188</td>
<td></td>
</tr>
<tr>
<td>$D = 5$</td>
<td>113.734</td>
<td>15.234</td>
<td>4.234</td>
<td>1.140</td>
<td>0.672</td>
<td></td>
</tr>
<tr>
<td><strong>Backward Induction Method</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D = 2$</td>
<td>0.297</td>
<td>0.234</td>
<td>0.188</td>
<td>0.156</td>
<td>0.156</td>
<td></td>
</tr>
<tr>
<td>$D = 3$</td>
<td>1.703</td>
<td>0.844</td>
<td>0.454</td>
<td>0.281</td>
<td>0.250</td>
<td></td>
</tr>
<tr>
<td>$D = 5$</td>
<td>116.078</td>
<td>27.203</td>
<td>6.484</td>
<td>1.515</td>
<td>0.891</td>
<td></td>
</tr>
<tr>
<td><strong>Alpha-beta pruning</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D = 2$</td>
<td>0.094</td>
<td>0.094</td>
<td>0.094</td>
<td>0.094</td>
<td>0.109</td>
<td></td>
</tr>
<tr>
<td>$D = 3$</td>
<td>0.187</td>
<td>0.141</td>
<td>0.125</td>
<td>0.125</td>
<td>0.125</td>
<td></td>
</tr>
<tr>
<td>$D = 5$</td>
<td>1.562</td>
<td>0.781</td>
<td>0.422</td>
<td>0.219</td>
<td>0.171</td>
<td></td>
</tr>
</tbody>
</table>
alpha-beta pruning for the Backward Induction Method.

The experimental data are the same as in Section 2.2. Table 5 shows the results of the experiments. Alpha-beta pruning is more powerful than R1. This is typical for a deep search. R1 shows a slight improvement over the Probability Method. The search times of R1 with depth 3 are nearly equal to those of alpha-beta pruning with depth 5.

In Table 6 and Fig. 10 we compare the search times and percentages of winning moves (see Table 3) of R1 with depth 3 and alpha-beta pruning with depth 5. We can see that the percentage of winning moves of R1 with depth 3 is greater than that of alpha-beta pruning with depth 5 for \( S = 40, 46 \) and 50 while the search time of R1 with depth 3 is less than or equal to that of the alpha-beta pruning with depth 5.

We also compare the search time and percentage of winning moves of R1 with depth 5 and alpha-beta pruning with depth 9 in Table 7 and Fig. 11. The search time of R1 with depth 5 is less than that of alpha-beta pruning with depth 9. At \( S = 40 \) and 46, the percentage of winning moves of R1 with depth 5 is greater than that of the alpha-beta pruning with depth 9.

<table>
<thead>
<tr>
<th>( S )</th>
<th>40</th>
<th>46</th>
<th>50</th>
<th>53</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R1 ((D=3))</strong></td>
<td>Search times</td>
<td>1.469</td>
<td>0.688</td>
<td>0.422</td>
<td>0.250</td>
</tr>
<tr>
<td></td>
<td>Percentage of winning moves</td>
<td>61.7 62.8 71.8 79.2 78.8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| **Alpha-beta \((D=5)\)** | Search times | 1.562 | 0.781 | 0.422 | 0.219 | 0.171 |
|  | Percentage of winning moves | 51.5 57.2 69.2 82.7 84.2 |

<table>
<thead>
<tr>
<th>( S )</th>
<th>40</th>
<th>46</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R1 ((D=5))</strong></td>
<td>Search times</td>
<td>113.734</td>
<td>15.234</td>
</tr>
<tr>
<td></td>
<td>Percentage of winning moves</td>
<td>67.3 69.3 80.3</td>
<td></td>
</tr>
<tr>
<td><strong>Alpha-beta ((D=9))</strong></td>
<td>Search times</td>
<td>203.343</td>
<td>37.438</td>
</tr>
<tr>
<td></td>
<td>Percentage of winning moves</td>
<td>58.3 65.3 91.2</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 11. Comparison of R1 \((D=5)\) and alpha-beta pruning \((D=9)\).
4. Conclusions

We make the following conclusions of this paper.

1) We made experimental comparison of the Probability Method with the Backward Induction and Bayesian Methods using the Othello game. The items compared were the percentage of winning moves selected by each method and their search times. The Probability and Bayesian Methods are better than the Backward Induction Method in terms of the percentage of winning moves. The search times of the Bayesian Method are greater than those of the Probability and Backward Induction Methods. Therefore we were able to show that the Probability Method is more effective than both the Backward Induction and Bayesian Methods for the combined percentage of winning moves and search time.

2) We also proposed a pruning procedure, R1, for the Probability Method and compared it with the alpha-beta pruning procedure for the Backward Induction Method. The experiments show that alpha-beta pruning is more powerful than R1, but R1 is better in some phases. We have, therefore, to propose a more powerful pruning procedure for the Probability Method.

3) In Section 2, we assumed that we can search till the end of the game from all nodes except the root node. Nevertheless, the experimental results show that the Probability Method can choose more winning moves than the Backward Induction Method whether this assumption is satisfied or not. Therefore, we should also be able to apply the Probability Method to the other two person games such as Chess, Shogi and Go.

REFERENCES


