TECHNICAL REPORT NO. 329

Nonminimax Propagation and Go Moku

by

Jong Chan Kim

and

Paul W. Purdom, Jr.

April 1991

COMPUTER SCIENCE DEPARTMENT

INDIANA UNIVERSITY

Bloomington, Indiana 47405-4101

Nonminimax Propagation and Go Moku

Jong Chan Kim and Paul Walton Purdom, Jr.
Department of Computer Science
Indiana University
Bloomington, Indiana 47405

April 5, 1991

Abstract An experimental investigation of propagation methods with the game Five-In-A-Row shows cases where nonminimax propagation methods usually beat minimax. Previous research on the minimax pathology has shown that minimax is not the optimum way to back up the estimates of the value of game positions, but the previous work concentrated on games that were devised to illustrate the phenomenon. This paper shows that an advantage comes from using a nonminimax propagation methods with a common game.

1 Introduction

The standard approach to game-playing is based on a partial game-tree search algorithm, MINIMAX. Computer programs playing games like chess and checkers improve their performances by increasing the search-depth. Recently, computers such as DEEP THOUGHT and HITECH have beat Grandmaster-level human players. This was achieved through deeper search resulting from the increased computing speed obtained by special hardware. The experience with chess programs is that deeper search leads to better play [8]. On the other hand, Nau [3, 5], Beal [2], and Pearl [12] showed that for *some* reasonable games and evaluation functions, there is a *minimax pathology*, where the quality of the move selected by minimax propagation gets worse as the search is made deeper. In a uniform binary game tree with a probabilistic evaluation function, the rate of error amplification per game cycle increases as the square root of the number of alternative moves available to each player [10].

This paper is concerned with propagation methods and the quality of static evaluation functions on the game Five-In-A-Row (known as OH-MOK, GO MOKU, or PENTE). Although no evidence of minimax pathology was found, cases were found where propagating a weighed average of the best moves was significantly better than minimax propagation. This shows that avoiding the error amplification of minimax can be important in real games.

2 Game Description

The game Five-In-A-Row is a two-person zero-sum perfect-information game. The game board has 19 rows and 19 columns, so that the board consists of 361 points. From left to right, numbers 1 to 19 are used to represent the x-coordinates. From top to bottom, numbers 1 to 19 are used to represent y-coordinates. A point is represented by x- and y-coordinates: (x, y). One player uses white stones, and the other player uses the black stones. The players alternate placing one stone at a time on any unoccupied point on the board. The player who first puts five adjacent stones in a row in vertical, horizontal, or diagonal directions wins.

This game is essentially the same as the game GO MOKU, which has been played for hundreds of years. However, modern GO MOKU has a few additional rules to compensate for the advantage of moving first. We did not consider these additional rules.

3 Static Evaluation Function

In our program, the static evaluation function assigns a value to each linear chain consisting of stones of one color. The chain may contain empty spaces, but it can not contain stones of the other color. Each chain is specified by a starting stone and a direction. (Chains can overlap.) Each chain has a left termination point and a right termination point consisting of a stone of the wrong color or of the edge of the board. (The termination points for vertical chains are actually top and bottom points.) When the distance to the termination point is more than 5, it is treated as though it were 5. A chain is *open* if both ends are at distance five. (Distance five gives just enough room to form a chain of five stones beginning with the starting stone.) A chain is *dead* when the distance between the ends is less than five. All other chains are *closed*. Each chains have a value $V_{o,s}$ that depends the openness, o, and the number of stones, s, of the chain. Dead chains have a value of 0. All other things being equal, a chain with a large number of stones is much more valuable than one with a small number of stones. Also, open chains are more valuable than closed ones. Details of the values we assigned to chains are given in the next section.

The value of the white stones (V_W) is the sum of the values of all chains containing white stones, while the value of black stones (V_B) is the sum of the values of all chains containing black stones. To evaluate the value of the board in a position with white to move, we use the formula

$$V = V_W - dV_B,$$

where d is a *defense factor*. When evaluating a position with black to move we use the formula

$$V = V_B - dV_W.$$

(Since the program does a fixed depth search, it never needs to compare values generated with these two formulas.)

The defense factor improves the quality of the evaluation function. With a defense factor of 1, a well optimized evaluation function for odd numbers of depths is too offensive for even numbers of depths and also a well optimized evaluation function for even numbers of depths is too defensive for odd numbers of depths. With a fixed evaluation function, as the defense factor increases up to an optimal value, the number of mistakes is reduced, so that generally the quality of decision making is improved. When two players use very large defense factors, they become so defensive that the game ends in a draw. Generally, odd numbers of search depths need small defense factors, and even numbers of search depths need large defense factors.

To speed up the program, an incremental version of the evaluation function is used. The placing of a new stones changes only a few chains. The new value of a position is calculated from the old value plus the change in value that results form the altered chains.

4 Adjustment of Parameters

The data set used in this experiment has 32 initial board configurations. For each initial state, 2 games are played: one player starts first, and then the other player starts first, initiating a total of 64 games.

Sets of games were played and the values of d and $V_{o,s}$ were adjusted to obtain good play. To reduce the number of parameters, the values of $V_{o,s}$ were required to be of the form

$$V_{open,i} = a * b^i,$$

$$V_{closed,i} = b^i,$$

$$V_{dead,i} = 0.$$

During the adjustment of the parameters minimax propagation with 2 ply lookahead was always used. We found that $a = 1, b = 64, d = 64$ leads to fairly good play. The set $a = 1, b = 64, d = 256$ is even better, but it leads to overflow problems when used with 3 ply search. In 3 ply search the set $a = 1, b = 64, d = 1$ is better than the set $a = 1, b = 64, d = 64$. The appendix shows the results from selected runs used for parameter adjustment.

5 Propagation Methods

The main objective of this experiment is to see how various propagation methods work. After a given depth of search, the evaluation values of terminal nodes propagated to the starting position. The propagation value of each nonterminal node is determined from evaluation values of its children. Actual implementation used NEG-MAX notation. Five maximum values are considered among sorted sibling nodes: EV_1, EV_2, EV_3, EV_4, and EV_5. EV_i is the i^{th} maximum value. The propagation value of a node is decided by $\sum_{i=1}^{5} weight_i \times EV_i$. A set of $weight$'s decides a propagation method.

In the minimax propagation, the only maximum value is propagated. This value is expressed by:

$$MM = 1.0 \times EV_1 + 0.0 \times EV_2 + 0.0 \times EV_3 + 0.0 \times EV_4 + 0.0 \times EV_5$$

The other propagation methods considered are:

$$P81100 = 0.8 \times EV_1 + 0.1 \times EV_2 + 0.1 \times EV_3 + 0.0 \times EV_4 + 0.0 \times EV_5$$

$$P52111 = 0.5 \times EV_1 + 0.2 \times EV_2 + 0.1 \times EV_3 + 0.1 \times EV_4 + 0.1 \times EV_5$$

$$P43210 = 0.4 \times EV_1 + 0.3 \times EV_2 + 0.2 \times EV_3 + 0.1 \times EV_4 + 0.0 \times EV_5$$

Every propagation method competes with the other ones, thus there were 6 pairs of competitions: MM versus P81100, MM versus P52111, MM versus P43210, P81100 versus P52111, P81100 versus P43210, and P52111 versus P43210.

6 Results and Discussion

In order to find out how propagation methods work when the quality of evaluation functions differs, three sets of parameters are used: PS1, PS2, and PS3. For PS1, $a = 1, b = 1$, and $d = 1$. For PS2, $a = 1, b = 64$, and $d = 1$. For PS3, $a = 1, b = 64$, and $d = 64$. Table 2 contains the result of competion between sets of parameters on

Parameter Set	a	b	d
PS1	1	1	1
PS2	1	64	1
PS3	1	64	64

Table 1: Three sets of parameters

Depth	Propagation Constant	Set	TW	PW	Set	TW	PW
2	MM	PS1	0	0	PS2	64	32
2	P81100	PS1	0	0	PS2	64	32
2	P52111	PS1	0	0	PS2	64	32
2	P43210	PS1	0	0	PS2	64	32
2	Total	PS1	0	0	PS2	256	128
2	MM	PS2	31	0	PS3	33	1
2	P81100	PS2	16	0	PS3	48	16
2	P52111	PS2	20	3	PS3	43	14
2	P43210	PS2	16	1	PS3	48	17
2	Total	PS2	83	4	PS3	172	48
3	MM	PS1	0	0	PS2	64	32
3	MM	PS2	55	24	PS3	9	1

Table 2: Comparisons between parameter sets

some fixed depths and propagation methods. *Depth* means the lookahead depth. *Set* is the parameter sets in Table 1. *TW* means the number of total winnings, and *PW* means the number of pair winnings. As shown in Table 2, in 2-ply search, PS3 is, in decision quality, better than PS2, and PS2 is better than PS1. In 3-ply search, the PS2 is better than PS3, and the PS3 is better than PS1.

In the tables below, *p-value* means the probability that random play would result in one of the players winning at least m out of n games. (This is a two-tailed test.) Thus, the *p-value* is

$$2^{-n}\left(\sum_{i\geq m}\binom{n}{i}+\sum_{i\leq n-m}\binom{n}{i}\right),$$

where $m > n/2$. If $m = n/2$, the $i = n/2$ term is included in only one of the sums [9]. A small p-value suggest that winning program is better than the other one, rather

Winner					*Loser*		
Propagation Constant	*TW*		*PW*		*Propagation Constant*	*TW*	*PW*
	No	*p-value*	*No*	*p-value*			
MM	34	0.71	7	0.77	P81100	30	5
MM	33	0.90	8	1.00	P52111	31	7
MM	36	0.38	10	0.45	P43210	28	6
P52111	34	0.71	4	0.69	P81100	30	2
P81100	33	0.90	4	1.00	P43210	31	3
P43210	35	0.53	3	0.25	P52111	29	0

Propagation Constants	*MM*	*P81100*	*P52111*	*P43210*
Total of TW	103	93	94	94
Total of PW	25	11	11	12

Table 3: Winners for $a = 1$, $b = 1$, $d = 1$, and $l = 2$ ply

than just lucky.

Parameter sets PS1, PS2, and PS3 are used in Table 3, 4, and 5 respectively, and PS2 and PS3 are used in Table 6 and 7 respectively. Each of these tables contains the result of competition between propagation methods, and uses a fixed a, b, d, and l. Tables 5 and 6 use well-adjusted parameters. They also show results that are significant at the 5% level. At the lookahead depth 2, minimax propagation is significantly better than the other propagation methods. At the lookahead depth 3, minimax propagation is usually worse than the other propagation methods. The depth 3 results show the error amplification that can occur when minimax propagation is used with heuristic evaluation functions.

Even though deep search has proven beneficial in some real games, it does not mean that the pessimistic effect of the minimax pathology can be ignore in game programming. Error amplification due to minimaxing is an established fact. It may significantly degrade the quality of decisions in practical games [10]. More appropriate propagation rules can make substantial improvements in decision quality by minimizing this deterioration. Recently Ingo Althöfer [1] reported that in game tree models with stochastically distributed evaluation error an incremental negamax algorithm reduced the minimax pathology.

Winner					*Loser*		
Propagation	*TW*		*PW*		*Propagation*		
Constant	*No*	*p-value*	*No*	*p-value*	*Constant*	*TW*	*PW*
MM	36	0.38	11	0.48	P81100	28	7
MM	38	0.17	9	0.15	P52111	26	3
MM	39	0.10	15	0.21	P43210	25	8
P52111	34	0.71	5	0.73	P81100	30	3
P81100	33	0.90	5	1.00	P43210	31	4
P52111	32	1.00	3	1.00	P43210	32	3

Propagation Constants	*MM*	*P81100*	*P52111*	*P43210*
Total of TW	113	91	92	88
Total of PW	35	15	11	15

Table 4: Winners for $a = 1$, $b = 64$, $d = 1$, and $l = 2$ ply

Winner					*Loser*		
Propagation	*TW*		*PW*		*Propagation*		
Constant	*No*	*p-value*	*No*	*p-value*	*Constant*	*TW*	*PW*
MM	45	1.56×10^{-3}	14	9.77×10^{-4}	P81100	19	1
MM	45	1.56×10^{-3}	14	9.77×10^{-4}	P52111	19	1
MM	46	6.17×10^{-4}	16	1.31×10^{-3}	P43210	18	2
P81100	35	0.45	4	0.38	P52111	28	1
P43210	35	0.53	4	0.38	P81100	29	1
P43210	37	0.26	5	6.25×10^{-2}	P52111	27	0

Propagation Constants	*MM*	*P81100*	*P52111*	*P43210*
Total of TW	136	83	74	90
Total of PW	44	6	2	11

Table 5: Winners for $a = 1$, $b = 64$, $d = 64$, and $l = 2$ ply

Winner					*Loser*		
Propagation	*TW*		*PW*		*Propagation*		
Constant	*No*	*p-value*	*No*	*p-value*	*Constant*	*TW*	*PW*
P81100	38	0.17	10	0.18	MM	26	4
P52111	41	3.28×10^{-2}	12	3.52×10^{-2}	MM	23	3
P43210	37	0.26	10	0.30	MM	27	5
P52111	44	3.69×10^{-3}	12	4.88×10^{-4}	P81100	20	0
P43210	46	6.17×10^{-4}	14	1.22×10^{-4}	P81100	18	0
P52111	32	1.00	0	1.00	P43210	32	0

Propagation Constants	*MM*	*P81100*	*P52111*	*P43210*
Total of TW	76	76	117	115
Total of PW	12	10	24	24

Table 6: Winners for $a = 1$, $b = 64$, $d = 1$, and $l = 3$ ply

Winner					*Loser*		
Propagation	*TW*		*PW*		*Propagation*		
Constant	*No*	*p-value*	*No*	*p-value*	*Constant*	*TW*	*PW*
MM	39	0.10	10	9.23×10^{-2}	P81100	25	3
P52111	39	0.10	13	0.17	MM	25	6
P43210	37	0.26	13	0.38	MM	27	8
P52111	34	0.71	9	0.80	P81100	30	7
P43210	34	0.71	8	0.79	P81100	30	6
P43210	34	0.53	8	0.58	P52111	29	5

Propagation Constants	*MM*	*P81100*	*P52111*	*P43210*
Total of TW	91	85	102	105
Total of PW	24	16	27	29

Table 7: Winners for $a = 1$, $b = 64$, $d = 64$, and $l = 3$ ply

A Winners for parameter adjustment

The table below contains some result selected from competition between the parameter set ($a = 1, b = 64$, and $d = 64$) and other parameter sets which differ in one of a, b, d from the parameter set. Lookahead depth 2 is used. In the table, TW means the number of total winnings, and PW means the number of pair winnings.

Winner					*Loser*				
a	b	d	TW	PW	a	b	d	TW	PW
1	64	64	40	13	2	64	64	11	0
1	64	64	26	8	4	64	64	14	1
1	64	64	31	10	8	64	64	6	0
1	64	64	31	10	16	64	64	7	0
1	64	64	28	5	32	64	64	7	0
1	64	64	64	32	1	1	64	0	0
1	64	64	64	32	1	2	64	0	0
1	64	64	64	32	1	4	64	0	0
1	64	64	64	32	1	8	64	0	0
1	64	64	64	32	1	16	64	0	0
1	64	64	64	32	1	32	64	0	0
1	64	64	32	0	1	128	64	32	0
1	64	64	32	0	1	256	64	32	0
1	64	64	33	1	1	64	1	31	0
1	64	64	33	1	1	64	16	31	0
1	64	64	33	1	1	64	32	31	0
1	64	64	32	0	1	64	64	32	0
1	64	96	40	8	1	64	64	24	0
1	64	128	53	21	1	64	64	11	0
1	64	128	52	20	1	64	64	12	0
1	64	1024	52	20	1	64	64	12	0
1	64	4096	52	20	1	64	64	12	0

References

[1] Althöfer, Ingo, *An incremental negamax algorithm* Artificial Intelligence **43**(1):57–66, 1990.

[2] Beal, D., *An analysis of minimax* In M.R.B. Clarke, editor, Advances in Computer Chess 2, pp. 17–24, Edinburgh University Press, 1980.

[3] Nau, D.S., *Pathology on game trees: A summary of results*, Proc. 1st Nat. Conf. on Artificial Intelligence pp. 102 – 4, 1980.

[4] Nau, D.S., *An investigation of the causes of pathology in games*, Technical Report TR-999, Computer Science Dept., Univ. of MD., 1981.

[5] Nau, D.S., *Pathology on game trees revisited and an alternative to minimaxing.* Artificial Intelligence **21**(1-2):221–244, 1983.

[6] Nau, D.S., Purdom, Paul, and Tzeng, Chung Hung, *Experiments on alternatives to minimax,* University of Maryland Tech. Report No. 1333, Computer Sci. Dept., Univ. of Md., 1983.

[7] Nau, D.S., *The last player theorem*, Artificial Intelligence, 18:53–65, 1982.

[8] Newborn, Monty and Kopec, Danny, *The twentieth annual ACM North American Computer Chess Championship*, CACM **33**(7):92–104, 1990.

[9] Purdom, Paul and Brown, Cynthia, *The Analysis of Algorithms*, Holt, Rinehart, and Winston, 1985.

[10] Pearl, Judea, *Heuristics: Intelligent Search Strategies for Computer Problem Solving,* Addison-Wesley, 1984.

[11] Pearl, Judea, *Asymptotic properties of minimax trees and game-searching procedures,* Artificial Intelligence, **14**(2):113–138, 1980.

[12] Pearl, Judea, *On the nature of pathology in game searching,* Artificial Intelligence, **20**(4):427–453, 1983.