AN EXPLANATION OF AND CURE FOR MINIMAX PATHOLOGY

Bruce Abramson

October 1985
CSD-850034

An Explanation of and Cure for Minimax Pathology *

Bruce Abramson **

Department of Computer Science
Columbia University
New York, N.Y. 10027

Computer Science Department
University of California, Los Angeles
Los Angeles, California 90024

Abstract

The minimax procedure has long been the standard method of evaluating nodes in game trees. The general assumption underlying its use in game-playing programs is that increasing search depth improves play. Recent work has shown that this assumption is not always valid; for a large class of games and evaluation functions, searching deeper decreases the probability of making a correct move. This phenomenon is called game tree pathology.

Two structural properties of game trees have been suggested as causes of pathology: independence among the values of sibling nodes, and uniform depth of wins and losses. This paper examines the relationship between uniform win depth and pathology from two angles. First, it proves mathematically that as search deepens, an evaluation function that does not ask whether wins can be forced from mid-game positions becomes decreasingly likely to choose forced wins. Second, it experimentally illustrates the connection between recognition of mid-game wins and pathological behavior. Two evaluation functions, which differ only in their ability to recognize wins in mid-game, are run on a series of games. Despite recognizing fewer mid-game wins than the theoretically predicted minimum needed to avoid pathology, the function that checked for them cleared up the pathological behavior of the one that did not.

The analytic and empirical aspects of this paper combine to form one major result: As search deepens, so does the probability that failing to check for forced wins will change the game's outcome. This strengthens the hypothesis that uniform win depth is the cause of pathology.

---

* This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165, and by the National Science Foundation under grants IST-84-18879 and IST-85-15302.

** Current address, UCLA

1. Introduction

Computer programs that play two-player games generally adhere to the game-theoretic paradigm, the minimax procedure [Shannon '50], which is optimal for finite two-person zero-sum games of perfect information [LR '67] (p. 71). Unfortunately, tree size precludes the implementation of most popular games according to the specifications of this theory. Theoretically, before play begins, both players can see the entire game tree, including the actual value of each node. In most implementations, however, the trees contain more nodes than can be stored (or viewed) simultaneously — for all practical purpose an infinite number. This forces players to move without knowing all possible completions of the game. Devoid of this information, they can neither effectively compute the true minimax value nor determine the strategy that leads to it. In order to reconcile implementation with theory, minimax has been extended to partial game trees. Statically evaluated tip nodes are treated like leaves; the tree is searched to some arbitrary depth, all nodes at that depth are evaluated, and the estimates are minimaxed back up the tree. The appeal of this procedure is obvious — since minimax is an optimal strategy for finite games, estimated minimax should approximate an optimal strategy for "infinite" games.

Unfortunately, this extension does not estimate the minimax value, it minimaxes estimated values. In general, the two are not equivalent; computing a function of estimates instead of an estimated function is a cardinal sin of statistics. Statistically sound or not, there is a significant collection of game playing programs that attests that not only does minimaxing estimates work, but the deeper the search, (and thus the greater the functional dependence on those estimates), the better the quality of play [Nilsson '80] [Berliner '79]. Nevertheless, a theoretical difficulty with minimaxing estimates was pointed out in [Nau '83b]: for a large class of game trees and evaluation functions, as long as the search does not reach the end of the tree, (in which case a correct decision would be guaranteed), searching deeper causes decisions to become increasingly random. The prediction of games exhibiting this type of pathological behavior suggested two interesting questions: Do any known games belong to this class? And why hasn't pathology been observed in existing game playing programs?

Section 2 discusses work that has been done on board splitting, a game which behaves pathologically when an intuitive, reasonably accurate evaluation function is used. Some differences between board splitting and popular nonpathological games are considered as possible causes of pathology. Section 3 identifies one of these features in the board splitting game tree, and shows that this structural flaw increases the probability of making an error as search deepens. A new evaluation function, which compensates for this unusual feature of board splitting, is introduced. When the game is played minimaxing this new function, the pathology disappears. Section 4 offers some conclusions.

2. A Pathological Game: Board Splitting

Board splitting was devised by Pearl [Pearl '84] as an example of a game whose tree has a uniform branching factor (B), a uniform leaf depth (D), and a random distribution of wins and losses. Play proceeds as follows: a square B^D-by-B^D board is covered with randomly distributed 1's and 0's. The first player splits the board vertically into B sections, keeps one in play, and discards the rest. The second player splits the remaining portion horizontally, doing the same. After D rounds, only one square remains. If that square contains a 1, the horizontal splitter (H) wins. Otherwise, the vertical splitter (V) wins. To compensate V for going first, the board is set up by flipping a coin weighted in her favor, such that a 1 is generated with probability $p < .5$, and a 0 with probability $(1 - p) > .5$. The value of p needed to make the game fair is dependent on B [Pearl '84] *. In order to use board splitting as a model for larger, more complex games, the tree must be treated as if it were too large to be seen in its entirety. The minimax procedure searches the tree to some arbitrary depth, k, where $0 \leq k \leq 2(D-1)$. An heuristic evaluation function is then applied to all nodes at the specified level, and these estimates are minimaxed back up the tree. The search depth is bounded by $2(D-1)$ to insure that neither player can see the last round prematurely. A simple function which has been used in the past assigns each tip node a value equal to the number of 1's it contains [Pearl '84] [Nau '82]. Call this evaluation function N(g). V tries to minimize 1's (thereby maximizing 0's), and H tries to maximize 1's.

$Tip1(g)$ = The number of 1's in g

$$N(g) = \begin{cases} Tip1(g) & \text{if g is a tip node} \\ MIN\{N(g') | g' \text{ is a child of g }\} & \text{if g is a min node} \\ MAX\{N(g') | g' \text{ is a child of g }\} & \text{if g is a max node} \end{cases}$$

Nau showed that N(g) evaluates a given board fairly accurately; the more 1's it contains, the more likely it is to be a win for H, and the smaller the board, the more accurate the evaluation. Nevertheless, programs that use N(g) behave pathologically for sufficiently large B and D. In other words, the probability of correctly choosing the successor node which is most likely to lead to a win is not a monotonically nondecreasing function of search depth; there are cases where searching ahead another round (increasing k by 2) decreases the probability of choosing correctly [Nau '82] [Nau '83a]. This result runs counter to the intuition developed through the observation of thirty years of game-playing programs, in which increased search depth improved play, and constitutes an example of

* Set p equal to the unique solution to the equation $(1-x)^B = x$ in the interval (0,1). A table of solutions to this equation is given in [Nau '82].

the theoretically predicted pathology.

Various cures have been offered for this pathological behavior. Most of them diagnose the minimax procedure as the primary cause, and alleviate pathology by removing minimax. In its place, product propagation rules that estimate the conditional probability of winning the entire game from each node are used [Nau '83a] [NauPT '83] [TP '83]. Although this approach has cured all observed pathologies, it has not answered the basic question: why is the minimax procedure nonpathological in games such as chess and checkers? Two (not necessarily contradictory) conjectures have been forwarded, both focusing on an evaluation function's sensitivity to certain characteristics of a game tree's structure. The first, developed in [Nau '82] and [Nau '83a], considers the meaning of the "strength" of a position. In games like chess and checkers, most board features which influence a position's strength change incrementally. Since any two sibling nodes are only marginally different, their strengths are interdependent. The random setup of board splitting, however, frequently leads to siblings with radically different values. Nau showed that pathological game trees satisfy certain preconditions, one of which is that the value of a tip node may be dependent only to a limited extent on the values of its siblings. This sibling independence, it has been hypothesized, is the cause of pathology.

[Pearl '83] showed that pathology can only be avoided by using evaluation functions whose accuracy improve by over 50% at each successive level in the tree. This type of drastic improvement is unreasonable for most evaluation functions; in general, mid-game nodes are not evaluated that much more accurately than their immediate ancestors. As the search frontier approaches the end-game, however, the estimates rapidly become more accurate. Most common game trees are not uniform in structure. Rather, they are riddled with early terminal positions, or *traps*. Close ancestors of traps, then, are actually end-game, not mid-game nodes, and their estimated values are considerably more reliable than those of other nodes at the same level. Although most evaluation functions are not 50% more accurate for a given node at level $k+1$ than for a given node at level k, the presence of terminal positions in the vicinity of the search frontier significantly improves the function's accuracy when taken over all nodes at the deeper level. Since deeper searches expose more traps, the noise introduced by an additional minimax operation is counterbalanced by increasingly accurate evaluations. This led to the second conjecture: pathology is caused by the absence of traps. According to this hypothesis, the introduction of even a small number of traps may significantly dampen the noise amplification due to minimaxing [Pearl '83]. A useful evaluation function, then, should not only discriminate among nodes based on strength, but detect traps as well.

In an attempt to model common games more realistically, [Pearl '84] dropped the requirement of uniform win depth, and analyzed game trees in which each node was a trap (or, in this case, a terminal node) with probability q. In general, pathology can be avoided

if q exceeds the *critical trap density*, $q_c = 1 - \frac{(1+B)^{1-(1/B)}}{B}$. The next section introduces an evaluation function for board splitting, which, despite having a trap density less than q_c, is nonpathological.

3. Understanding and Curing Pathology

Board splitting, unlike most games, has a uniform game tree. Since all leaves are located at level 2D, there don't seem to be any traps to detect. However, the salient feature of traps is not that they are leaves, but that the values associated with them are exact. Leaves are not the only nodes with this property. Any node that is recognized as a forced win or loss has an exact value as well. Thus, the existence of leaves in mid-tree is not crucial to the avoidance of pathology; the recognition of forced wins can serve the same purpose. There are configurations in board splitting which can easily be identified as forced wins. The most obvious forced wins are boards which contain a row of 1's (win for H), a column of 0's (win for V), or a main diagonal of 1's (win for H, who always goes last). These patterns have two particularly nice properties: they are as obvious to the human designers of a program as they are to the computer that plays it, and they can be checked for at minimal additional overhead while scanning the board to count 1's. Although a reasonable case could be made for the inclusion of other patterns, this decision should not affect the basic result: ***evaluation functions that do not recognize forced wins behave pathologically; those that do, do not.***

The evaluation function described in section 2, N(g), chooses nodes using a single criterion: the number of 1's on the board. Thus, it frequently overlooks forced wins in favor of configurations with more 1's, albeit less strategically arranged. Y(g), shown below, modifies N(g) by taking into account the arrangement, as well as the number, of 1's. Tip nodes are evaluated by checking for a row or diagonal of 1's. If such a row exists, the node is assigned the maximum value of N(g), B^{2D} (the number of squares in the initial board — effectively ∞). This assures that wherever possible, a forced win will be chosen by H and avoided by V. If a column of 0's exists, the value $-B^{2D}$ (effectively $-\infty$) does the opposite, guaranteeing that the node will be chosen by V and avoided by H. Otherwise, the number of 1's is counted, just like in N(g). These values are then minimaxed back up the tree.

$$Tip2(g) = \begin{cases} B^{2D} & \text{if g contains a row or diagonal of 1's} \\ -B^{2D} & \text{if g contains a column of 0's} \\ \text{The number of 1's in g} & \text{Otherwise} \end{cases}$$

$$Y(g) = \begin{cases} Tip2(g) & \text{if g is a tip node} \\ MIN\{Y(g') | g' \text{ is a child of g}\} & \text{if g is a min node} \\ MAX\{Y(g') | g' \text{ is a child of g}\} & \text{if g is a max node} \end{cases}$$

3.1 Errors Caused by Ignoring Traps

This section compares the performances of three evaluation functions, N(g), Y(g), and R(g). R(g) chooses nodes randomly by assigning random values to tip nodes, and then performing minimax. Y(g) and N(g), on the other hand, consider the number of 1's on a board as a measure of its strength. They differ in only one respect: Y(g) introduces nodes with completely accurate values in mid-tree. How often will this correct a mistake that N(g) would make? Define an incorrect decision as the selection of a non-trap node as the best (max or min) child of a given parent despite the existence of a winning trap *. Clearly, N(g) and Y(g) will choose the same child of any parent with no traps among its children. If there is a win trap, Y(g) will always (correctly) choose it. N(g), which does not look for traps, may or may not. To determine the effect of increased search depth on the probability that N(g) and R(g), which do not check for traps, will find them, consider the following scenario: On her first move, H looks ahead k levels in the tree, (k even), and evaluates square boards using N(g). What is the probability that she will miss a trap containing a row of 1's? (Analogous arguments can be applied to all other cases, namely varying V's lookahead, evaluating rectangular boards, and missing other trap patterns).

Let $S = B^{D-k/2-1}$ be the number of rows (and columns) in g, where g is a board at level k in the game tree.

Let p represent the probability that a 1 was placed in a given square in the original board.

Then $P \stackrel{\text{def}}{=}$ Pr[g is a trap containing a row of 1's]
$= (1 -$ Pr[all rows in g contain at least one 0])
$= (1 -$Pr[a given row contains at least one 0]^S)
$= (1 - (1 -$ Pr[a given row is all 1's]$)^S$)
$= (1 - (1 - p^S)^S)$.

By definition, if an incorrect decision was made by an evaluation function F at level k, there must be some node at level (k-1), G, with a child of maximum value among its trap children, g_t, and one of maximum value among its non-trap children, g_{nt}, for which $F(g_{nt}) > F(g_t)$. The probability of an error being made by an evaluation function F looking ahead to level k, then, can be expressed as:

---

* Other reasonable defintions are possible. Several authors [Pearl '83] [Nau '83a] consider a decision incorrect only if a "loss" node was chosen when a "win" was available. The definition used here considers a decision incorrect if a node of unknown exact value is chosen when a "win" trap should have been recognized.

$$E(F,k) = \sum_{i=1}^{B-1} Pr[\text{G has exactly i trap children}] Pr[F(g_{nt}) > F(g_t)].$$

An incorrect decision made by F at level k, however, does not necessarily imply that H will miss an opportunity to force a win; the minimax procedure might have labeled that trap as inaccessible to H, anyway. For the trap to have been accessible to H after passing through a min level, say (k-1), F must have missed a trap child of every max child of the min node. In other words, $E(F,k-1) = [E(F,k)]^B$. To be accessible after a max level, an error must have been made on at least one min child. Thus, $E(F,k-2) = (1-(1-E(F,k-1))^B)$. This recurrence relation alternates up the tree, until $E(F,0)$ indicates the probability that H should have been able to guide the game towards a forced win, but, because of F's inability to recognize traps, did not. (This mistake need not cost H the game. H should, however, already have a guaranteed win, but, because of this error, does not. The game's outcome will only change if V is able to capitalize on this mistake.) If $E(F,0)$ is an increasing function of search depth, the number of opportunities that H will miss increases with search depth. If V is equally likely to capitalize on any given error, the more opportunities V has, the more games H will lose. This is exactly the definition of pathology: as H searches deeper, V wins more games. Thus, pathology can be expected if $E(F,0)$ increases with search depth. Differentiating $E(F,k-j)$ with respect to $E(F,k-j+1)$ gives a positive expression for all j. In other words, $E(F,0)$ is an increasing function of search depth if and only if $E(F,k)$ is. If poor trap recognition is, in fact, responsible for pathology, any function F in the range of k's for which $\frac{d}{dk}E(F,k) > 0$ should behave pathologically.

In the calculation of $E(F,k)$, only $Pr[F(g_nt) > F(g_t)]$ is dependent on F. The probability that G's children include exactly i traps is the same regardless of which function is used.

Let $Pr[I]$ =Pr[G's children include exactly i traps]=$\binom{B}{i} p^i (1-p)^{B-i}$.

Let the probability that F returns a non-trap, given that G has i trap children, be denoted by $Pr[FNTI] = Pr[F(g_{nt}) > F(g_t) | \text{G has exactly i trap children}]$.

Then $E(F,k) = \sum_{i=1}^{B-1} Pr[I] Pr[FNTI] = \sum_{i=1}^{B-1} \binom{B}{i} p^i (1-p)^{B-i} Pr[FNTI]$.

R(g), which chooses nodes randomly, returns non-traps with probability equal to the proportion of non-trap children, $Pr[RNTI] = \frac{B-i}{B}$.

Thus, $E(R,k) = \sum_{i=1}^{B-1} \binom{B}{i} p^i (1-p)^{B-i} (\frac{B-i}{B})$
$= \sum_{i=1}^{B-1} \binom{B}{i} p^i (1-p)^{B-i} - \frac{1}{B} \sum_{i=1}^{B-1} i \binom{B}{i} p^i (1-p)^{B-i}$
$= \sum_{i=0}^{B} \binom{B}{i} p^i (1-p)^{B-i} - (1-p)^B - p^B - \frac{1}{B} [\sum_{i=0}^{B} i \binom{B}{i} p^i (1-p)^{B-i} - Bp^B]$
$= (1-p) - (1-p)^B$.

Intuitively, this means that the board chosen by R(g) is a trap with probability $(1-p)$, and

this is an error unless all B boards were non-traps. This function can be shown to increase with search depth, (that is, $\frac{d}{dk}E(R,k) > 0$), and thus, R(g) will behave pathologically.

For N(g), however, the case is not that simple. $N(g_{nt}) > N(g_t)$ implies that there is some value x, $(S+1) \leq x \leq (S^2 - S)$, such that $N(g_{nt}) = x$, and $N(g) < x$ for all traps. Setting up the i traps with a maximum of (x-1) 1's means first choosing a row to contain all 1's, then setting up the remaining 1's in the other (S-1) rows, or

$$[Sp^S \sum_{j=0}^{x-S-1} \binom{S^2-S}{j} p^j (1-p)^{S^2-S-j}]^i.$$

To calculate Pr[NNTI], then, choose one of the (B-i) non-traps, set it up with x 1's, and set up all i traps with fewer than x. Thus, $E(N,k) = \sum_{i=1}^{B-1} Pr[I] Pr[NNTI]$, where

$$Pr[NNTI] = (B-i) \sum_{x=S+1}^{S^2-S} \binom{S^2}{x} p^x (1-p)^{S^2-x} [Sp^S \sum_{j=0}^{x-S-1} \binom{S^2-S}{j} p^j (1-p)^{S^2-S-j}]^i.$$

This expression is rather difficult to analyze directly because of the triply nested incomplete binomial distributions. However, a strict upper bound can be obtained by completing the two innermost distributions, and multiplying each term by $(Sp^S)^{1-i}$, which is never less than 1. Similarly, a strict lower bound can be obtained by selecting individual terms from the two innermost distributions, and multiplying the result by a number smaller than 1. Specifically, set $j = 0$, and $x = S^2 - S$, multiply each term by $(Sp^S)^{B-i}$ (always less than 1), and replace $\binom{S^2}{S^2-S}$ with the smaller term, S^S. (Many other bounds are possible, and these are by no means the tightest. Although they may seem artificial, they were chosen because they can be analyzed easily in terms of $E(R,k)$, and are tight enough to illustrate the behavior of $E(N,k)$). This gives the series of inequalities: $Pr[UBNTI] > Pr[NNTI] > Pr[LBNTI]$, where

$$Pr[UBNTI] = (B-i)Sp^S \quad \text{and}$$
$$Pr[LBNTI] = (B-i)S^{S+B} p^{S^2-S+SB} (1-p)^{BS^2-BS+S}.$$

Both $E(UB,k) = Sp^S \sum_{i=1}^{B-i} Pr[I](B-i) = (Sp^S B)E(R,k)$

and

$$E(LB,k) = S^{S+B} p^{S^2-S+SB} (1-p)^{BS^2-BS+S} \sum_{i=1}^{B-i} Pr[I](B-i)$$
$$= (S^{S+B} p^{S^2-S+SB} (1-p)^{BS^2-BS+S} B)E(R,k)$$

are increasing functions of search depth. Although this fact in and of itself is insufficient to prove that $E(N,k)$ is an increasing function as well, examining the endpoints gives some hint to $E(N,k)$'s behavior. If H can see only to her next move, then

$k = 0, S = B^{D-1}, P = (1 - (1 - p^{B^{D-1}})^{B^{D-1}})$, and

$$E(UB, k)_{k=0} = B^D p^{B^{D-1}}[(1 - p^{B^{D-1}})^{B^{D-1}}) - (1 - p^{B^{D-1}})^{B^D}].$$

If, on the other hand, H can see all the way down to her next-to-last move, $k = 2(D-1), S = B, P = (1 - (1 - p^B)^B)$, and

$$E(LB, k)_{k=2(D-1)} = Bp^B[(1 - p^B)^B - (1 - p^B)^{B^2}].$$

Combining these values with the aforementioned inequalities gives :

$$E(N, k)_{k=0} < E(UB, k)_{k=0} < E(LB, k)_{k=2(D-1)} < E(N, k)_{k=2(D-1)}.$$

Thus, there must be some values of k for which $E(N, k)$ is an increasing function, and N(g) can be expected to behave pathologically in that range.

To summarize, this section claimed that trap detection is an integral part of a good, nonpathological evaluation function. The connection between performance and trap detection was illustrated by showing that an evaluation function, N(g), which is known to become less likely to win as search depth increases, also becomes less likely to guide play towards winning traps. The connection between these two is straghtforward. A failure to force wins when the opportunity arises gives the opposing player more chances to win. The next section describes a series of experiments which shows that an evaluation function that recognizes these traps, Y(g), does not behave pathologically.

3.2 Curing Pathology by Recognising Traps

Game tree pathology is an observed phenomenon. Even for board splitting, no definite criteria have been developed for predicting exactly when minimaxing N(g) will behave pathologically. The previous section used Pearl's conjecture that pathology is due to the absence of traps to identify a flaw in N(g), its inability to recognize certain patterns as obvious wins or losses. A new evaluation function, Y(g), recognizes those configurations. The probability that an evaluation function that does not recognize traps will err was shown to be an increasing function of search depth. Y(g), by identifying traps, avoids these errors.

Y(g)'s ability to recognize these patterns indicates that it should outperform N(g); it does not prove that Y(g) is nonpathological. It is altogether conceivable that because Y(g) recognizes only a few select patterns as forced wins, it will behave pathologically as well. In fact, because pathology is an *observed* phenomena, it is impossible to *prove* that Y(g), or any evaluation function on any game, for that matter, will never behave pathologically. In particular, it is interesting to observe the behavior of Y(g), because the probability with which a node is a trap is smaller than the critical trap density, q_c. P, defined in the previous section, is the probability that a given board does not contain a row of 1's. Thanks to symmetry, the probability that a given board does not contain a column of 0's is also equal to P. Since there are fewer diagonals than rows or columns, the probability of a diagonal trap is less than P. All told, Y(g) introduces traps with a density of less than $3P$. This density reaches a maximum when S is at its smallest, or $S = B$. Even at this level,

$$3P = 3(1 - (1 - p^B)^B) < 1 - \frac{(1+B)^{1-(1/B)}}{B} = q_c.$$

Nevertheless, it can be shown experimentally that for several cases for which N(g) behaves pathologically, Y(g) does not.

The outcome of the following experiment is shown in figures 1 through 6. For a fixed B and D, 100 random games were generated. In each instance, one player saw only her possible next moves (lookahead fixed at 0), while the other player looked ahead k moves. K was varied by 2's, either from 0 to 2(D-1), or from 1 to (2D-3). When the lookahead length was even, tips were MAX nodes (square boards for H, rectangular boards for V). When odd, the tips were MIN nodes (rectangular for H, square for V). To insure that neither player could see the endgame too early, lookahead was always cut off at H's next-to-last move, level 2(D-1) in the original game tree. (In other words, when k=2(D-1), the player with the k-move lookahead evaluated tip nodes at level 2(D-1) until play actually reached that level, at which point both players were allowed to view the end-game simultaneously). For each lookahead length, the same 100 games were played, with both players using the same evaluation function, first N(g), then Y(g). The results of this experiments are shown for three pair of B and D.

The graphs of figures 1 through 6 show the playing quality of the player whose looka-

head depth varied, by plotting lookahead depth (x-axis) vs. number of victories out of a maximum of 100 (y-axis). Nonpathological behavior is characterized by monotonically nondecreasing curves; looking ahead further is never a loss. The major result is that Y(g) did not behave pathologically for any of the cases tested. At first glance, however, this may not appear terribly impressive. After all, even N(g), which becomes increasingly likely to miss traps as lookahead grows, wasn't pathological in all cases (see figures 1 and 2). It was not until the board size reached 3^6-by-3^6 that pathology appeared. Perhaps Y(g) will behave pathologically for some values of B and D larger than those tested. A response to this charge, then, requires a second glance at the graphs. Even when nonpathological, N(g) was a rather weak evaluation function. Given an initial eight move lookahead in a ten move game, H was still only able to win 66% of the time (figure 1). Y(g), on the other hand, was always quite strong — given the same eight move lookahead using Y(g), H had a 90% winning record. Furthermore, as B and D increased, the number of games won using N(g) with the maximum lookahead depth stayed about the same (66,66,65,65,70,66). When Y(g) was used, the victory margin increased steadily, to the point where V won 99 times by looking ahead 7 moves in a 10 move game (figure 5). These two facts indicate that far from becoming pathological as the game size grows, Y(g) becomes stronger.

The critical trap density, however, indicates that Y(g) should be pathological. How could Y(g) beat the theoretical minimum? An explanation may lie in the very definition of Y(g). The analysis in [Pearl '84] started with a tree with uniform win depth, and showed that minimax would behave pathologically. To bring this model closer to real games, the uniform win *depth* constraint was dropped, and every node in the tree was made a trap with constant probability q. Pathology was shown to disappear at $q = q_c$. The implicit model used in section 3.1 to define Y(g) goes one step further: it drops the requirement of uniform win *density,* and makes each node a trap with probability $q(depth)$, where q is now a function of depth in the tree. Thus, deeper searches not only uncover *more* traps, they uncover traps *with greater probability*. This is almost certainly true in a game like chess; there is a much greater density of 50 move checkmates than of 4 move checkmates. As the models come closer to approximating real games, then, the trap density required to avoid pathology decreases. This strengthens the claim that a failure to recognize traps, which would be a disaster in any real game, is what causes pathological behavior of evaluation functions on simplified models.

4. Conclusions

The theoretical prediction of game tree pathology in [Nau '83b], and the subsequent observation of pathological behavior in board splitting in [Nau '82], raised an obvious question: What characteristics of game trees cause pathological behavior? Two plausible answers have been posited, independence among sibling nodes [Nau '82] [Nau '83a], and the absence of traps [Pearl '83]. This paper examined the relationship between traps and pathology in board splitting. The pathological behavior of N(g), a function which has been shown to evaluate individual boards fairly accurately [Nau '82], was explained. Although it performed well on most individual boards, N(g)'s inability to recognize traps doomed it to frequently missing what should have been an obvious best choice. This type of error becomes increasingly likely as search depth increases. Each time a forced win is missed, the opposing player is given an opportunity to win a game that should have already been lost. As search deepens and the number of these errors increases, so does the probability that the game's outcome will be changed, thereby predicting pathological behavior. A modified evaluation function, Y(g), was designed to combat these errors. It did this by recognizing certain mid-game setups as wins and losses. For several cases in which N(g) behaved pathologically, Y(g) did not. Furthermore, Y(g) avoided pathological behavior despite introducing traps with a density smaller than the theoretically predicted minimum. It was able to do this by distributing the traps nonuniformally throughout the tree, so that more traps appeared at deeper levels. These results represent the first empirical evidence of the importance of traps to the avoidance of pathology, and strengthen the claim that there is a causal nature between them.

Acknowledgements

I would like to thank Mordechi Yung, Eugene Pinsky, and my advisor, Richard Korf, for their helpful discussions and suggestions.

Bibliography

[Berliner '79] H. Berliner, "The B^* Tree Search Algorithm: a Best-first Proof Procedure," *Artificial Intelligence* **12** (1979), 23-40.

[LR '67] R.D. Luce and H. Raiffa, *Games and Decisions*, John Wiley and Sons, New York, 1967.

[Nau '82] D.S. Nau, "An Investigation of the Causes of Pathology in Games," *Artificial Intelligence* **19** (1982), 257-278.

[Nau '83a] D.S. Nau, "Pathology on Game Trees Revisited, and an Alternative to Minimaxing," *Artificial Intelligence* **21** (1983), 221-244.

[Nau '83b] D.S. Nau, "Decision Quality as a Function of Search Depth on Game Trees," *JACM* **30,4** (October 1983), 687-708.

[NauPT '83] D.S. Nau, P. Purdom, C.H. Tzeng,"Experiments on Alternatives to Minimax," University of Maryland, October 1983.

[Nilsson '80] N.J. Nilsson, *Principles of Artificial Intelligence* Tioga, Palo Alto, California 1980.

[Pearl '83] J. Pearl, "On the Nature of Pathology in Game Searching," *Artificial Intelligence* **20** (1983), 427-453.

[Pearl '84] J. Pearl, *Heuristics*. Addison-Wesley, Reading, Massachusetts, 1984.

[Shannon '50] C.E. Shannon,"Programming a Computer for Playing Chess," *Philosophical Magazine* **41** (1950), 256-275.

[TP '83] C.H. Tzeng and P.W. Purdom,"A Theory of Game Trees," *Proceedings of the National Conference of Artificial Intelligence, AAAI*, 1983, 416-419.

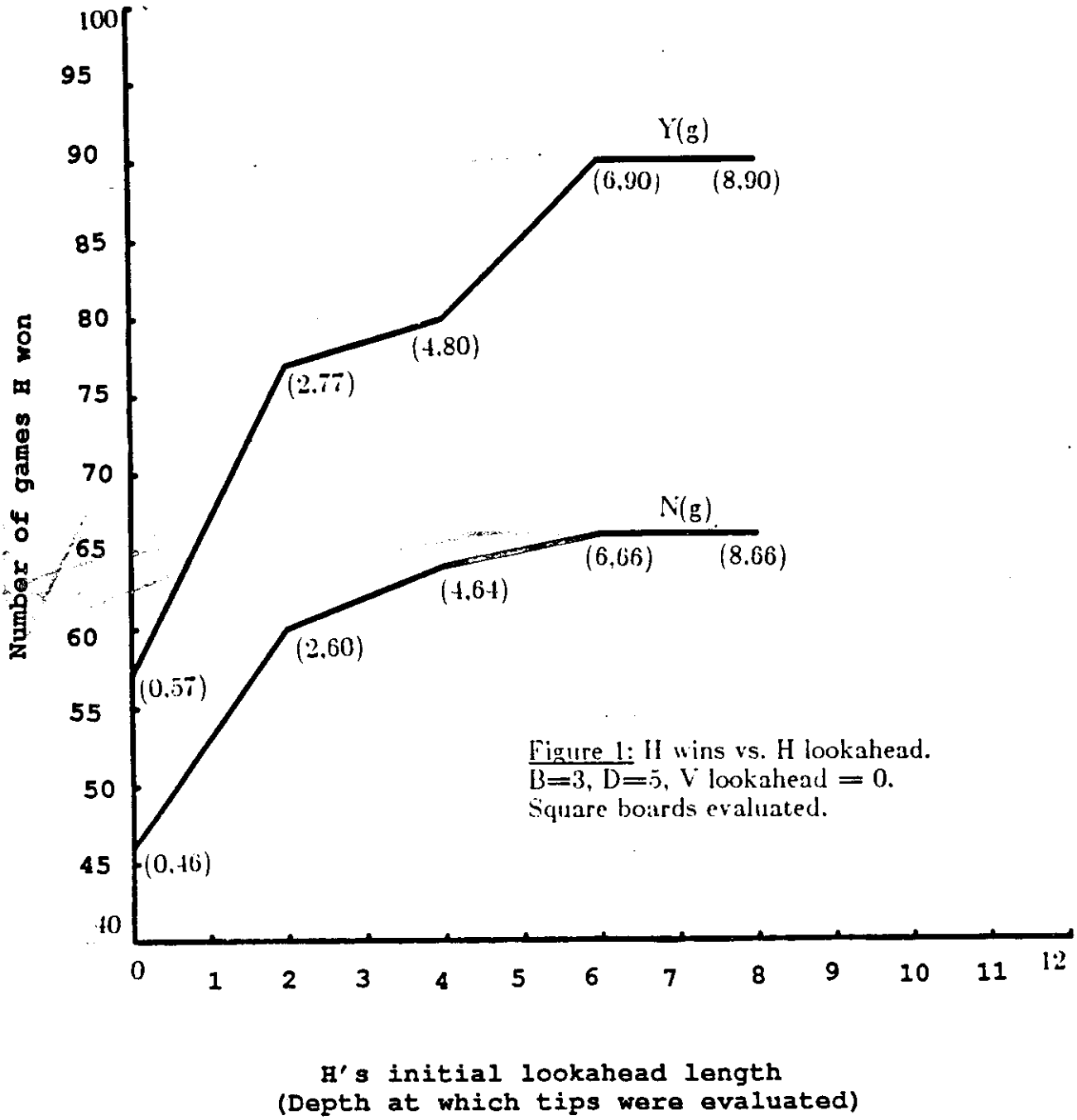

Figure 1: H wins vs. H lookahead.
B=3, D=5, V lookahead = 0.
Square boards evaluated.

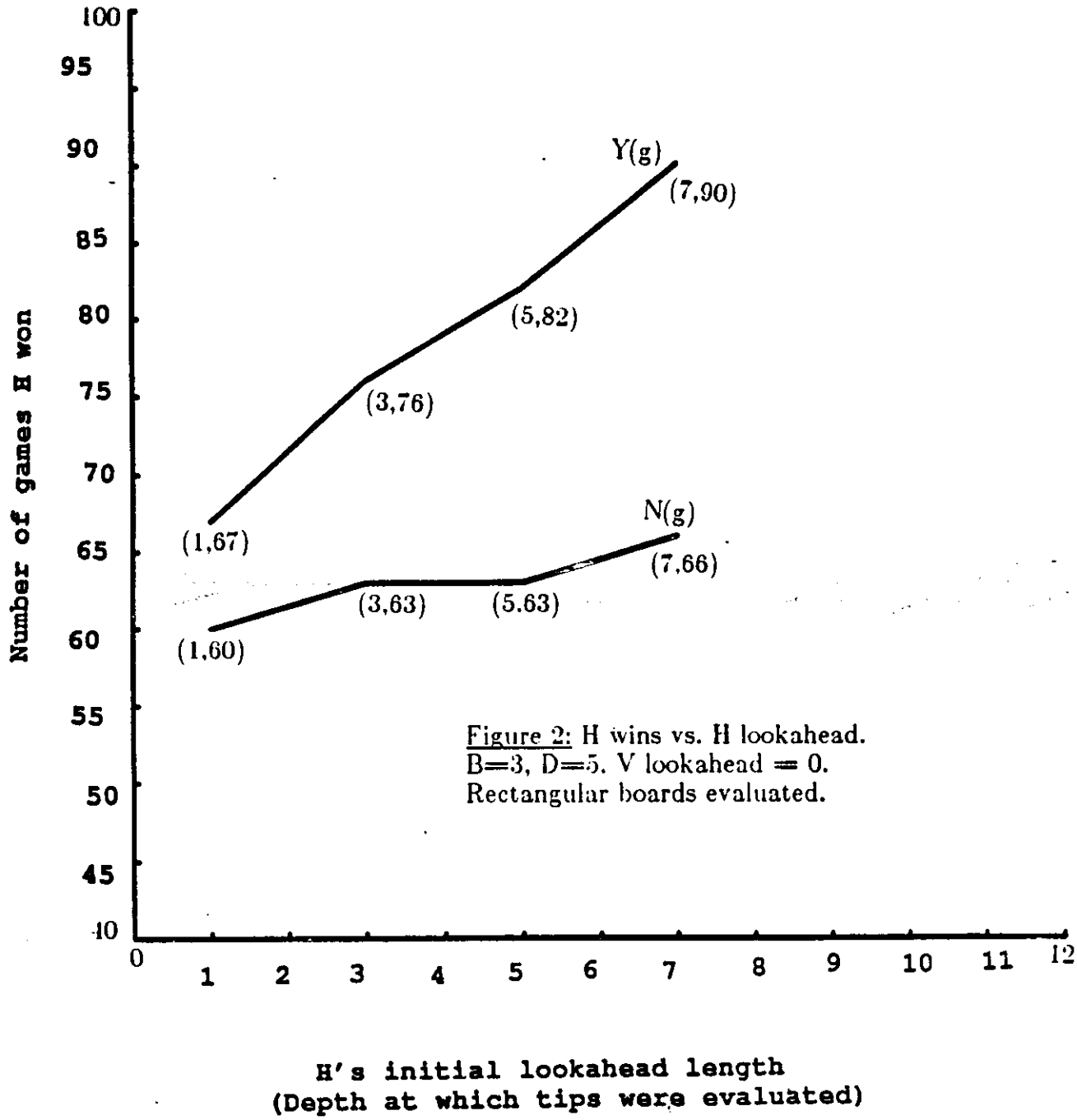

Figure 2: H wins vs. H lookahead.
B=3, D=5. V lookahead = 0.
Rectangular boards evaluated.

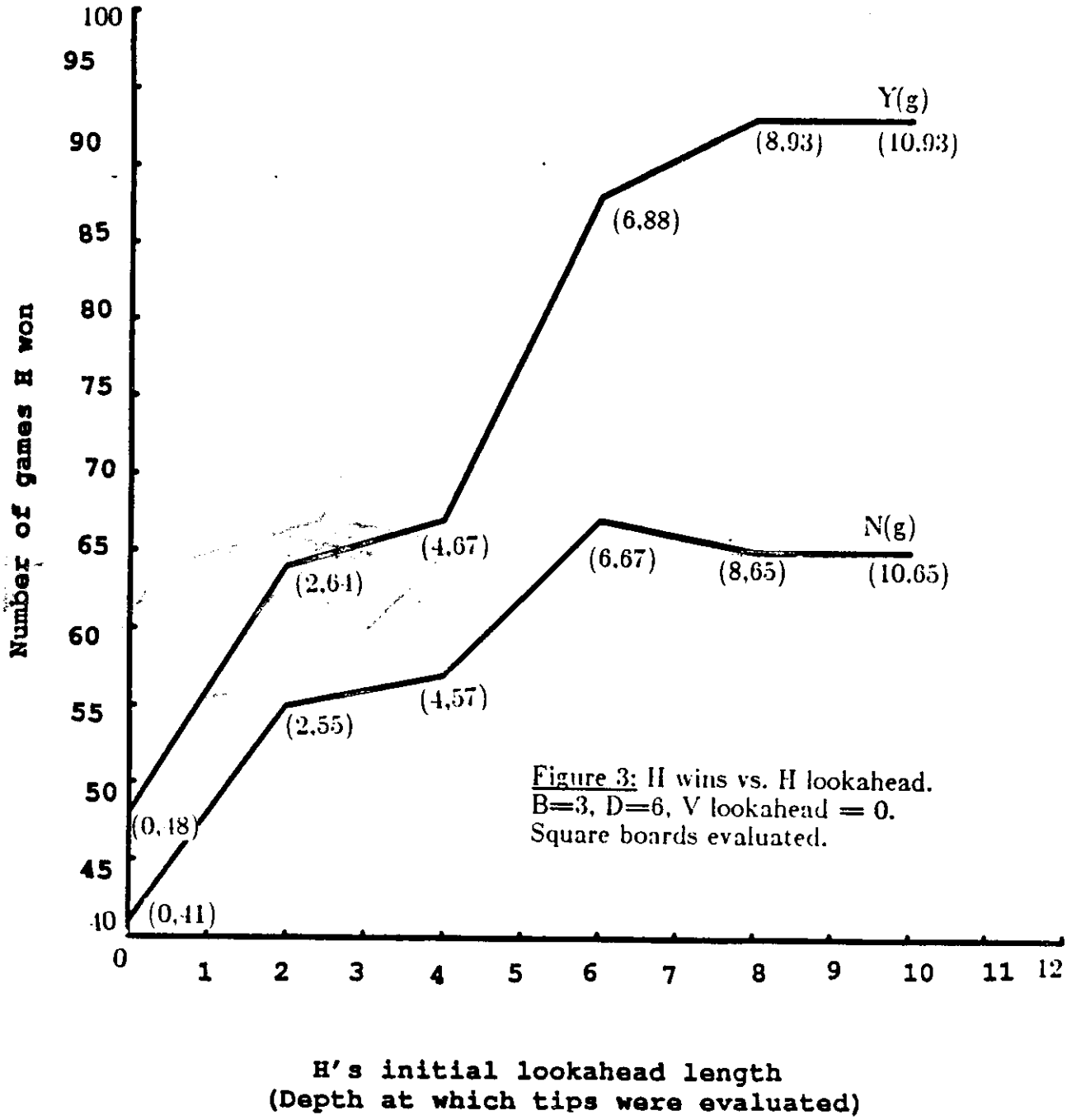

Figure 3: H wins vs. H lookahead.
B=3, D=6, V lookahead = 0.
Square boards evaluated.

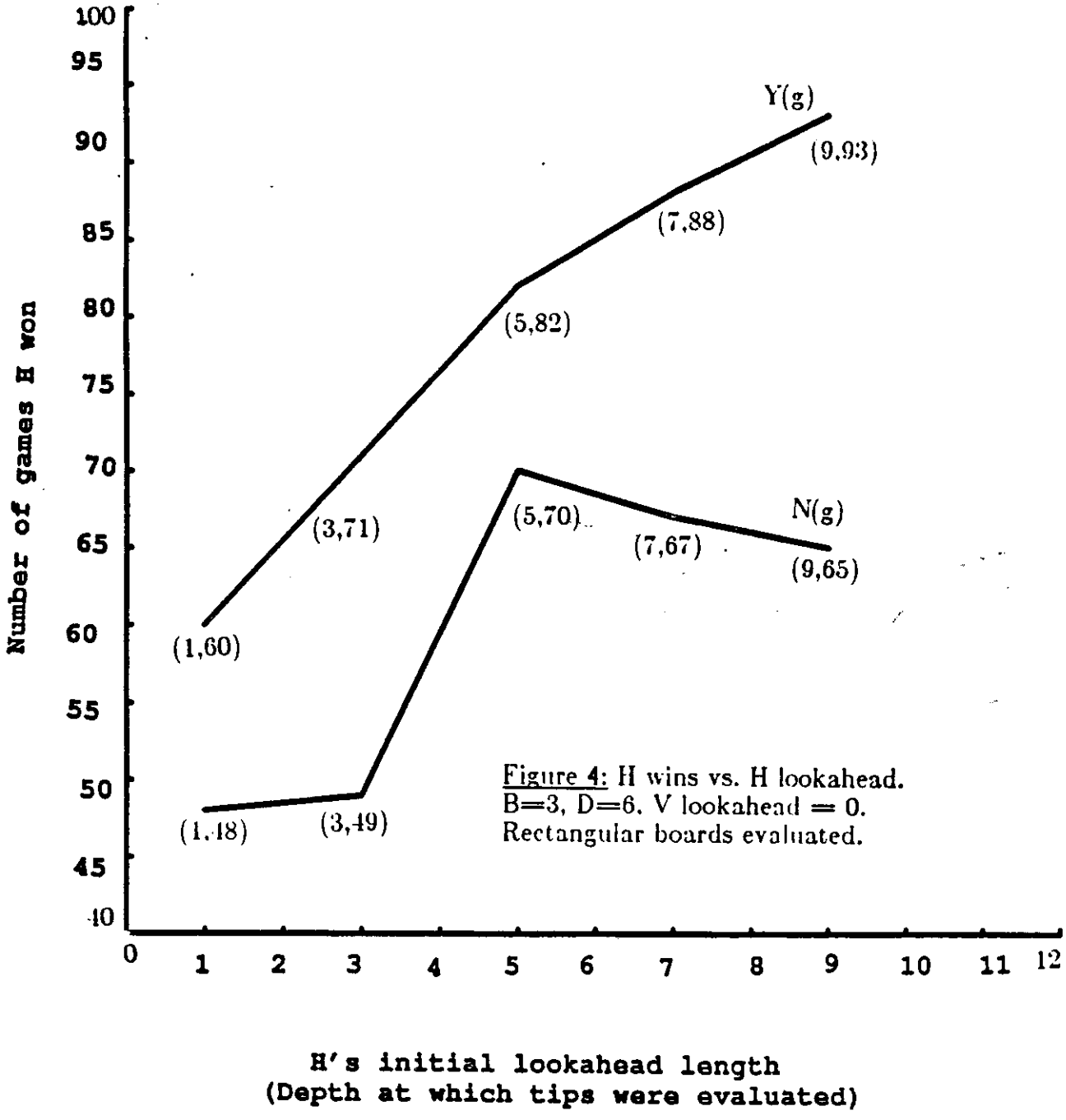

Figure 4: H wins vs. H lookahead.
B=3, D=6. V lookahead = 0.
Rectangular boards evaluated.

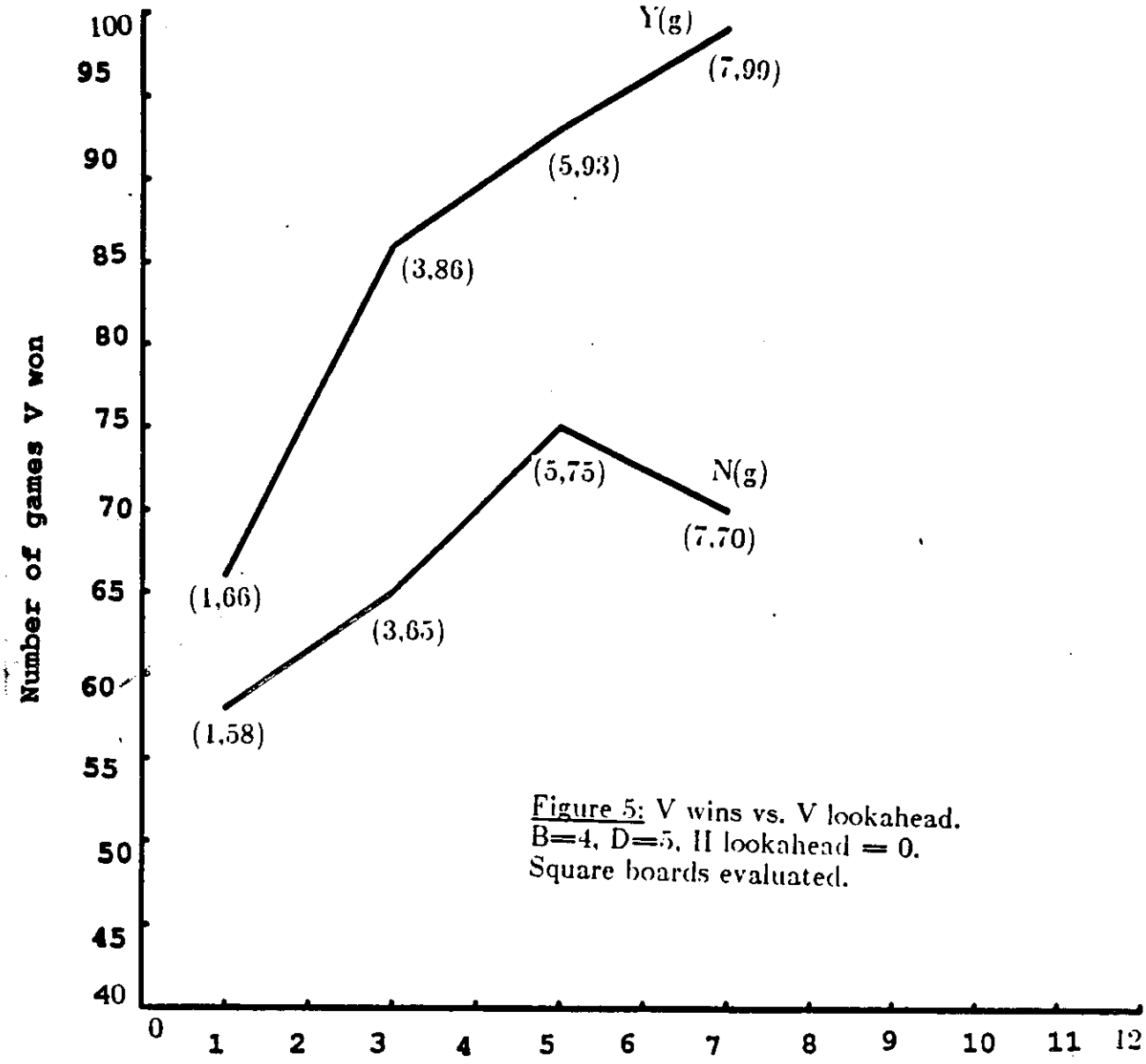

Figure 5: V wins vs. V lookahead.
B=4, D=5, H lookahead = 0.
Square boards evaluated.

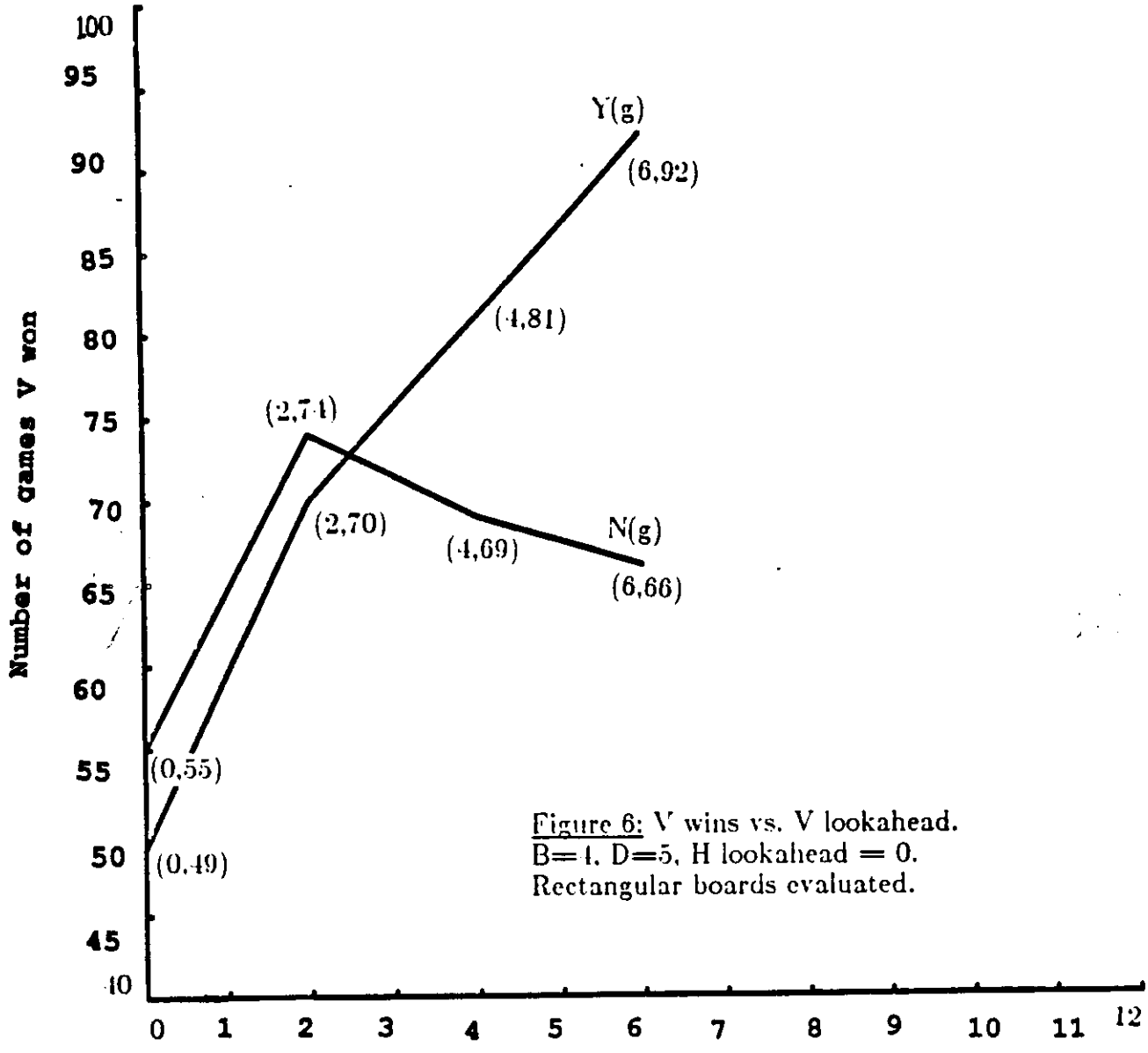

Figure 6: V wins vs. V lookahead.
B=4, D=5, H lookahead = 0.
Rectangular boards evaluated.